Relatório de Análise – Escalonador de Processos

Este relatório apresenta a análise do projeto de implementação de um escalonador de processos utilizando listas duplamente encadeadas circulares. O objetivo do trabalho foi desenvolver uma estrutura própria para gerenciar processos em filas de prioridade, respeitando as regras definidas em enunciado: execução cíclica (round-robin), anti-inanição e bloqueio/desbloqueio por recurso de I/O (DISCO).

A estrutura escolhida para implementar as filas foi a lista duplamente encadeada circular. Essa decisão se deve ao fato de que ela permite inserções e remoções em tempo constante (O(1)) nos extremos da lista, o que é ideal para simular filas de processos. Além disso, a circularidade evita a necessidade de casos especiais para reconectar cabeça e cauda.

A análise de complexidade é o ponto central deste relatório. As principais operações do escalonador são: inserir um processo no final da fila, remover um processo do início da fila e executar um ciclo de CPU. Todas essas operações foram projetadas para funcionar em tempo constante O(1). - Inserção no final da fila (addFinal): como a lista mantém ponteiros diretos para cabeça e cauda, a inserção no final não exige percorrer a lista. O processo é simplesmente conectado à cauda e os ponteiros ajustados, garantindo O(1). - Remoção do início da fila (removeInicio): basta atualizar o ponteiro da cabeça para o próximo nó e reconectar com a cauda. Assim como a inserção, é feito em tempo constante O(1). - Execução de um ciclo (executarCicloDeCPU): a escolha do processo é feita diretamente verificando se as filas estão vazias, seguindo a ordem de prioridade. Não há percorrimento da lista; apenas remoções e inserções já descritas em O(1). O controle de anti-inanição é feito com um contador simples, também em tempo constante. Assim, cada ciclo do escalonador custa O(1). Na leitura inicial do arquivo de entrada, temos custo proporcional ao número de processos. Para n processos, a leitura e criação das instâncias de Processo é O(n). Portanto, o custo total do programa pode ser modelado como O(n), já que cada processo será inserido e removido um número finito de vezes da fila.

O mecanismo de anti-inanição foi implementado através de um contador que rastreia quantas vezes seguidas processos da fila de alta prioridade foram atendidos. Quando esse contador atinge 5, o escalonador é forçado a atender um processo da fila de prioridade média ou baixa. A checagem e atualização desse contador são operações em O(1), não afetando a complexidade geral.

Os bloqueios e desbloqueios por recurso de I/O (DISCO) foram simulados mantendo uma fila específica de processos bloqueados. No início de cada ciclo, um processo é retirado dessa fila e retornado à sua prioridade original. Tanto a inserção quanto a remoção nessa fila também são operações em O(1).

Um teste de desempenho foi realizado com 100.000 processos gerados automaticamente. O tempo total de execução foi de 3,66 segundos, confirmando a eficiência do algoritmo. Esse resultado é coerente com a análise teórica, pois o custo total cresce linearmente com a quantidade de processos, como esperado para uma implementação O(n).

Conclui-se que a implementação atende aos requisitos do trabalho, garantindo correção no funcionamento do escalonador, cumprimento das regras de prioridade, anti-inanição e bloqueios, além de apresentar um desempenho satisfatório em cenários de alta carga. A escolha da lista duplamente encadeada circular foi adequada por permitir operações críticas em tempo constante, resultando em uma complexidade global O(n).