Exercícios com Threads

INF01151 - Sistemas Operacionais II - Prof. Eder John Scheid

8 de setembro de 2025

Este é o primeiro de quatro exercícios práticos que compõem parte da avaliação prática (MPP). Os quatro exercícios, no total, valem 1,0 ponto da MPP; portanto, cada exercício corresponde a 0,25 ponto (25% do total).

Objetivo

Praticar criação de threads, passagem de argumentos e sincronização por `join`, **sem semáforos** e **sem busy-wait**.

Exercícios

Exercício 1. Olá, Threads *Linguagem: C (pthreads)*

Crie T threads. Cada thread imprime: `Hello, sou a thread <id>` e termina.

- **Requisitos:** (i) `id` passado na criação; (ii) a `main` chama `pthread_join()` em todas; (iii) T pela linha de comando; (iv) **não** usar semáforos nem espera ativa.
- **Dica de compilação/execução:**
  ```
  gcc -pthread hello.c -o hello
  ./hello <T>
  ```

Exercício 2. Ordem de término (sleep aleatório) *Linguagem: Java*

Crie T threads; cada uma dorme um tempo aleatório $[0, 300]$ ms e, ao acordar, imprime `FIM <id>` e termina.

- **Requisitos:** (i) a `main` cria todas e depois faz `join()` em todas; (ii) a **ordem das linhas** `FIM` no console representa a ordem real de término; (iii) T pela linha de comando; (iv) **não** usar semáforos nem busy-wait (usar apenas `Thread.sleep` e `join`).
- **Dica de execução:**
  ```
  javac Main.java
  java Main <T>
  ```

Exercício 3. Soma em variável global *Linguagem: C (pthreads)*

Objetivo didático: evidenciar, na prática, a *condição de corrida* ao atualizar uma variável global sem proteção.

Crie um vetor **global** `int A[N]` com N **constante no código** (ex.: `#define N 1000000`) e uma variável **global** `long long SUM = 0`. Divida A em T fatias contíguas. Cada thread percorre **apenas sua fatia** e **acumula diretamente em** `SUM` (isto é, sem mutex, sem semáforos, sem operações atômicas). A `main` chama `pthread_join` em todas, calcula a soma sequencial e compara os resultados.

- **Requisitos:** (i) `A` e `SUM` **globais**; (ii) N é **constante** definido no código (não via linha de comando); (iii) T pode vir da linha de comando (ou ser constante, a critério); (iv) **não usar** qualquer mecanismo de sincronização (`pthread_mutex_t`, semáforos, atômicos, etc.); (v) imprimir `SUM_paralelo`, `SUM_sequencial` e `OK`/`ERRO`.
- **Experimento sugerido:** executar várias vezes (e com `-O2`) para observar resultados incorretos ocasionais devido a *condição de corrida*; variar T para intensificar a disputa.
- **Dica de compilação/execução:**
  `gcc -pthread soma_global_corrida.c -o soma`
  `./soma <T>`.

Entrega

- Envie no Moodle um arquivo comprimido (*e.g.,* `.zip` ou `.tar.gz`) contendo os 3 arquivos fonte (*i.e.,* `.c` e `.java`).

Observações

- Não usar semáforos. Evitar *busy-wait*. Priorizar `join` para sincronizar término.
- Java Threads: `https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html`
- POSIX Threads: `https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html`