Universidad Nacional Autónoma de México
Facultad de Ciencias

## Computación Concurrente

## Práctica 1

Bonilla Reyes Dafne - 319089660
García Ponce Jose Camilo - 319210536
Juárez Ubaldo Juan Aurelio - 421095568

# Introducción a la Programación Multihilos

## Descripción de los programas

- `AvailableProcessors.java:`

  Programa para ejecutar `Runtime.getRuntime().availableProcessors()` e imprimir el resultado.

- `DeterminanteConcurrente.java:`

  Programa para obtener el determinante de una matriz $3 \times 3$, extendiendo la clase `Thread` y usando 6 hilos.

- `DeterminanteConcurrenteRunnable.java:`

  Programa para obtener el determinante de una matriz $3 \times 3$, implementando la interfaz `Runnable` y usando 6 hilos.

- `DeterminanteSecuencial.java:`

  Programa para obtener el determinante de una matriz $3 \times 3$ de manera secuencial (sin usar hilos).

- `DeterminanteConcurrente2.java`

  Programa para obtener el determinante de una matriz $3 \times 3$, extendiendo la clase `Thread` y usando 2 hilos.

## Reporte de ejercicios

1. **Lee lo siguiente `https://www.evanjones.ca/software/threading-linus-msg.html` y comparte en máximo 4 líneas de computadora a que se refiere Linus Torvalds con un contexto de ejecución y como se relaciona con la definición en la sección 1 de esta práctica.**

   Un contexto de ejecución es un conjunto del estado de una tarea (el estado de la `CPU`, asignaciones de memoria, los permisos, etc.). Esto se relaciona con la definición en la sección 1 de la práctica al ampliar la idea de que un proceso es un programa en ejecución, proponiendo que tanto procesos como hilos son simplemente variaciones de un `COE` en lugar de entidades separadas.

2. **¿Cuántos hilos tiene disponibles tu computadora?**

   **Ejecuta `Runtime.getRuntime().availableProcessors()`, si son más de uno en el equipo escriban el de cada uno.**

- **Dafne:** 12
- **Camilo:** 4
- **Juan:** 4

3. **Revisa el programa *Determinante concurrente* y responde ¿Cuánto tiempo tarda en ejecutarse?**

   - **Dafne:** 900400ns
   - **Camilo:** 1533179ns
   - **Juan:** 2868500ns

4. **El programa Determinante concurrente está implementado extendiendo la clase `Thread`. Implementa el programa utilizando la interfaz `Runnable`.**

   El programa se implementó en el archivo `DeterminanteConcurrenteRunnable.java`

5. **Implementa el programa Determinante concurrente de forma secuencial y compara el tiempo de ejecución. ¿Es mayor o menor?**

   Es menor, ya que este tomó 1800ns.

6. **Si es menor, utiliza la ley de Amdahl para calcular el porcentaje que se puede paralelizar.**

   Esta pregunta se omite, ya que obtuvimos que el programa secuencial tomó menos tiempo.

7. **Si no es menor, realiza la implementación para dos hilos (en vez de seis) y compáralo con la implementación de seis hilos y con la de uno. ¿Cómo se relacionan?**

   Dado que el programa concurrente fue mayor, entonces realizamos la implementación de este ejercicio en el archivo `DeterminanteConcurrente2.java`. Ahora bien, veamos que la implementación para dos hilos tomo 687800ns, es decir, tomó más que el secuencial, aunque menos que con 6 hilos.

8. **Describe con tus propias palabras en máximo dos líneas para qué sirve el método `join()`. Si no utilizas el método `join()` en *Determinante Concurrente*, ¿sigue funcionando?**

   - El metodo join sirve para asegurarnos que el thread actual espere a que el thread en el que se llamo el metodo `join()`, sea completado antes de realizar la siguiente instruccion.
   - Si removemos `join()` el programa se ejecutara, sin embargo, el resultado podria o no podria ser correcto, ya que podriamos intentar usar `partial` antes de que algun thread haya realizado el calculo, lo que podria traer una race condition y causar que el resultado sea erroneo.