Laboratório de Programação Concorrente

Lab4 - 25.1

File Indexing Pipeline

Objetivo

O objetivo deste laboratório é aplicar conceitos de concorrência e sincronização usando threads, semáforos e estruturas de dados compartilhadas, por meio da implementação de um sistema concorrente de indexação de arquivos de texto em três etapas: leitura, tokenização e indexação.

Em sistemas de processamento de dados em larga escala, é comum estruturar o fluxo de dados em **pipelines**: etapas sequenciais de transformação e processamento, onde a saída de uma etapa é a entrada da próxima. Neste laboratório, você irá desenvolver um sistema de indexação de palavras de arquivos de texto, estruturado como um pipeline com três etapas:

1) Leitura de arquivos do disco.
2) Tokenização do conteúdo lido em palavras.
3) Indexação das palavras em uma estrutura global de índice invertido.

Cada etapa deverá ser executada por uma ou mais threads, e a comunicação entre elas será feita por meio de buffers que devem ser thread-safe. Assim, o Pipeline deve ser estruturado da seguinte forma:

- **Leitor:** lê arquivos e insere o conteúdo em um buffer intermediário.
- **Tokenizador:** consome conteúdo lido, divide em palavras e insere em novo buffer.
- **Indexador:** consome as palavras e atualiza uma estrutura de índice invertido (mapa palavra → arquivos onde aparece e contagem).

Neste laboratório, nós disponibilizamos o código serial de um sistema Produtor-Consumidor que considera um **buffer ilimitado** e **nenhum controle de acesso concorrente**. Vocês devem implementar a concorrência neste sistema, considerando **a estrutura do pipeline descrito acima**.

Desenvolva uma solução segura para evitar condições de corrida e outros problemas de sincronização, além de garantir os seguintes requisitos:

- Deve haver **3 tipos de threads**, cada um responsável por uma etapa do pipeline.
- **Os buffers devem ter capacidade máxima**, use 50 itens como exemplo.
- As threads devem ser sincronizadas corretamente usando a anotação **synchronized e/ou semáforos**.
- A aplicação deve finalizar corretamente após o término do processamento. Utilize null como sinalizador de término no pipeline (como EOF).
- Não use bibliotecas de alto nível como ExecutorService, BlockingQueue, etc. O foco é o uso explícito de threads e semáforos.
- O programa deve funcionar corretamente com qualquer número de arquivos e qualquer conteúdo textual.

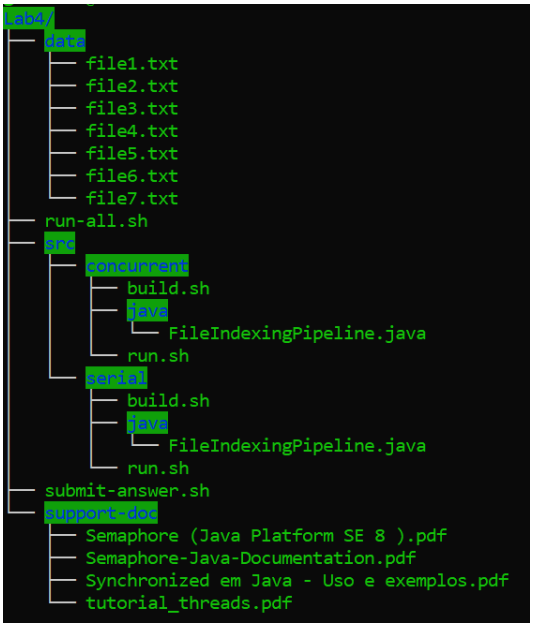
Prazo

22/07/2025 às 16h

Visão geral do código base

https://github.com/giovannifs/fpc/tree/master/2025.1/Lab4

O código está organizado de acordo com a hierarquia abaixo:



- ***src/serial/:*** Diretório que disponibiliza a implementação serial do pipeline de indexação de arquivos.

- ***src/concurrent/:*** Diretório onde você implementará a versão concorrente do pipeline de indexação de arquivos. **Note que, inicialmente, este diretório contém uma cópia da implementação serial, então você deve alterá-la para implementar a concorrência!**
- **scripts auxiliares nos diretórios serial e concurrent:**
  - *build.sh:* script para compilar o código do diretório
  - *run.sh:* script para executar as implementações do diretório.

    run.sh filepath1.txt filepath2.txt … filepathN.txt
- ***run-all.sh:*** Script para compilar e executar as implementações serial e concorrente, em seus respectivos diretórios. Este script considera que os arquivos do diretório data serão passados como argumentos das execuções.
- ***submit-answer.sh:*** Script que será utilizado para a submissão da resposta do laboratório.
- ***support-doc:*** Disponibiliza documentação e tutorial sobre o uso de threads, semáforos e sincronizações em Java.

Execução do código

1. Clone o repositório do código base

   git clone [link do repositório]

2. Navegue até o diretório do Lab4:

   cd fpc/2025.1/Lab4/

3. Execute o script run_all.sh para executar todos os cenários considerando os arquivos dataset como argumentos das execuções

   bash run_all.sh

   bash run_all.sh > result.out (redireciona a saída para o arquivo result.out, onde você pode analisar com mais detalhes)

Entrega

Você deve criar e manter um repositório privado no GitHub com a sua solução. No entanto, a entrega do laboratório deverá ser realizada por meio de submissão online utilizando o script submit-answer.sh, disponibilizado na estrutura de arquivos do próprio laboratório. Uma vez que você tenha concluído sua resposta, seguem as instruções:

1) Crie um arquivo lab4_matr1_matr2.tar.gz somente com o “src” do repositório que você trabalhou. Para isso, supondo que o diretório raiz de seu repositório privado chame-se lab4_pc, você deve executar:

   ```
   tar -cvzf lab4_matr1.tar.gz lab4_pc/src
   ```

2) Submeta o arquivo lab4_matr1_matr2.tar.gz usando o script submit-answer.sh, disponibilizado no mesmo repositório do laboratório:

   ```
   bash submit-answer.sh lab4 path/to/lab4_matr1_matr2.tar.gz
   ```

Lembre-se que você deve manter o seu repositório privado no GitHub para fins de comprovação em caso de problema no empacotamento ou transmissão online. Alterações no código realizadas após o prazo de entrega não serão analisadas.