Laboratório 5: Sincronização usando Mutex

1. Objetivos

- Desenvolver aplicações concorrentes que acessam recursos compartilhados.
- Compreender o mecanismo de exclusão mútua para proteger regiões críticas.

2. Materiais

- Distribuição Linux/Unix
- Ambiente de desenvolvimento para C/C++.
- Bibliotecas de programação: *pthreads* e outras.

3. Atividades

1. Faça um aplicativo com threads que processa uma matriz e realiza a soma dos elementos. Cada thread deve realizar a leitura de uma linha por vez e a armazenar a soma em uma variável compartilhada. Por exemplo, considere 3 threads (t1, t2, t3) e uma matriz de 10 linhas. As threads não sabem qual linha farão a leitura, até o momento que precisarem de mais dados para processar. Assim, a t1 pode ler linhas 0,2,8,9, a t2 pode ler as linhas 1,6, a thread 3 pode ler as linhas 3,4,5,7. É importante observar que após leitura de cada linha, a soma é armazenada em uma variável compartilhada antes de continuar a próxima leitura. Ao final, exibe a soma total.
2. Faça um aplicativo que execute dois tipos de threads: **ping** e a **pong**. As threads **ping** imprimem na tela a mensagem *ping* e as threads **pong** imprimem na tela a mensagem *pong*. As threads devem cooperar entre si para imprimir sempre alternadas as palavras *ping* e *pong*. O programa deve possibilitar informar a quantidade de threads a serem executadas de cada tipo.

Instruções para entrega via Moodle:

Colocar a solução de cada questão em uma pasta separada nomeada por ex1, ex2 respectivamente.
Cada pasta deve conter o *Makefile* para o exercício e *README* (se necessário).
Inclua em todos os arquivos de código fonte um cabeçalho com a funcionalidade, autor(es) e data.
Adicione comentários antes dos nomes das funções descrevendo a finalidade e os parâmetros de entrada e saída.
Adicione comentários nos principais trechos de códigos do programa.
Compactar em um único arquivo (**tar.gz**) e enviar via Moodle.
Os exercícios podem ser feitos **duplas** ou **trio**.
