Laboratório de Programação Concorrente

Lab3 - 25.1

Producer Conditional Consumer Problem

Objetivo

Muitos problemas em Ciência da Computação são resolvidos com uma estratégia que envolve dois grupos de processos: produtores e consumidores. O consumidor, para realizar o seu trabalho, precisa de itens produzidos por processos produtores. Tipicamente, um buffer compartilhado mantém os itens produzidos à espera de consumidores.

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 **Produtores Genéricos e Consumidores Condicionais**. Este problema é uma variação do padrão Produtor-Consumidor clássico, em que os consumidores não processam qualquer item do buffer, mas apenas aqueles que satisfazem uma determinada condição.

A sua solução deve lidar com a execução simultânea de produtores e consumidores e o controle de acesso concorrente utilizando a anotação **synchronized e/ou semáforos**. Desenvolva uma solução segura para evitar condições de corrida e outros problemas de sincronização.

Sua implementação deverá garantir os seguintes comportamentos:

1) **Buffer limitado e seguro:**
    a) O buffer deve ter **capacidade máxima de 100 itens**.
2) **Produtores:**
    a) Inserem números inteiros aleatórios no buffer.
    b) Devem aguardar se o buffer estiver cheio, evitando sobrescrever dados ainda não consumidos.
    c) Podem operar de forma concorrente entre si.
3) **Consumidores:**
    a) Existem dois tipos distintos:
        i) Um que consome apenas números pares.
        ii) Outro que consome apenas números ímpares.
    b) Devem remover itens do buffer de forma concorrente, sem acessar posições inválidas ou consumir o mesmo item.
    c) Se um consumidor remover um item que não atende à sua condição, ele deve reinserir o item no final do buffer.
    d) Caso o buffer esteja vazio, os consumidores devem aguardar até que novos itens sejam produzidos.

4) **Concorrência robusta:**
    a) A solução deve funcionar corretamente com qualquer número de produtores e consumidores executando simultaneamente.
    b) Você deve testar o sistema com diferentes quantidades de produtores e consumidores para avaliar o comportamento sob carga variada.

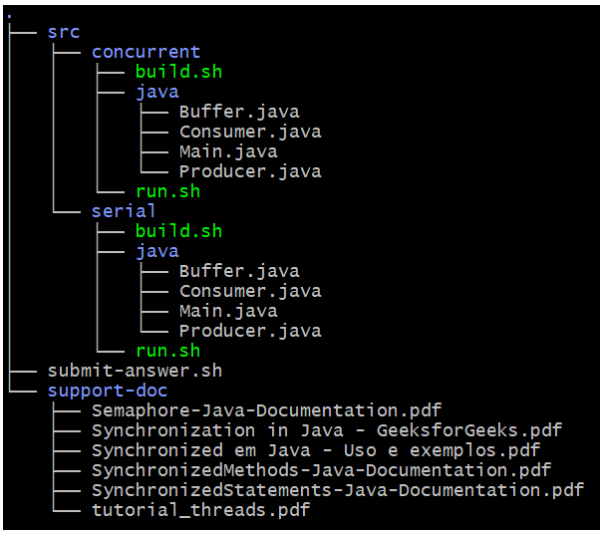
Prazo

08/07/2025 às 16h

Visão geral do código base

https://github.com/giovannifs/fpc/tree/master/2025.1/Lab3

O código está organizado de acordo com a hierarquia abaixo:



- ***src/serial/:*** Diretório que disponibiliza a implementação serial do sistema produtor-consumidor.
- ***src/concurrent/:*** Diretório onde você implementará a versão concorrente do sistema produtor-consumidor. **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. Este exige os seguintes argumentos:

        - *<numero_produtores>*
        - *<numero_items_por_produtor>*
        - *<tempo_producao> (em milisegundos)*
        - *<numero_consumidores>*
        - *<tempo_consumo> (em milisegundos)*

- ***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. Execução da versão serial do código
   a. Navegue até o diretório da implementação serial:

      cd fpc/2025.1/Lab3/src/serial

   b. Compile o código

      bash build.sh

   c. Execute a versão serial especificando os parâmetros:

      bash /run.sh 1 100 150 1 150

3. Execução da versão concorrente do código
   a. Navegue até o diretório da implementação serial:

      cd fpc/2025.1/Lab3/src/concurrent

   b. Compile o código

      bash build.sh

   c. Execute a versão serial especificando os parâmetros:

      bash /run.sh 1 100 150 1 150

Experimente diferentes configurações para ambiente de execução do sistema, por exemplo, com múltiplos produtores e consumidores e

diferentes tempo de produção e consumo dos itens. Observe se o comportamento do sistema faz sentido em cada um dos cenários experimentados.

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 lab3_matr1.tar.gz somente com o “src” do repositório que você trabalhou. Para isso, supondo que o diretório raiz de seu repositório privado chame-se lab3_pc, você deve executar:

   ```
   tar -cvzf lab3_matr1.tar.gz lab3_pc/src
   ```

2) Submeta o arquivo lab3_matr1.tar.gz usando o script submit-answer.sh, disponibilizado no mesmo repositório do laboratório:

   ```
   bash submit-answer.sh lab3 path/to/lab3_matr1.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.