Segundo trabalho prático de implementação
Leitores e escritores sem inanição

Computação Concorrente (MAB-117) — 2019/2
Prof. Silvana Rossetto

[1]DCC/IM/UFRJ
12 de novembro de 2019

Descrição do problema

O problema dos **leitores/escritores** é um problema clássico de concorrência. As condições lógicas do problema são:

- mais de um leitor pode ler ao mesmo tempo;
- apenas um escritor pode escrever de cada vez;
- não é permitido ler e escrever ao mesmo tempo.

Programa principal

Neste trabalho, você deve projetar e implementar uma versão simplificada desse tipo de problema, **garantindo ausência de inanição das threads**.

As threads leitoras e escritoras deverão receber um identificar (inteiro) único. **O espaço de leitura e escrita compartilhada será uma única variável do tipo inteiro.** As threads leitoras deverão ler o valor dessa variável e escrevê-la em um arquivo de saída (um arquivo exclusivo para cada thread leitora). As threads escritoras deverão escrever o valor do seu identificador único na variável compartilhada.

Programa auxiliar e arquivo de log

Uma parte importante deste trabalho será a geração de um **log de saída** para registrar todas as operações realizadas no sistema e a implementação de um **programa auxiliar** para verificar se a execução ocorreu com sucesso.

Para isso, as threads deverão registrar cada ação realizada, juntamente com o valor corrente da variável compartilhada.

A thread principal do programa deverá inicializar as estruturas de dados e variáveis necessárias e disparar as demais threads. Quando todas as threads terminarem suas execuções e o log for completamente escrito no arquivo de saída, o programa deverá ser encerrado.

Os códigos executados pelas threads (programa principal) poderá ser implementado nas linguagens **C** (usando a biblioteca `Pthreads`) ou **Java**, **usando de forma eficiente os recursos de concorrência e os mecanismos de sincronização vistos na disciplina**. O programa auxiliar poderá ser implementado em qualquer linguagem.

Entrada

Para a execução do programa, os seguintes dados deverão ser fornecidos como entrada: *quantidade de threads leitoras, quantidade de threads escritoras, número de leituras (das*

threads leitoras), número de escritas (das threads escritoras), e o nome do arquivo de log. Os nomes dos arquivos gerados pelas threads leitoras deverão receber como nome o código da thread seguido da extensão `.txt` (por ex., $< 5.txt >$ será o nome do arquivo gerado pelas leituras realizadas pela thread 5).

Saída

O programa auxiliar que irá avaliar o log de saída e validar a execução do programa principal deverá reproduzir, de forma sequencial, a execução de cada função registrada no log e avaliar se as operações foram executadas corretamente **garantindo as restrições do problema e a ausência de inanição das threads**.

A linguagem usada para implementar o programa auxiliar é de livre escolha. (Pode ser C, Java, ou linguagens de scripting como Lua, Python, etc..)

Escolhida a linguagem de implementação do programa auxiliar, uma alternativa para gerar o arquivo de log é escrevê-lo usando a sintaxe da própria linguagem e depois apenas executá-lo. Nesse formato, o arquivo de log pode ser executado diretamente, dado que as funções chamadas por ele estão implementadas em outro arquivo.

O programa auxiliar deve gerar como saída uma mensagem que valida a execução do programa principal, **destacando os pontos onde a estratégia para garantir ausência de inanição foi acionada**; ou uma mensagem de erro apropriada indicando em qual operação ocorreu erro e sua descrição.

Etapas do trabalho

A execução do trabalho deverá ser organizada nas seguintes etapas:

1. Compreender o problema principal, pensar e esboçar a solução do problema;
2. Projetar as estruturas de dados e as funçoes que deverão ser implementadas;
3. Projetar e implementar o programa auxiliar que deverá avaliar o log de execução do programa principal;
4. Construir um conjunto de casos de teste para avaliação da aplicação;
5. Implementar a solução projetada, avaliar a corretude dos programas, refinar a implementação e refazer os testes;
6. Redigir os relatórios e documentar os códigos.

Artefatos que deverão ser entregues

1. **Relatório**: documentação do projeto da solução (esboço das estruturas de dados e lógica principal dos algoritmos implementados), testes realizados e resultados obtidos.
   O relatório deverá conter informações suficientes para o professor compreender a solução proposta sem precisar recorrer ao código fonte do programa.
2. Arquivos com os **códigos fonte** e README (roteiro para compilação e execução dos programas).

Data de entrega e critérios de avaliação

O trabalho deverá ser feito em **dupla** e deverá ser entregue até o dia **30 de novembro de 2019**, via formulário do Drive, disponível no site da disciplina. O código implementado deverá ser disponibilizado em algum repositório de código (ex., GitHub ou GitLab).

Os seguintes itens serão avaliados com o respectivo peso:

- Compilação e execução correta dos dois programas: 5 pontos
- Relatório (incluindo projeto da solução e testes realizados): 2 pontos
- Modularidade, organização e documentação do código fonte, uso eficiente dos mecanismos de sincronização: 3 pontos

Não é permitido copiar a solução do trabalho de outros colegas ou de terceiros.

Os alunos integrantes da equipe poderão ser chamados pelo professor para apresentar/explicar o trabalho.