CI-0117 Semana 8 Avance de la tarea programada - Programación Paralela y Concurrente
Jennifer Villalobos Fernández | B67751

---

1. Explicar la estrategia de desarrollo propio para la distribución de la carga de la lectura de un archivo entre varios trabajadores.
Pensaba en hacer una estrategia similar a la número 1, sin embargo, en lugar de dividir el total de lineas del archivo entre la cantidad de trabajadores, pensaba darle un "rango" equitativo de líneas a cada trabajador, según el total de líneas del archivo. Estos, leerían una línea a la vez, pero la acción se repetiría consecutivamente según ese rango. Por ejemplo, si el rango es 5, cada trabajador leería una línea a la vez, pero realizaría esta tarea 5 veces seguidas, dentro de su Lock.

- Indicar si la estrategia considera un balance en la distribución de la carga: sí, cada trabajador tendría el mismo rango, o muy similar.
- Indicar el orden de lectura de las líneas del archivo de texto: cada trabajador leerá su bloque de líneas, luego el siguiente tomará la posición resultante y seguirá con su bloque de líneas, así sucesivamente.

2. Construir una clase "FileReader" para poder distribuir las líneas de un archivo de texto entre varios trabajadores siguiendo varias estrategias.

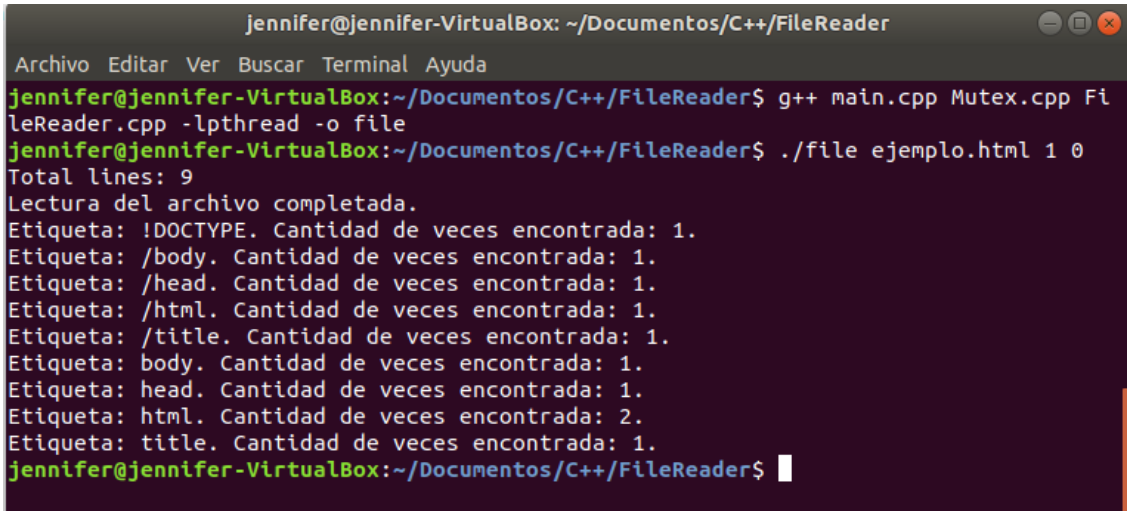
- Definir las constantes que podrá utilizar en esta clase. Por ejemplo, para las estrategias: para las estretegias utilicé varias variables: el archivo por abrir (este está en duda, ya que creo que sólo voy a utilizar el nombre del archivo por abrir), el mapa de las etiquetas HTML posibles, el rango que cada estrategia debe cubrir (en realidad este parámetro solo se aplica para la primera), la identificación del hilo actual (posiblemente esta será removida), la estrategia a utilizar, el nombre del archivo (dicho anteriormente) y, además, la posición actual en el archivo. Esta última se agregó pero aún existen ciertos fallos al ejecutar el programa, por eso no se subió al repositorio.
- Determinar los elementos necesarios para almacenar internamente en la clase: la clase cuenta con varios atributos privados: la cantidad de trabajadores, la estrategia a utilizar, el total de líneas, el archivo, el rango (más otro llamado RangeMaster, en caso de que el total de líneas no sea divisible entre la cantidad de trabajadores), la posición actual, el nombre del archivo y el mapa de las posibles etiquetas HTML.
- Determinar las tareas que debe realizar el constructor de esta clase: por el momento, ninguna. Las tareas de inicialización se ejecutan en el método Read().
  - Indicar los parámetros necesarios para construir una instancia de esta clase: ninguno, cuando se crea un hilo Lector, este llama al proceso Read() e inicializa todo lo necesario, incluso los datos que ocuparán sus trabajadores.
  - Pensar cómo determinar el total de líneas del archivo: el total de líneas se determina mediante el método countLines(). Este recibe el archivo ya abierto, por lo que no infringe la restricción de que cada instancia solo debe abrir el archivo una vez.
  - Escribir la estrategia para lograr que todas las líneas del archivo sean asignadas a algún trabajador y para no dejar líneas sin procesar, especialmente en el caso que la cantidad

total de líneas del archivo no sea exactamente divisible por el número de trabajadores: en el método Read() se realiza la división del total de líneas entre la cantidad de trabajadores. Si el residuo de esta es 0, el rango será el mismo para todos. En caso contrario, la variable rangeMaster, que será enviada al trabajador 0, tendrá un rango del bloque que le corresponde a cada hilo, más las líneas que "sobraron" de la división.

- Definir las tareas que debe realizar el destructor: por el momento, cada instancia destruye su propio archivo luego de utilizarlo, esa tarea pienso dársela al destructor.
- Analizar si su clase va a tener los métodos "hasNext()" y "getNext()" para poder iterar sobre los elementos que corresponden a cada uno de los trabajadores o, en su defecto, indicar la manera en que cada trabajador va a obtener las líneas del archivo que le corresponde. Explicar el método para manejar el final del archivo para cada uno de los trabajadores; también para determinar cuál es su siguiente línea de lectura: por el momento, creo que no serán utilizados. La estretegia que pienso utilizar consiste en crear un vector de posiciones (variables tipo fpost), el cual en cada index (0, 1, 2, ...) tendrá la posición actual del archivo correspondiente al id del maestroLector. Es decir, en el index 0, estará la posición actualizada (luego de cada estrategia realizada por cada trabajador) del archivo del maestroLector 0. Antes de que un trabajador realice una estrategia, tomará esta posición, y luego de realizarla, actualizará el index del vector con la posición en la que quedó.
- Recuerde que, como restricción del proyecto, su clase solo podrá mantener en memoria principal una de las líneas del archivo por cada trabajador. También como restricción del proyecto, cada instancia de esta clase solo podrá tener su archivo abierto una sola vez: ambas restricciones se toman en cuenta y no se infringen.

3. Construya un programa de prueba para demostrar la funcionalidad de su clase lectora.

- Haga pruebas utilizando todas las estrategias posibles: la prueba se realizó solo con la estrategia por default, es decir, solo un trabajador realiza la lectura de todo el archivo.
- En la estrategia dinámica, construya la instancia con tres tabajadores pero procese el archivo utilizando solo dos, debe comprobar que estos dos pueden procesar el archivo completamente: el programa posee los métodos de las 4 estrategias (tentativos), pero no el de la inventada. Las pruebas con varios hilos no pudieron realizarse debido a los inconvenientes con la implementación de las posiciones en los métodos de las demás estrategias.



(a) Ejecución del programa con ejemplo.html, un trabajador y la estretgia 0 (por default).