Projeto 1. Resolvendo problemas clássicos de concorrência e sincronização usando Semáforos e Monitores

1. Objetivos

- Desenvolver aplicações concorrentes que acessam recursos compartilhados.
- Compreender os mecanismos de semáforos e monitores para sincronização.

2. Materiais

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

3. Descrição

O professor Campiolo solicitou aos alunos de Sistemas Operacionais (SO) a apresentação de conteúdos da disciplina para alunos de outros períodos de Ciência da Computação. As apresentações ocorrerão à tarde em uma sala fechada. O professor ficou responsável por controlar o número de apresentadores e de espectadores na sala. A ordem de apresentação é a ordem de chegada na sala. Todo aluno que irá apresentar, deve assinar a lista na entrada da sala. Quando o professor avisa que as apresentações vão iniciar, ninguém mais pode entrar na sala. No entanto, os alunos que assistem podem sair a qualquer momento. Após todas as apresentações, o professor atribui as notas para os alunos de SO. Após receberem a nota, os alunos de SO devem assinar a lista na saída da sala e saírem. O professor volta a aguardar um tempo para que novos alunos de SO e espectadores de Computação cheguem para as apresentações. Após o término de todas as apresentações, o professor deve esperar todos se retirarem e fechar a porta. Faça uma implementação usando semáforos e outra usando variáveis de condição e mutex (alternativa a monitores em C) para simular e controlar o comportamento das entidades: professor, alunos SO e alunos Computação.

Considerações:

- tipos de threads: Professor, Alunos SO (apresentadores), Alunos Computação (plateia) .
- Há **N** alunos de SO e somente uma parte deve entrar por vez na sala (o professor controla por meio do iniciar apresentações).
- Professor executa as ações: **`iniciar_apresentacoes`**, **`liberar_entrada`**, **`atribuir_nota`**, **`fechar_porta`**.
- Alunos SO executam as ações: **`entrar_sala`**, **`assinar_lista_entrada`**, **`aguardar_apresentacoes`**, **`apresentar`**, **`assinar_lista_saida`**.
- Alunos de Computação executam as ações: **`entrar_sala`**, **`assistir_apresentacao`**, **`sair_apresentacao`**.
- exiba mensagens para mostrar as ações, por exemplo, *alunoSO_1 entra na sala*, *professor inicia apresentacoes*, *alunoComputacao_3 entra na sala*, e assim por diante.

Instruções para entrega via Moodle:

Colocar a solução de cada questão em uma pasta separada nomeada por `ex_sem`, `ex_mon` respectivamente.
Cada pasta deve conter o *Makefile* para o exercício e *README*.
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 **individual ou duplas** .
