Laboratorio 3: Sistemas Operativos

Profesor: Viktor Tapia
Ayudantes de cátedra: Muryel Constanzo y Nicolás Schiaffino
Ayudantes de Laboratorio: Ian Rossi y Luciano Yevenes

16 de mayo de 2025

1. Reglas Generales

Para la siguiente tarea se debe realizar un código programado en lenguaje C o C++ y Java. Se exigirá que los archivos se presenten de la forma más limpia y legible posible. Deberá incluir un archivo README con las instrucciones de uso y ejecución de sus programas junto a cualquier indicación que sea necesaria, y un archivo MAKE para poder ejecutar el programa.

2. Tarea

Durante la expedición al polo sur el profesor se encuentra con algo que jamás habría imaginado ver en su vida una nave espacial alienígena chocada y justo a su lado un trozo de hielo perfectamente rectangular que ha sido excavado. Notando que está cerca de una base científica noruega, el profesor se dirige allá para conseguir explicaciones.

Una vez llega a la base se da cuenta que está abandonada, y cuando se topa con la oficina del departamento biológico, encuentra una muestra de tejido de una criatura desconocida. Determinado a saber más, decide analizarla, pero claramente no están las condiciones ideales para poder programar un algoritmo, por lo que usando la infraestructura de la base le envía un mensaje a ustedes, delegándoles la tarea de crear un programa que pueda hacer este análisis.

Deben analizar las muestras, que han sido digitalizadas como pares de matrices que deben ser multiplicadas. Ustedes saben que este proceso requiere de un decente grado de computación por lo que conviene separar la tarea en forks o threads, sin embargo ustedes no saben cual de estos dos métodos conviene más.

2.1. Especificaciones

Se solicita implementar dos versiones del programa:

- **Versión en C o C++:** Esta implementación debe hacer uso de *forks* para la creación de procesos hijos, y *pipes* para la comunicación entre ellos.
- **Versión en Java:** Debe emplear *threads* para realizar la paralelización de la multiplicación de matrices.

El objetivo final es comparar el rendimiento de ambas versiones utilizando archivos de prueba provistos o generados, enfocándose principalmente en métricas de tiempo de ejecución. Con base en esta comparación, se deberá elaborar un informe detallado que analice cuál de las dos alternativas resulta más eficiente bajo las condiciones de la tarea.

2.2. Formato del Archivo de Entrada

El programa debe recibir como entrada un archivo de texto plano (`.txt`) que contenga dos matrices a multiplicar. Este archivo debe seguir el siguiente formato:

- La primera línea contiene dos números enteros separados por un espacio, que representan las dimensiones de la primera matriz (`filas columnas`).
- Las siguientes líneas contienen los elementos de la primera matriz, dispuestos fila por fila.
- A continuación, una línea en blanco que separa ambas matrices.
- Luego, una línea con dos enteros separados por un espacio, que indican las dimensiones de la segunda matriz.
- Finalmente, se listan los valores de la segunda matriz, también fila por fila.

A continuación se muestra un ejemplo de archivo de entrada que representa la multiplicación de una matriz de tamaño 2×3 con otra de tamaño 3×2.:

```
2 3
1 2 3
4 5 6

3 2
7 8
9 10
11 12
```

2.3. Implementación con Forks y Pipes

Para la versión desarrollada en `C/C++`, se utiliza la llamada al sistema `fork()` para crear procesos hijos, los cuales se encargan de realizar parte del cálculo de la multiplicación de matrices. Cada proceso hijo computa una o más filas de la matriz resultante, permitiendo aprovechar la ejecución concurrente en sistemas multiprocesador.

La comunicación entre el proceso padre y sus hijos se realiza mediante *pipes*. Cada hijo, una vez terminado su cálculo, envía su resultado parcial a través del *pipe* correspondiente. El proceso padre recopila los fragmentos del resultado y construye la matriz final, la matriz resultante se debe escribir en un archivo de salida llamado **salidaFork.txt**.

