Proyecto Programación Concurrente y Paralela: conteo de palabras

El proyecto consiste en elaborar un programa en C++ reciba por la entrada estándar un texto y retorne la cantidad total de veces que aparece cada palabra en el documento. La lista de palabras debe estar ordenada alfabéticamente.

El enunciado del problema es simple, sin embargo, esta tarea tiene sus retos. Si el archivo es lo suficientemente grande tendrán que tomar medidas especiales para poder procesar el archivo en su totalidad y de la forma lo más rápida posible.

Requisitos de alto nivel

1. El programa DEBE poder leer un documento desde la entrada estándar.
    a. Solamente pueden leer el archivo una vez
2. Debe retornar una lista de palabras junto con la cantidad de veces que aparece cada palabra. La lista debe estar ordenada alfabéticamente. El formato es:
    a. Palabra: total
    b. Una palabra por línea
3. El diseño es relativamente libre, pero el programa debe hacer uso de concurrencia y paralelismo visto en clase.
4. Deben aprovechar las técnicas de paralelismo y concurrencia para lograr que su programa sea lo más rápido posible.
5. El programa NO debe fallar con archivos arbitrariamente grandes. Asuma que el archivo no cabe en memoria
    a. Los archivos son archivos de texto simples, separados por espacios en blanco y saltos de línea. Cada línea mide a lo sumo 1MB, sin embargo, el archivo podría medir varios GB.
6. No debe haber fugas de memoria
7. Se permite escribir a disco para reducir la presión de la memoria. Sin embargo, recuerde que esto hará que su programa sea más lento.
8. El programa deberá correr en un sistema Linux.

Otros aspectos

1. Pueden trabajar en grupos de hasta 3 personas máximo
2. Se espera que todos los miembros del equipo realicen un aporte equitativo. En caso de que no sea así, la nota será ponderada de acuerdo con el aporte individual.

3. DEBEN usar Git para gestionar su código. Se recomienda encarecidamente que utilicen branches por persona/feature, que creen pull requests regularmente, se revisen entre ustedes y luego mergeen el código en la rama principal.
4. DEBEN usar clang-format para formatear cada archivo del proyecto. Esto debe ser realizado en cada commit.
5. DEBEN agregar dentro del repo un Readme en formato markdown que indique al menos:
    a. Miembros del equipo
    b. Partes elaboradas por cada miembro
    c. Cómo compilar el proyecto
    d. Cómo correr el proyecto
    e. Diseño de su programa. Consideren usar diagramas mermaid dentro de su documento markdown, ya que Github puede renderizarlo correctamente.
    f. Problemas encontrados
    g. ¿Qué funciona y qué no funciona?

Fechas de evaluación

Lunes 6 de mayo: diseño del proyecto. Será discutido grupalmente en clase y será evaluado como una tarea.

13 y 20 de mayo: se presentarán avances del progreso del proyecto. Serán evaluados como una tarea.

26 de mayo a medianoche: entrega del proyecto. Se evaluará lo que esté en el repo.

27 de mayo: presentación final del proyecto. Será evaluado como una tarea.

Evaluación

Se evaluarán los siguientes aspectos:

1. Orden y calidad del código
    a. Uso apropiado de clases y programación orientada a objetos
    b. Uso de clang-format. En particular, no se revisarán proyectos que no estén debidamente formateados.
    c. Deben compilar usando los flags -Werror, -Wall y -Wextra .
2. Correctitud del programa
3. Eficiencia y escalabilidad del programa:
    a. El programa usa algoritmos y estructuras de datos eficientes
    b. El programa aprovecha apropiadamente todos los recursos
4. Se hace un consumo eficiente de la memoria

5. En caso necesario, el spilling (guardar a disco) se realiza correctamente
6. No hay fugas de memoria
7. El programa no falla
8. Documentación

Desglose de la evaluación

Calidad del código: 25%

Eficiencia y escalabilidad del programa: 25%

Spilling: 10%

Correctitud: 10%

No hay fugas de memoria: 10%

Documentación: 20%

Recomendaciones

1. No lo dejen para el final
2. Piensen bien el diseño de su proyecto y cómo van a dividir el trabajo entre todos los recursos disponibles. Consideren aspectos como:
    a. Consumo de memoria
    b. División del trabajo entre threads
    c. Comunicación entre threads
3. Solamente pueden leer el archivo una vez