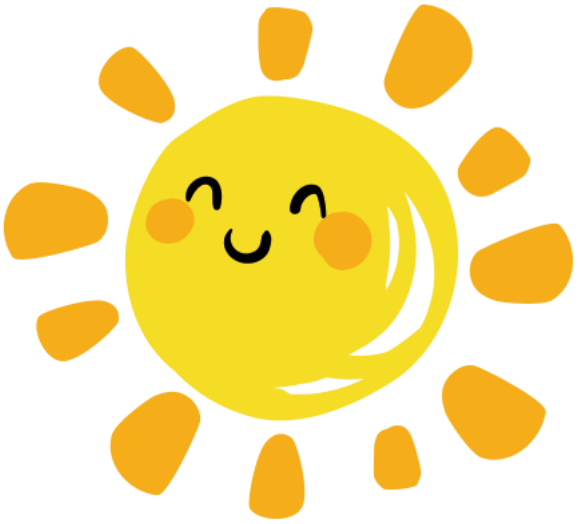

PRACTICA 1

Introducción a los Hilos y aplicaciones

Profesor: Gilde Valeria Rodríguez Jiménez.
email: gildevroji@gmail.com

Ayudante de teoría: Hermilo Cortés González.
email: hermilocg@hotmail.com

Ayudante de teoría: Rogelio Alcantar Arenas.
email: rogelio-aa@ciencias.unam.mx

Ayudante de laboratorio: Hermilo Cortés González.
email: hermilocg@hotmail.com

Ayudante de laboratorio: Luis Angel Leyva Castillo
email: luis_angel_howke@ciencias.unam.mx

FACULTAD DE CIENCIAS, UNAM
Computación Concurrente

Introducción

Los sistemas operativos modernos se basan en el principio de la multiprogramación, es decir, en la posibilidad de manejar distintos hilos de ejecución de manera simultánea con el objetivo de paralelizar el código e incrementar el rendimiento de la aplicación.

Esta idea también se plasma a nivel de lenguaje de programación. Algunos ejemplos representativos son los APIs de las bibliotecas de hilos Pthread, Win32, Java, C, C++, C#, Go, Ruby, etc. Incluso existen bibliotecas de gestión de hilos que se enmarcan en capas software situadas sobre la capa del sistema operativo, con el objetivo de independizar el modelo de programación del propio sistema operativo subyacente.

La idea de hilo surge de la posibilidad de compartir recursos y permitir el acceso concurrente a esos recursos. La unidad mínima de ejecución pasa a ser el hilo, asignable a un procesador. Los hilos tienen el mismo código de programa, pero cada uno recorre su propio camino con su PC, es decir, tiene su propia situación aunque dentro de un contexto de compartición de recursos.

Informalmente, un hilo se puede definir como un proceso ligero que tiene la misma funcionalidad que un proceso pesado, es decir, los mismos estados. Por ejemplo, si un hilo abre un fichero, éste estará disponible para el resto de hilos de una tarea.

Las ventajas de la programación multihilo se pueden resumir en las tres siguientes:

- Capacidad de respuesta, ya que el uso de múltiples hilos proporciona un enfoque muy flexible. Así, es posible que un hilo se encuentra atendiendo una petición de E/S mientras otro continúa con la ejecución de otra funcionalidad distinta. Además, es posible plantear un esquema basado en el paralelismo no bloqueante en llamadas al sistema, es decir, un esquema basado en el bloqueo de un hilo a nivel individual.
- Compartición de recursos, posibilitando que varios hilos manejen el mismo espacio de direcciones.
- Eficacia, ya que tanto la creación, el cambio de contexto, la destrucción y la liberación de hilos es un orden de magnitud más rápida que en el caso de los procesos pesados. Recuerde que las operaciones más costosas implican el manejo de operaciones de E/S. Por otra parte, el uso de este tipo de programación en arquitecturas de varios procesadores (o núcleos) incrementa enormemente el rendimiento de la aplicación.

En este PDF se entenderan algunas maneras del como se utilizan los Hilos en el lenguaje de Programación de Java.

Se anexan archivos .java, leelos antes de comenzar la practica, se ejemplifican 2 posibles maneras de usar los hilos, lo que realizan es muy básico pero esto nos ayudara a entender más el funcionamiento.

Implementando la clase Runnable

En esta versión se implementa la clase Runnable, donde el único Método es run(), dentro de run() se ejecutara el comportamiento de los hilos. Podemos hacer un comportamiento casi generico para todos los hilos ejecutados, por lo que si queremos que los hilos hicieran tareas muy especificas, no nos ayuda mucho esta manera, pero se puede llegar a hacer.

Implementando la clase Runnable dentro de Thread

Por otra parte, se puede implementar la clase Runnable dentro del mismo Thread, de esta manera se puede asignar el comportamiento de cada hilo un tanto más manual y más acorde a lo que queramos hacer. Por ejemplo, si quisieramos que un hilo sumara los numeros pares del 0 al 100 y el otro los número impares, podriamos usar esta manera.

Practica

1.- Introducción y jugando con Hilos

Modifica el codigo de Hilos.java, de tal manera que se haga lo siguiente:

- Genera una lista de Hilos.
- Agrega 10 hilos a esta mediante un for e inicialisandolos.
- Hazle join a cada hilo con un for o foreach
- Agrega una captura de pantalla a tu reporte de como quedo tu codigo al final.

2.- Multiplicación de Matrices Secuencial

Realiza un programa que realice la multiplicación de matrices.

Considera que seran matrices de tamaño n x n

De entrada seran 2 matrices de tamaño n x n y el retorno sera la matriz resultante.

