Trabalho Prático: Sistema de Controle de Tráfego Aéreo Concorrente em C

1. Introdução

Este trabalho prático propõe a simulação de um **Sistema de Controle de Tráfego Aéreo (ATC)** em um espaço aéreo dividido em setores. O objetivo é aplicar os conceitos de **programação concorrente** e **sincronização de *threads*** para gerenciar o movimento de múltiplas aeronaves, garantindo a segurança (ausência de colisões) e a eficiência (fluxo contínuo e priorização).

O desafio reside em modelar o ATC como um recurso compartilhado que deve conceder permissão de acesso a setores aéreos, evitando **colisões** entre aeronaves e gerenciando a **prioridade** de vôos de emergência.

2. Objetivo

Implementar, em linguagem C, um simulador de um sistema de controle de tráfego aéreo utilizando *threads* POSIX (pthreads). O sistema deve gerenciar o acesso de aeronaves a setores aéreos exclusivos, utilizando mecanismos de sincronização para:

- Garantir que **apenas uma aeronave** ocupe um setor aéreo por vez;
- Implementar um mecanismo de **prioridade** para aeronaves em situação de emergência;
- Assegurar que **nenhuma colisão** (acesso simultâneo ao mesmo setor) ocorra.

3. Especificação do Sistema

3.1. Entidades

3.1.1. Setores Aéreos (Recursos Compartilhados)

- O espaço aéreo é modelado como uma sequência linear de **M** setores;
- Cada setor (S_0, S_1, …, S_{M-1}) é um recurso crítico que deve ser acessado de forma exclusiva (no máximo uma aeronave por setor);
- Uma aeronave precisa adquirir acesso ao próximo setor de destino **antes** de liberar o acesso ao setor no qual se encontra atualmente.

3.1.2. Aeronaves (*Threads*)

O simulador deverá criar **N** *threads*, cada uma representando uma aeronave em vôo. Cada aeronave deve possuir, ao menos os seguintes atributos:

- ID: Identificador único;
- Prioridade: Um número inteiro (sem sinal) entre 1 e 1.000 que indica a prioridade da aeronave para ingressar nos setores. Quanto maior o número, maior a prioridade;
- Rota: Uma sequência de setores a serem atravessados pela aeronave (ex: S_0 ao S_1, S_1 ao S_2, S_2 ao S_3). A rota (sequência de setores) gerada por uma aeronave pode ser de qualquer ordem, ter tamanho diferentes e deve ser estabelecida quando a aeronave é criada.

O ciclo de vida de uma aeronave é:

1. **Solicitar** acesso ao próximo setor de sua rota (setor de destino);
2. **Aguardar** a liberação do setor de destino (se ocupado);
3. **Adquirir** o acesso ao setor de destino;
4. **Liberar** o acesso do setor em que se encontrava anteriormente (setor de origem);
5. **Simular vôo no setor** utilizando sleep() por um tempo aleatório;
6. **Repetir** os passos até o final da rota.

3.2. Mecanismo de Controle Centralizado (*Thread*)

Para gerenciar a prioridade e evitar problemas de concorrência (aeronaves presas esperando umas pelas outras), deve ser implementado um **Mecanismo de Controle Centralizado** (executado por uma *thread* de controle). Este controle deverá usar mecanismos de controle de concorrência (*mutexes*, semáforos e/ou variáveis de condição) para:

- **Controlar Prioridades:** Quando uma aeronave solicita acesso a um setor ocupado, ela deve ser colocada em uma fila de espera em função da sua prioridade. Assim que o setor for liberado, a aeronave de **maior prioridade aguardando na fila** deve ser a próxima a adquirí-lo, **passando à frente** de aeronaves **menos prioritárias** que também estejam esperando;
- **Prevenir Deadlocks:** O sistema deve implementar uma política de alocação de recursos que garanta que o movimento circular de espera não ocorra. Uma sugestão é a implementação de uma política de "tentativa e retenção" controlada, onde a aeronave deve liberar ambos os setores (atual e futuro) se não conseguir o setor de destino em um tempo limite, ou a implementação de uma lógica inspirada no "Algoritmo do Banqueiro" (https://pt.wikipedia.org/wiki/Algoritmo_do_banqueiro) para setores aéreos.

