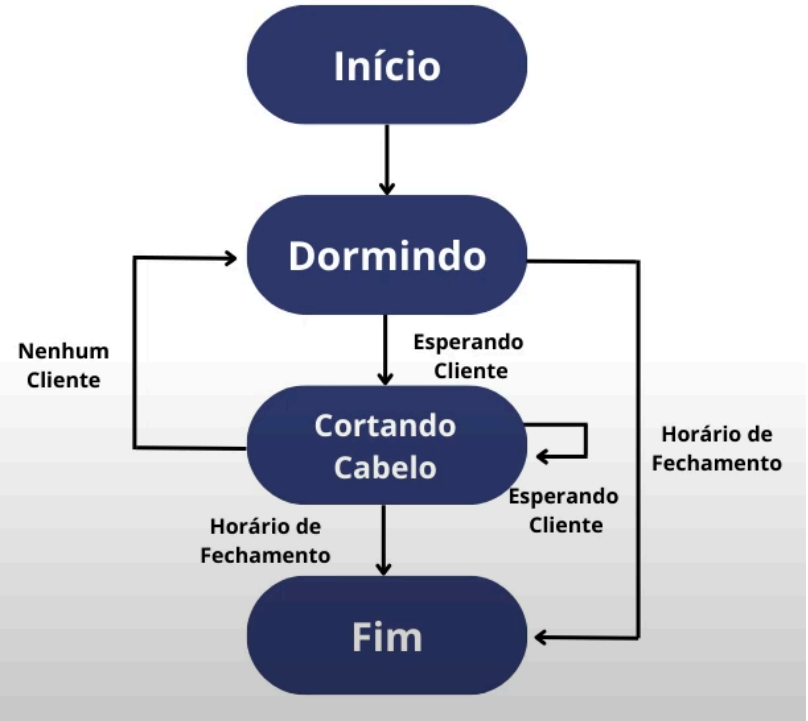
Aluno: Rafael Palheta Tokairin

Relatório: Problema do Barbeiro Sonolento

Fluxograma:



Descrição do Problema

O problema do *Barbeiro Sonolento* é um clássico da computação que aborda questões de **sincronização** em sistemas concorrentes. Ele ilustra a necessidade de coordenação entre múltiplas threads, representando o barbeiro e os clientes, para evitar situações como **condições de corrida**, **deadlocks** ou **espera ocupada**.

A situação é modelada da seguinte forma:

1. Existe **um barbeiro**, **uma cadeira para o corte** e uma **sala de espera** com um número limitado de cadeiras.
2. Quando um cliente chega:
    2.1. Se o barbeiro estiver dormindo (nenhum cliente sendo atendido), o cliente acorda o barbeiro e senta-se na cadeira de corte.
    2.2. Se o barbeiro estiver ocupado, o cliente espera em uma das cadeiras disponíveis da sala de espera.
    2.3. Se não houver cadeiras livres na sala de espera, o cliente vai embora.
3. Quando o barbeiro termina de cortar o cabelo de um cliente:
    3.1. Se houver clientes na sala de espera, o barbeiro chama o próximo cliente.
    3.2. Se não houver clientes esperando, o barbeiro volta a dormir.

Análise do Problema

O desafio principal está em gerenciar o acesso à **fila de clientes** de maneira segura e eficiente, evitando inconsistências causadas por threads que tentem acessar os mesmos recursos ao mesmo tempo. Este problema pode ser dividido em três subproblemas:

1. **Sincronização**: É necessário garantir que apenas uma thread acesse a fila de clientes por vez para evitar problemas como alterações simultâneas.
2. **Controle de Espera**: Gerenciar adequadamente o número de cadeiras na sala de espera para evitar que mais clientes sejam adicionados do que o permitido.
3. **Interatividade entre Threads**: As ações do barbeiro e dos clientes precisam estar sincronizadas para garantir que o barbeiro esteja ocupado apenas quando há clientes e que os clientes sejam atendidos na ordem correta (FIFO).

Implementação da Solução

A solução implementada no código utiliza as seguintes abordagens:

1. **Estruturas de Dados**:
   1.1. Uma **fila circular** é usada para gerenciar os clientes de maneira eficiente, garantindo a ordem FIFO.
   1.2. Variáveis globais *begin*, *end* e *size* monitoram o estado da fila (início, fim e quantidade de clientes).
2. **Mutual Exclusion**:
   2.1. Um **mutex** (*pthread_mutex_t travaFila*) é usado para garantir que apenas uma thread (barbeiro ou cliente) acesse a fila de clientes por vez, evitando condições de corrida.
3. **Funções Principais**:
   3.1. ***enqueue***: Adiciona um cliente na fila (ou indica que ele foi embora, caso a sala de espera esteja cheia).
   3.2. ***denqueue***: Remove um cliente da fila para ser atendido pelo barbeiro.
   3.3. ***cliente***: Simula a chegada de um cliente, tentando entrar na fila.
   3.4. ***barbeiro***: Simula o ciclo de trabalho do barbeiro, atendendo os clientes na fila.
4. **Threads**:
   4.1. Uma thread principal representa o barbeiro, que continuamente verifica e atende clientes.
   4.2. Threads adicionais são criadas para cada cliente que chega, simulando o comportamento de múltiplos clientes concorrentes.