2.4. Implementación en Java con Threads

La implementación en `Java` emplea hilos (*threads*) para paralelizar la multiplicación de matrices. Cada `Thread` es responsable de calcular una o más filas de la matriz resultante, dividiendo la carga de trabajo de manera equitativa.

Para la sincronización y el manejo adecuado del acceso a estructuras compartidas, se utilizan arreglos sincronizados, garantizando la coherencia del resultado y evitando condiciones de carrera.

Al finalizar el trabajo de todos los hilos, el hilo principal se encarga de recolectar los resultados parciales y generar la matriz final. Esta se almacena en un archivo de salida llamado **salidaThread.txt**.

2.5. Informe de Comparación y Análisis

El informe debe incluir un análisis comparativo entre las versiones desarrolladas en `C/C++` (usando *forks* y *pipes*) y en `Java` (usando *threads*). Este análisis debe basarse en pruebas reales realizadas por ambos integrantes del grupo, en sus respectivos computadores personales.

Especificaciones de los equipos de prueba

Se deben incluir las siguientes características técnicas para cada computador:

- Modelo del procesador (incluyendo cantidad de núcleos e hilos)
- Memoria RAM disponible
- Sistema operativo utilizado
- Arquitectura del sistema (32 o 64 bits)

Preguntas de análisis

El informe debe responder, como **mínimo**, a las siguientes preguntas:

1. ¿Cuál de las dos implementaciones tuvo un mejor rendimiento en términos de tiempo de ejecución? ¿A qué crees que se debe esto?
2. ¿Se observó alguna diferencia significativa en el uso de recursos del sistema (CPU, RAM) entre ambas soluciones?
3. ¿En qué escenarios consideras más adecuado el uso de procesos (`forks`) frente a hilos (`threads`)?
4. ¿Qué estrategias de optimización aplicarías al programa con menor rendimiento para que iguale o supere la eficiencia del programa mejor evaluado?

Las respuestas deben ser reflexivas y estar fundamentadas en la experiencia real durante la ejecución de las pruebas.

3. BONUS (20 pts extras)

Ante el hallazgo de nuevas muestras biológicas con estructuras internas más sofisticadas, los análisis básicos ya no son suficientes. Las matrices obtenidas a partir de estas muestras contienen patrones que podrían corresponder a formas de organización celular, redes neuronales primitivas, o incluso códigos simbólicos de origen desconocido.

Por esta razón, el profesor ha solicitado una ampliación del sistema actual que permita no solo multiplicar matrices, sino también realizar un análisis más profundo de sus propiedades estructurales y patrones internos. Esta nueva versión debe ser capaz de manejar secuencias encadenadas de operaciones y aplicar filtros que permitan identificar características como simetrías, repeticiones o comportamientos regulares, lo cual podría ser clave para entender la lógica subyacente en estos datos biológicos.

- **Multiplicación en cadena:** El sistema debe ser capaz de multiplicar tres o más matrices consecutivas (`A × B × C ...`), validando que las dimensiones sean compatibles.

- **Análisis estructural:** Una vez obtenida la matriz final, se deberá implementar un algoritmo que detecte si la matriz es simétrica (i.e., $A = A^T$), lo que podría indicar propiedades especiales en los datos analizados.

Formato del Archivo de Entrada (Bonus)

Para la versión extendida del sistema, el archivo de entrada debe contener múltiples matrices que serán multiplicadas en cadena. El formato será el siguiente:

- La primera línea contendrá un número entero n, que indica la cantidad total de matrices a procesar.
- Por cada matriz, se incluirá:
  - Una línea con dos números enteros separados por espacio que representan las dimensiones de la matriz (`filas columnas`).
  - Las siguientes líneas corresponden a las filas de la matriz, con los valores separados por espacios.
- Cada matriz estara separada por una línea en blanco.

A continuación se muestra un ejemplo válido de archivo de entrada para la multiplicación de tres matrices:

```
3
2 3
1 2 3
4 5 6

3 2
7 8
9 10
11 12

2 2
1 0
0 1
```

Este archivo representa la operación:

$$A_{2\times3} \times B_{3\times2} \times C_{2\times2}$$

El sistema deberá validar la compatibilidad de dimensiones entre matrices adyacentes y, en caso de ser inválidas, notificar un error antes de iniciar el procesamiento.