3.- Multiplicación de Matrices Concurrente

Es posible realizar el producto de dos matrices usando varios hilos de forma independiente. Primero recordemos cómo se obtiene el producto de dos matrices:

Sean A = (a_{ij}) una matriz de n x r y B = (b_{ij}) una matriz de r x m. La matriz producto C = AB es una matriz de tamaño n x m que está definida por las entradas

$$c_ij = \sum_{k=1}^{r} a_{ik} * b_{kj}$$

Es decir, el elemento c_{ij} es igual al producto punto entre la i-ésima fila de A y la j-ésima columna de B. Directamente de esta definición obtenemos un algoritmo secuencial para calcular AB, donde el cálculo principal es el producto punto, que ser vería simila al siguiente código:

```
For (k = 0; k< r; ++k){
        C[i][j] += A[i][k] * B[k][j]
}
```

Para calcula el producto de forma concurrente usando varios hilos de ejecución, podemos asignar parejas de la forma (i,j) a cada hilo y este se encargará de obtener la entrada c_{ij}.

Para el caso de una matriz de n x n tenemos la opción de partir en bloques (submatrices) y luego hacer que cada hilo se encargargue de calcular las entradas de un bloque.

En pocas palabras una manera de resolverlo es asignarle un hilo a cada renglon y que este se ejecute.

Realiza un programa que realice la multiplicación de matrices con hilos.

Considera que seran matrices de tamaño n x n

De entrada seran 2 matrices de tamaño n x n y el retorno sera la matriz resultante.

4.- Contador Compartido: Synchronized y Durmiendo Hilos

En el archivo Contador.java, nos encontramos con lo que viene siendo un contador, donde varios hilos acceden a el para ir actualizando su valor, en esta actividad el objetivo es que el valor del contador sea 20000 y 30000.

Lo primero que deben hacer es ejecutar el programa varias veces y ver que valores da, ¿Qué puedes decir de este resultado?, ¿Por qué lo da?.

Utiliza lo que viene siendo Thread.sleep(<Tiempo de mimir>) para dormir el hilo que selecciones, esto con el fin de realentizar al Hilo, ¿Por qué crees que hariamos esto?.

Finalmente, inveztiga para que funciona synchronized (Pues lo utilizaremos durante el curso), descomenta el bloque de codigo y vuelvelo a ejecutar. Dada tu investigación, ¿Qué sucede cuando el bloque tiene synchronized?.

Descomenta el siguiente bloque de codigo (comentando ahora este) y ejecutalo nuevamente, ¿Qué diferencia ves? ¿Por qué sucede esto?.

Genera algun ejemplo que se te venga a la mente donde se use sinchronized.

Al final, lo que entregaras de esta sección:

- El contador obteniendo 20000 (o aproximado) usando Thread.sleep().
- El contador obteniendo 30000 (o aproximado) usando Thread.sleep().
- El ejemplo usando synchronized que crearon.

TEORÍA

1. ¿Por qué se pone un InterruptedException en el método main?
2. ¿Para que sirve el método Join?
3. ¿Qué pasa si no le hacemos Join a los hilos?
4. Explica de manera consisa como usar Hilos extendiendo la clase Thread.
5. ¿Cuáles son las ventajas en implementar Runnable contra extender de Thread?
6. ¿Se puede predecir el orden en el que se imprimira el mensaje de la clase Hilos?
7. En el archivo Hilos2.java, ¿Qué pasa si sacamos la instancia de la clase “h” de t1, es decir, poner h por ejemplo, antes de declar t1?
8. Explica como podriamos tener comportamientos diferentes implementando Runnable
9. Para la sección 3 realiza lo siguiente:
    a. Se incluyen una carpeta con 3 tipos de matrices, 10x10, 100x100 y 1000x1000, para cada prueba ejecutala y obtene el tiempo obtenido, tanto su versión secuencial asi como su versión concurrente, ¿Ves algun cambio significativo? ¿Mejoro el tiempo o empeoro?
    b. Genera una grafica con tus resultados obtenidos, usando 1,10 y 100 hilos.
    c. Finalmente pondremos a prueba la teoria , pues realizaremos la siguiente tabla con la info siguiente:

# Hilos	Aceleración Teórica	Aceleración Obtenida	% Código en paralelo

    d. En la columna de Aceleración Teórica, utilizaremos la Ley de Amdahl y la compareremos con la Aceleración obtenida y responde la siguientes preguntas.
    e. ¿Se cumple siempre dicha ley?
    f. ¿En que casos no se cumple?

g. ¿Por qué crees que pasa esto?
h. ¿Cuál seria el nivel máximo de mejora?
i. Para obtener una mejora aun mayor, en terminos de Hardware, ¿Qué es mejor?, tener muchos mas nucleos con hilos o tener mayor frecuencia de reloj. Justifica.

10. Responde TODAS las preguntas de la sección 4 de la parte practica (7 preguntas)
11. Escribe lo aprendido con esta practica, asi como descubrimientos que obteniste mientras la realizaron.

ENTREGABLE

Las respuestas a las preguntas se realizan en el editor de su elección. Deben de poner las referencias bibliográficas en donde consultaron la información, esta debe de ir en formato APA y en formato PDF.

Debe incluir caratula, en la cual debe incluir el nombre del equipo, integrantes y numero de cuenta.

El codigo realizado debe venir bien documentado, asi tambien con un readme para ejecutar dicho codigo.

Se les dará 0.5 extra si la realizan en LaTeX. (Mandar PDF y .tex con imágenes en caso de usar).

Debe de llevar el siguiente formato:

[Practica1_Nombre del equipo.[zip]

EJEMPLO:

Practica1_EnchiladasDeYakult.zip