Módulo 2 - Laboratório 5
Problemas clássicos de concorrência usando locks e variáveis de condição (leitores/escritores)

Computação Concorrente (MAB-117)
Prof. Silvana Rossetto

[1]DCC/IM/UFRJ

Introdução

O objetivo deste Laboratório é continuar implementando problemas clássicos de concorrência usando locks e variáveis de condição.

Atividade 1[1]

Objetivo: Projetar e implementar uma aplicação com o padrão **leitores/escritores**, fazendo variações no padrão básico.

Descrição: Implemente uma aplicação com threads que acessam de forma concorrente um **vetor de valores inteiros**. Nessa aplicação existem dois tipos de threads:

- **Leitora**: percorre o vetor de inteiros e imprime na tela seu conteúdo e a média dos valores encontrados;
- **Escritora**: modifica o conteúdo do vetor escrevendo o valor do seu identificador de thread na aplicação na primeira e última posição do vetor, e o dobro desse valor nas demais posições.

Observe que não há condição de corrida quando somente threads leitoras acessam o vetor, pois as leituras não o modificam e podem ser simultâneas. Os acessos simultâneos permitem um maior desempenho em leitura. No entanto, uma thread escritora deve ter acesso exclusivo ao vetor para fazer suas modificações, sem acessos concorrentes de outras threads, sejam leitoras ou escritoras, para evitar condições de corrida.

Existem três tipos de soluções para esse problema de sincronização:

1. **Sem prioridades** (definição básica do problema): sempre que um leitor quiser ler e não houver escritor escrevendo (pode haver escritor esperando), ele tem acesso à estrutura de dados. Nesta solução, um escritor pode ter de esperar por muito tempo (inanição), caso novos leitores cheguem com frequência.
2. **Prioridade para escritores**: quando um escritor desejar escrever, nenhum leitor pode iniciar uma leitura enquanto o escritor não for atendido. Nesta solução, um leitor pode ter de esperar por muito tempo (inanição), caso novos escritores cheguem com frequência.
3. **Prioridades iguais**: não há risco de inanição, pois leitores e escritores têm as mesmas chances de acesso à estrutura de dados; pode haver uma queda de desempenho em relação às soluções anteriores.

[1]Baseado no exercício proposto pelo prof. Maziero em: http://wiki.inf.ufpr.br/maziero/doku.php?id=so:leitores_escritores

Tarefa: Implemente as três versões para essa aplicação, uma para cada uma das soluções de sincronização descritas acima.

Roteiro:

1. Considere um número N de threads leitoras ($N >= 2$) e um número M de threads escritoras ($M >= 2$).
2. Acrescente no seu código a impressão de **informações que permitam acompanhar a execução da aplicação para verificar se as condições lógicas do problema são satisfeitas**.
3. Execute a aplicação **várias vezes** e avalie os resultados obtidos.
4. Altere o número de threads leitoras e escritoras e reexecute a aplicação.

Disponibilize o código implementado na (as três versões) em um ambiente de acesso remoto (GitHub ou GitLab). **Use o formulário de entrega desse laboratório para enviar o link do repositório do código implementado e responder às questões propostas.**