4. README

Debe contener como mínimo:

- Nombre, RoI y Paralelo de los integrantes.
- Especificación de los algoritmos y desarrollo realizado.

- Instrucciones de como compilar y correr el código.
- Supuestos utilizados.
- En caso de realizar el bonus, se deben entregar **dos archivos separados**: uno correspondiente a la versión base de la tarea, y otro con la versión extendida (bonus), cada uno con su propio README si es necesario.

5. Consideraciones Generales

- El uso de librerías está estrictamente limitado a aquellas esenciales para la implementación del laboratorio. Se permite el uso de:
  - Bibliotecas estándar básicas como `<stdio.h>`, `<stdlib.h>`, `<string.h>`, `<math.h>` en C, o `<iostream>`, `<vector>`, `<string>` en C++.
  - `unistd.h` para manejo de procesos y pipes.
  - `pthread.h` para la implementación de hilos.
  - `time.h` o bibliotecas estándar equivalentes para la medición de tiempos de ejecución.
  - Cualquier otra biblioteca de sistema necesaria exclusivamente para la gestión de `fork()`, `pipes` o `threads`.

  No está permitido utilizar librerías externas, de alto nivel o científicas que resuelvan automáticamente operaciones como la multiplicación de matrices, la paralelización o la comunicación entre procesos.

  En caso de requerir alguna librería adicional no listada explícitamente, su uso debe ser consultado previamente con los ayudantes y autorizado formalmente.
- En caso de desarrollar el **bonus**, se deben entregar **dos archivos de código fuente** claramente diferenciados: uno para la versión base del laboratorio y otro para la versión extendida con bonus. Ambos deben ser independientes y ejecutables por separado.
- Se deberá trabajar **OBLIGATORIAMENTE** en parejas.
- Deberá estar subido al Github correspondiente a mas tardar el día Domingo 8 de Junio a las 23:59 horas.
- Se descontarán 10 puntos por cada hora o fracción de atraso.
- La nota máxima obtenible, si se realiza también el bonus, es de 120.
- Las copias serán evaluadas con nota 0 en el promedio de las tareas.
- La tarea debe ser hecha en el lenguaje C o C++ y Java. Se asume que usted sabe programar en este lenguaje, ha tenido vivencias con el, o que aprende con rapidez.
- El código debe ser entregado en forma de **2 archivos**, uno `.cpp` o `.c` y otro `.java`, ambos nombrados en base al formato `"LAB1_ApellidoIntegrante1_ApellidoIntegrante2"`
- Los códigos serán pasados por un software anti-plagio, en caso de ser detectada copia o uso de una IA para la totalidad del Laboratorio, se pondrá nota 0, hasta que se realice una reunión con los grupos involucrados.
- Pueden crear todas las funciones auxiliares que deseen, siempre y cuando estén debidamente comentadas.

- Las tareas serán ejecutadas en Linux, cualquier tarea que no se pueda ejecutar en dicho sistema operativo, partirá de nota máxima 60.
- Las preguntas deben ser hechas por Aula a través del foro o servidor de Discord. De esta forma los demás grupos pueden verse beneficiados también.
- Si no se entrega README o MAKE, o si su programa no funciona, la nota es 0 hasta la recorrección.
- Se descontarán hasta 50 puntos por:
    - Mala implementación del Makefile.
    - No respetar el formato de entrega.
    - No respetar el formato del README.
    - Solicitar edición de código al momento de revisar.
    - Código poco prolijo y mal estructurado (ausencia de indentación adecuada, falta de consistencia en el estilo, nombres de variables confusos o poco descriptivos, comentarios insuficientes o irrelevantes).
- **Una vez publicadas las notas tendrán 5 días para apelar con el corrector que les revisó, después de este plazo las notas no tendrán ningún tipo de cambio**.