Relatório de Análise do Escalonador

1. Justificativa de Design

A Fila Circular é eficiente porque suas operações principais de inserir e remover processos são em tempo constante **O(1)**. Isso garante que o escalonador não fique lento mesmo com muitos processos nas filas.

2. Análise de Complexidade (Big-O)

As operações essenciais da estrutura de dados têm complexidade O(1) (tempo constante), o que é ideal para um componente de sistema que precisa ser rápido:

- `inserir(processo)`: O(1)
- `remover()`: O(1)
- `estaVazia()`: O(1)

3. Análise da Anti-Inanição

A lógica garante justiça forçando a execução de um processo de prioridade média ou baixa após 5 execuções consecutivas de alta prioridade, zerando um contador em seguida. Sem essa regra, os processos de baixa prioridade poderiam nunca ser executados se a fila de alta prioridade estivesse sempre cheia, causando inanição.

4. Análise do Bloqueio

O ciclo de vida de um processo que precisa de "DISCO" é:

1. Entra em sua fila de prioridade normal.
2. Na primeira vez que é selecionado, não executa e é movido para o final da **lista de bloqueados**.
3. No início de um ciclo futuro, é removido da lista de bloqueados e reinserido no final de sua **fila de prioridade original**.
4. Quando selecionado novamente, ele executa normalmente.

5. Ponto Fraco

O principal gargalo teórico é a

verificação sequencial das filas de prioridade (se alta não vazia, senão se média não vazia...). Uma melhoria seria usar uma única estrutura de dados mais avançada, como um Heap de Prioridade, onde o processo mais importante estaria sempre no topo, acessível em tempo O(1).