4. Requisitos de Implementação

- O código deve ser escrito em **linguagem C** e compilado com **gcc**. Juntamente com o código-fonte deverá ser fornecido um **Makefile** para compilação;
- Deve-se utilizar a biblioteca ***pthreads***;
- O programa deve aceitar como argumentos de linha de comando:
  - Número de Setores Aéreos (**M**); e
  - Número de Aeronaves (**N**).
- O estado de cada aeronave (setor atual, setor solicitado, prioridade, etc…) deve ser impresso no console de forma clara e legível, com *timestamps* (marcações de tempo) para facilitar a análise da concorrência e da prioridade;
- Ao final da simulação, o sistema deve **apresentar o tempo médio** de espera de cada aeronave para ingressarem nos setores.

5. Entrega e Avaliação

5.1. Grupos

O trabalho deverá ser realizado em grupos de 3 alunos(as). A escolha dos grupos deverá ser feita utilizando a ferramenta disponível no Moodle.

5.2. O que Entregar

O trabalho deverá ser entregue até às 11h59 do dia 03/12/2025 através do Moodle. Não serão aceitas entregas por email ou fora do prazo estipulado. A entrega deverá ser feita apenas por um membro do grupo.

- **Código-fonte:** Todos os arquivos **.c** e **.h** necessários para compilar e executar o sistema em um único arquivo compactado (“zip”);
- **Relatório (Documento PDF):** Um relatório técnico de 5 a 10 páginas contendo:
  - **Introdução:** Breve descrição do problema e dos objetivos;
  - **Design de sincronização:** Detalhamento do mecanismo de controle centralizado, explicando como os mecanismos de controle de concorrência utilizados protegem os setores e como as filas de espera e as prioridades são gerenciadas;
  - **Análise de segurança:** Explanação sobre a **segurança do sistema**, ou seja, como é garantido que não ocorram *deadlocks*;
  - **Análise de prioridade:** Capturas de tela da execução e análise comparativa do tempo médio de espera das aeronaves, demonstrando que a prioridade foi efetivamente implementada.

5.3. Critérios de Avaliação

A avaliação será feita com base na solução entregue pelo grupo (***S***), composta pelo código-fonte e relatório, e na apresentação individual de cada aluno(a) ao professor (A_i). A nota atribuída à cada aluno(a) *i* no trabalho ($NotaTrabalho_i$) será calculada da seguinte forma:

$$NotaTrabalho_i = (A_i * S) / 10$$

ATENÇÃO: Como indicado pela fórmula mostrada acima, **a nota atribuída à solução adotada será ponderada pelo desempenho do(a) aluno(a) durante a apresentação do trabalho**. Por exemplo, se o professor atribuir nota 10 para a solução adotada pelo grupo (***S=10,0***) mas o(a) aluno(a) receber nota 5,0 pela apresentação, devido ao desconhecimento dos conteúdos teóricos, práticos e/ou da solução do trabalho, a sua nota final do trabalho será 5. A ausência no dia da apresentação ou recusa de realização da apresentação do trabalho implicará em nota zero na apresentação, fazendo com que a nota final atribuída ao(à) aluno(a) também seja zero.

A tabela abaixo apresenta os critérios e respectivos pesos que serão utilizados para avaliação da solução (***S***) desenvolvida pelo grupo:

Critério	Peso (%)	Descrição
Segurança e Funcionalidade	40%	O sistema compila, executa e simula o movimento das aeronaves. Nenhuma colisão deve ser observada.
Implementação de Prioridade	30%	Uso correto e eficiente dos mecanismos de sincronização para implementar a fila de espera com prioridade, garantindo que aeronaves mais prioritárias sejam atendidas primeiro.
Qualidade do Código	15%	Legibilidade, comentários, modularidade e tratamento de erros.
Relatório	15%	Qualidade do relatório técnico, análise aprofundada da segurança e prioridade.