Relatório de Análise e Justificativa de Design
Disciplina: Algoritmos e Estruturas de Dados I
Professor: Dimmy Magalhães
Aluno: Rhuan Douglas Martins e sillva
Matrícula: 0030746

1. Justificativa de Design
A solução utiliza listas duplamente encadeadas implementadas manualmente para representar
as filas de processos de cada nível de prioridade e a fila de bloqueados. Essa estrutura permite inserção no final e remoção no início em tempo constante O(1) ao manter referências
para cabeça e cauda, o que é ideal para filas (FIFO) do escalonador. A manipulação explícita
de nós foi escolhida para cumprir a restrição de não utilizar coleções prontas e para ter controle total das operações de enfileirar/desenfileirar e ligar/desligar ponteiros.

2. Complexidade (Big-O)
- Inserir no final (adicionar): O(1)
- Remover do início (removerCabeca): O(1)
- Verificar se está vazio: O(1)
- Listar elementos (impressão): O(n) para imprimir toda a fila
No cenário do scheduler, cada ciclo realiza um número constante de operações de remoção/
inserção (exceto impressão), então custo por ciclo é O(1) amortizado, com custo de saída O(n) proporcional ao total de processos quando listamos as filas para logging.

3. Análise da Anti-Inanição
Implementamos um contador que registra quantas execuções consecutivas de processos de alta prioridade foram realizadas. Após 5 execuções consecutivas de alta prioridade, o scheduler força a execução de um processo da prioridade média (se existir), caso contrário
tenta da baixa. Isso garante que processos de prioridade média/baixa tenham oportunidade de progresso, reduzindo risco de starvation. Sem essa regra, um fluxo contínuo de processos
de alta prioridade poderia atrasar indefinidamente processos menos prioritários.

4. Análise do Bloqueio (recurso "DISCO")
Ciclo de vida de um processo que solicita DISCO:
- O processo está na sua lista de prioridade (alta/média/baixa).
- Ao ser selecionado para execução, se ele requer "DISCO" e ainda não foi bloqueado, ele é movido para a lista de bloqueados (flag jaBloqueado = true) e não executa.
- No início de cada ciclo subsequente, o processo mais antigo da lista de bloqueados é desbloqueado (removido da lista de bloqueados), sua flag de bloqueio é resetada e ele é reinserido no final da sua lista de prioridade original, voltando a disputar CPU.
- Essa política modela um atraso e posterior retorno, simulando espera por recurso e evitando que o processo monopolize CPU enquanto espera.

5. Ponto fraco e melhoria proposta

Ponto fraco: impressão do estado das filas a cada ciclo é O(n) e pode degradar desempenho com muitas linhas quando o número de processos cresce. Além disso, a política de desbloqueio libera apenas um processo por ciclo; se muitos processos ficarem bloqueados, o desbloqueio pode se tornar um gargalo.

Melhoria proposta: usar um contador de prioridades dinâmico ou aging:

- Aging: incrementar a prioridade efetiva de processos que esperam muito tempo para promover gradualmente processos de baixa/média prioridade, reduzindo a necessidade de uma regra rígida de anti-inanição.
- Para o desbloqueio, permitir liberar até K processos por ciclo ou simular tempos de espera variáveis (ex.: cada bloqueio dura T ciclos), controlando melhor o throughput.

Conclusão

A implementação atende aos requisitos do enunciado, usa estruturas implementadas do zero, garante prevenção de inanição, e modela bloqueio por recurso. As melhorias propostas podem ser adotadas para escalar para maiores volumes de processo.