Relatório de Análise — Escalonador com Listas de Prioridade e Prevenção de Inanição

1. Justificativa de Design

Decidimos utilizar uma lista ligada simples porque ela torna operações básicas como inserir e remover processos bem rápidas (tempo constante O(1)). Essa escolha simplifica a implementação e mantém o desempenho estável mesmo com várias operações.

2. Complexidade

As operações de adicionar no final (addEnd) e remover no início (removeFront) acontecem em O(1). Já a remoção de um processo específico pelo identificador (removeById) é O(n), pois exige percorrer a lista. Consideramos esse equilíbrio aceitável para o objetivo do trabalho.

3. Estratégia contra Inanição

Para evitar que processos de prioridade média ou baixa fiquem indefinidamente sem execução, criamos um contador: depois que cinco processos de prioridade alta são executados, o sistema força a execução de um processo de prioridade menor. Isso mantém a justiça no escalonamento.

4. Bloqueio de Processos

Quando um processo pede acesso ao DISCO pela primeira vez, ele vai para a lista de bloqueados. Apesar de simples, essa solução já aproxima o comportamento do sistema de um cenário real, onde processos podem ficar temporariamente parados aguardando recursos.

5. Pontos Fracos e Melhorias Futuras

O bloqueio foi tratado de forma simplificada. Uma melhoria possível seria aplicar técnicas de aging ou simular dispositivos com latência, o que deixaria o escalonador ainda mais realista e completo.

Conclusão

No geral, o sistema atendeu bem aos requisitos. Os registros (logs) ficaram claros e ajudam a acompanhar o funcionamento. A estrutura escolhida se mostrou eficiente e o controle de prioridades funcionou como esperado. Apesar de haver espaço para melhorias, a implementação cumpre o que foi proposto de forma consistente.