ATIVIDADE 01 - PROGRAMAÇÃO CONCORRENTE E DISTRIBUÍDA (2025.2)

Observações (Leia antes de ver os exercícios):

- **Data/Hora da Entrega**: 26/09/2025,sexta-feira, até as 23:59h (FIRME).
- **Entregáveis**: Gerar um único arquivo (e.g., zip) com a implementação. Não pode ser link.
- **Forma de Entrega**: Exclusivamente pelo SIGAA. Observe que o SIGAA aceita o envio de apenas um único arquivo.
- Apenas um dos membros da equipe precisa realizar a entrega.
- Todas as implementações precisam ser na linguagem Go.
- A lista com as equipes se encontra no final deste documento.
- Algumas equipes relataram problemas nos grupos preenchidos no formulário, e estes problemas foram ajustados. Qualquer outra situação só poderá ser analisada por mim a partir do sábado (27/09/2025), quando voltarei a ter pleno acesso à Internet.
- IMPORTANTE: Por conta de uma longa viagem, estarei offline (sem acesso à Internet) na quarta-feira e quinta-feira. Portanto, se a equipe estiver com alguma dúvida sobre como realizar a implementação, faça uma escolha a partir do que você está entendendo do exercício. O importante é enviar a implementação no prazo estabelecido e da forma como você entendeu.

Exercícios

[Equipe 01] Implemente uma aplicação de reservas com múltiplos clientes (*threads*) tentando reservar recursos limitados (e.g., assentos, quartos).

[Equipe 02] Implemente uma variação do problema Produtor-Consumidor, em que múltiplos threads produtores e consumidores interagem por meio de vários buffers circulares independentes, em vez de um único buffer compartilhado. Cada produtor deve escolher aleatoriamente um dos buffers disponíveis para inserir um item, enquanto cada consumidor escolhe aleatoriamente um buffer para consumir. Os buffers têm capacidade limitada, e deve haver controle adequado de concorrência.

[Equipe 03] Implemente um sistema de thread pool, no qual um conjunto fixo de threads (por exemplo, 4, 8 ou 16) é criado antecipadamente e permanece em execução aguardando tarefas a serem processadas. As tarefas são produzidas continuamente por um ou mais threads produtores e armazenadas em uma fila de tarefas compartilhada (queue). Cada thread do pool retira tarefas da fila de forma sincronizada e as executa. A implementação deve evitar condições de corrida e garantir que cada tarefa seja processada exatamente uma vez.

[Equipe 04] Implemente uma aplicação de download de um único arquivo grande, onde vários threads baixam partes diferentes do arquivo simultaneamente. Cada thread deve ser responsável por uma faixa específica de bytes, e o conteúdo baixado deve ser armazenado corretamente na posição correspondente do arquivo final. Implemente controle de largura

de banda total, dividindo-a entre os threads, e sincronização para evitar conflitos ao escrever no arquivo compartilhado.

[Equipe 05] Implemente um sistema de análise de logs onde múltiplos threads leem e classificam registros de um log por categoria (erro, info, warning). Os resultados são agrupados em estruturas separadas.

[Equipe 06] Implemente uma rede de sensores de temperatura, onde cada sensor (thread) envia dados periodicamente a um servidor central (thread). O servidor agrega dados (e.g., calcula a média das temperaturas do sensor) e grava-os em um arquivo compartilhado.

[Equipe 07] Este problema envolve dois threads que trocam mensagens repetidamente — um envia um "ping" e o outro responde com um "pong" — simulando comunicação síncrona e coordenação entre entidades paralelas. Cada thread aguarda o outro antes de prosseguir, criando uma alternância controlada que garante que as mensagens sejam passadas na ordem correta.

[Equipe 08] Implemente o algoritmo de merge sort, dividindo recursivamente o array de entrada em subarrays que são ordenados concorrentemente por threads.

[Equipe 09] Implemente a multiplicação de duas matrizes grandes utilizando várias threads em paralelo. Divida o trabalho entre as threads atribuindo a cada uma o cálculo de linhas ou blocos da matriz resultante.

[Equipe 10] Dada uma lista grande de senhas, verifique (em paralelo) se cada uma atende a regras como:comprimento, presença de letras maiúsculas/minúsculas, números, símbolos e ausência de padrões conhecidos.

