# Laboratório 5
# Programação Concorrente

Manoel Marcelo da Silva - 121088349
Instituto de Computação – Universidade Federal do Rio de Janeiro
manoelms@ic.ufrj.br

14 de maio de 2024

**Questão 1.** O TAD lista implementado nesses arquivos poderia ser compartilhado por threads de um programa concorrente? Com qual finalidade e de que forma?

O tipo abstrato de dados implementado nestes arquivos pode ser compartilhado sim por threads em um programa concorrente, desde que seja garantido que a implementação de seu uso seja thread safe, ou seja, que garanta sua execução sem problemas concorrentes como corrida de dados. Seu uso é imprescindível em alguns algoritmos que se necessita de uma estrutura de dados que seja bastante dinâmica, com alocação dinâmica de memória, que permite adicionar e remover elementos facilmente, sem a necessidade de alocar espaço fixo no início.

Em um programa concorrente em que várias threads acessem o mesmo TAD, no escopo global/compartilhado, caso seja alterado - o que é bem provável que ocorra nesse tipo de estrutura - é necessário garantir que apenas uma thread o acesse, usando de mecanismos como exclusão mútua ou de sincronização.

**Questão 2.** O que poderia acontecer se o programa não implementasse exclusão mútua no acesso as operações da lista encadeada?

Neste caso, poderia acontecer de uma thread alterar o valor/chave de um nó enquanto outra thread a lê, assim causando inconsistências no resultado; Além disso é possível que ocorra o acesso a um nó deletado por uma thread por outra, causando segmentation fault; Inconsistências na lista também podem acontecer, ao inserir um novo nó na lista enquanto também é inserido outro nó por outra thread. Assim, ao não garantir uma exclusão mútua, causa-se esses problemas ao executar essas funções concorrentemente.

**Questão 3.** O que acontece com o tempo de execução do programa quando aumentamos o número de threads? Por que isso ocorre?

Após um certo número de threads o tempo de execução do programa aumenta relativamente, isso se deve à necessidade de troca de contexto/overhead por conta da obrigação da sequencialização do trecho do código, como se fosse uma fila. Ao aumentar bastante o número dessas threads, essa fila aumenta cada vez mais, consequentemente aumentando o tempo de espera e seu tempo de execução.

**Questão 4.** Em quais cenários o uso do rwlock pode ser mais vantajoso que o uso do lock de exclusão mútua?

O uso de 'rwlock' (mutex de leitura-escrita) pode ser mais vantajoso que o uso de 'lock' de exclusão mútua em cenários onde há um grande número de operações de leitura e um número menor de operações de escrita. O 'rwlock' permite que várias threads leiam simultaneamente, desde que não haja nenhuma thread escrevendo, o que pode melhorar significativamente o desempenho em comparação com um 'lock' de exclusão mútua tradicional, que em contraste, restringem tanto a leitura quanto a escrita, o que pode causar contenção e penalizar o desempenho.

**Questão 5.** Em que tipo de aplicação concorrente o padrão de sincronização coletiva com barreira precisa ser usado?

Existem vários tipos de aplicações e algoritmos em que é necessário garantir que todas as etapas de determinada iteração estejam concluídas antes que passe para a próxima fase/etapa do algoritmo, como por exemplo o Simplex ou Gauss-Jacobi. Neste contexto, a barreira é um tipo de sincronização coletiva que suspende a execução das threads de um aplicação em um dado ponto do código e somente permite que as threads prossigam quando todas elas tiverem chegado naquele ponto.