Laboratorio 3: Sistemas Operativos

Profesor: Viktor Tapia
Ayudante de cátedra: Juan Pablo Varas y Martín Rodriguez
Ayudante de Tarea: Javiera Cárdenas y Nicolás Toro

Octubre 2023

1 Reglas Generales

Para la siguiente tarea se debe realizar un código programado en lenguaje 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 de sus programas junto a cualquier indicación que sea necesaria y un archivo MAKE para poder ejecutar el programa

2 Enunciado

A partir de lo realizado en el Laboratorio 1, ahora se debe realizar una tarea bastante similar pero aplicando los contenidos ya aprendidos a lo largo del semestre para poder optimizar su búsqueda. Por lo que nuevamente se le pide encontrar palabras en una sopa de letras pero a partir del uso de herramientas nuevas que en archivos de gran tamaño debería reducir el tiempo de búsqueda.

Se le solicita realizar 2 búsquedas en base a cuadrantes utilizando **Threads** y **Fork/Join** de manera recursiva como principales herramientas.

2.1 Instrucciones

La sopa de letras sera una matriz cuadrada de largo variable (10x10, 12x12, etc) en la cual se encuentra la palabra a buscar de la siguiente forma:

```
A G L F F M T O
H J K G A T P E
J T Y U G A T O
G S N O G A T E
K L M T G A T H
L N H G T A T I
Y J G F D A T O
S E D F G T R S
```

Ejemplo de sopa de letras 8x8

La sopa de letras vendrá dentro de un archivo .txt llamado *sopa_de_letras.txt*, el cual estará organizado de la siguiente manera:

1. Primera linea. **N**: Dimensión de la Matriz (NxN)
2. Segunda linea. **P**: Palabra a buscar
3. Tercera linea. **Matrix**: Sopa de letras

Deberán crear un programa en Java que, por un lado utilice la clase Multithreading de manera recursiva y por otro se utilice la clase ForkJoinPool junto con RecursiveTask para encontrar encontrar la palabra. Para ello, se debe dividir la sopa de letras en 4 cuadrantes y crear un proceso para cada uno de ellos. Luego, cada proceso debe repetir lo mismo hasta quedar con cuadrantes cuadrados del largo de la palabra a buscar (**P**). Llegado a ese punto, cada proceso deberá buscar si la palabra se encuentra en su cuadrante y el proceso que lo encuentre debe indicar su posición por consola. Esto ultimo puede ser desde donde hasta donde se encuentra o indicando la posición del comienzo de la palabra.

Por ejemplo, si se tiene una sopa de 8x8 y una palabra de largo $P = 4$ (como el caso de la Figura 1), en cada caso debiesen quedar 4 procesos distintos, cada uno analizando un cuadrante de 4x4. Luego, el proceso que lo encuentre puede indicar la posición en cualquiera de los 2 formatos siguientes:

- fila 12, columna [7,9]
- fila 12, columna 7

Otro ejemplo puede ser una sopa de 20x20 y una palabra de largo $P = 5$ en donde deberían quedar 16 procesos distintos.

Finalmente, también se debe entregar un **excel** que incluya un breve análisis sobre si es eficiente o no la utilización de las herramientas entregadas para este ejercicio probando múltiples escenarios, tales como medir y comparar el tiempo de ejecución en [ms] con hebras, Fork/Join o ninguna de estas herramientas buscando casilla por casilla, agregando al menos un gráfico sobre los tiempos de ejecución de cada uno de estos escenarios. Para esto, también se le solicita incluir la marca, el modelo y frecuencia del procesador de su computador junto con funcionalidades extras que este tenga (Ejemplos: Hyperthreading, Turbo boost o Overclocking), debido a que cada procesador va a presentar un tiempo de ejecución distinto. El formato del excel a entregar se encuentra en Aula y presenta todas las áreas con los datos a incluir.

2.2 Consideraciones

Se deben tener en cuenta las siguientes consideraciones a la hora de hacer la tarea

1. Los procesos deben ser creados recursivamente, siempre dividiendo el mapa en 4 cuadrantes. No se aceptaran tareas que creen cada proceso *a mano*
2. La palabra pueden estar orientadas tanto en vertical como horizontal pero siempre cumpliendo el largo del cuadrante.
3. La palabra siempre estará completa dentro de los cuadrantes, no quedaran mitades o tercios en otros procesos
4. De la mano con la primera consideración, la hebras deben crearse dentro de otras hebras, no en paralelo

3 Archivo de prueba

Se adjunta con el enunciado 2 archivos de prueba. Sin embargo, considere que al momento de que su tarea sea corregida, el o los archivos serán distintos (en contenido, no en formato).

4 Presentación Aleatoria

Para cada tarea, se seleccionarán grupos al azar para presentar su tarea frente a ayudantes y eventualmente profesor, recibiendo una ponderación del 80% y 20% entre tarea y presentación respectivamente. Si su grupo presentó en una tarea, no volverá a salir nuevamente. Se comunicará días antes que grupos presentarán.

5 README

Debe contener como mínimo:

- Nombre, RoI y Paralelo de los integrantes.
- Especificación de los nombres de los archivos. (Cual corresponde a cada sección de la tarea)
- Instrucciones generales de compilación y uso.

6 Consideraciones Generales

- Se deberá trabajar de a pares. Se deberá entregar en Github en el repositorio de su grupo a mas tardar el **día 5 de Noviembre de 2023 a las 23:59 hrs.** Se descontarán 5 puntos por cada hora o fracción de atraso. Las copias serán evaluadas con nota 0 en el promedio de las tareas.
- La tarea debe ser hecha en Java. Se asume que usted sabe programar en este lenguaje, ha tenido vivencias con el, o que aprende con rapidez.
- 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.
- Los archivos deberán ser subidos a la carpeta correspondiente del repositorio.
- Las preguntas deben ser hechas por Aula o por Discord. De esta forma los demás grupos pueden beneficiarse en base a la pregunta.
- Si no se entrega README o MAKE, o si su programa no funciona, la nota es 0 hasta la recorrección.
- Se descontarán 50 puntos por:
  - Mala implementación del Makefile.
  - No respetar el formato de entrega.