[Equipe 11] Implemente uma busca paralela em uma estrutura como uma árvore ou um grafo, em que múltiplos threads percorrem diferentes ramos simultaneamente à procura de um nó-alvo ou conjunto de critérios.

[Equipe 12] Implemente uma aplicação de criptografia de arquivos grandes, onde o conteúdo é dividido em blocos, e cada bloco é processado por um thread. A criptografia pode usar um algoritmo simples (e.g., AES) apenas para simulação, com foco na divisão de carga e sincronização de escrita dos blocos no arquivo final.

[Equipe 13] Implemente um sistema em que múltiplas threads escrevem em um mesmo arquivo (ou buffer simulado), cada uma com seus próprios dados. A escrita deve ser feita de forma ordenada, preservando a sequência ou evitando corrupção.

[Equipe 14] Implemente uma aplicação que simula sensores (threads) gerando dados em tempo real (e.g., temperatura, umidade, pressão). Um conjunto de threads analisa continuamente os dados e identifica padrões ou eventos, e.g., média excedendo um limite.

[Equipe 15] Implemente um indexador de arquivos onde múltiplos threads percorrem diretórios e arquivos de texto para construir um índice de palavras. Cada thread processa um conjunto de arquivos e atualiza uma estrutura de índice compartilhada.

[Equipe 16] Implemente uma simulação de sistema bancário com diversas contas e vários threads simulando transações concorrentes entre contas (transferências, depósitos e saques). Cada thread representa um cliente realizando operações aleatórias de forma contínua. A consistência dos saldos deve ser mantida garantindo que o saldo total do sistema permaneça estável.

[Equipe 17] Implemente um pipeline de validação de dados em que múltiplas threads verificam campos diferentes (CPF, CNPJ, endereço, idade, etc.) de forma concorrente. Os resultados são agregados ao final. Meça a latência do processo e taxa de erros detectados.

[Equipe 18] Divida uma imagem grande em blocos e processe-os (e.g., aplicar filtro, converter escala de cinza) com *threads* concorrentes. Ao final, recombine os blocos para formar a imagem final.

[Equipe 19] Implemente um sistema que divide um vídeo em blocos (frames ou segundos) e aplica filtros em paralelo (e.g., brilho, contraste, rotação). Depois, as partes são recombinadas.

Equipes:

Equipe	Membros
01	LUCAS SANTIAGO MONTERAZO, Mikael Cavalcanti da Silva
02	JOAO PEDRO CAMPOS DE MEDEIROS, SARAH LEITAO MELO
03	JOAO VICTOR RIBEIRO COSTA DE OMENA, PEDRO VITOR MONTE, GUILHERME MARANHÃO, Pierre Chevrollier Oria
04	ITALO LEANDRO DE LIMA SILVA, MARIA EDUARDA FARIAS, JOÃO PEDRO DEO
05	FREDERICK ALMEIDA LOPES, FELIPE NEIVA
06	BRENO GABRIEL DE MELO LIMA, Lucas Henrique do Nascimento Silva
07	ARIEL RODRIGUES SOUSA DOS SANTOS, DANILO JOSE COUTINHO DE OLIVEIRA
08	RAFAEL DO NASCIMENTO MOURA
09	CAIO FERREIRA GOMES DA SILVA
10	ANA CASSIMIRO ALVES
11	JOAO PEDRO CAVALCANTI FERNANDES, RAFAEL RIOS CABRAL VICTAL, RENATO MOREIRA SERRANO DE ANDRADE, RODRIGO ROSSITER LADVOCAT CINTRA
12	GABRIEL BRAZ CAVALCANTE SILVA, HENRIQUE LINS PEREIRA AFFONSO DE AMORIM
13	MATHEUS BORGES FIGUEIROA
14	ERNESTO GONCALVES DE LIMA NETO, FERNANDA DE ARAUJO LIMA PASCOAL

15	GIOVANNY LIRA DE ARAUJO CUNHA, GABRIEL FERREIRA DA SILVA, GABRIEL FERNANDES NOBREGA
16	JULIANA SILVA
17	THOMAS PATRICK GOMES DE ALCANTARA, IGOR EDUARDO MASCARENHAS
18	AZEMAR DA ROSA TEIXEIRA NETO
19	JOAO LUCAS VIEIRA DOS SANTOS