Verenilson da Silva Souza, Guilherme Andrade de Medeiros

Algoritmo de Exclusão Mútua

1. Introdução:

A exclusão mútua é uma solução utilizada para evitar conflitos e garantir que recursos compartilhados sejam acessados de forma segura e ordenada em ambientes distribuídos. Existem algumas formas de implementar a exclusão mútua:

- Centralizada: Nesse modelo, um servidor único é responsável pelo gerenciamento do acesso à área crítica. Os processos solicitam autorização ao servidor antes de entrar na seção crítica.
- Descentralizada ou Distribuída: Aqui, os próprios processos coordenam entre si para gerenciar o acesso à área crítica. Mensagens são trocadas para garantir que apenas um processo entre na seção crítica por vez.
- Híbrida: Essa abordagem combina elementos dos modelos centralizado e distribuído. Pode envolver um coordenador centralizado, mas também permite a coordenação direta entre processos quando necessário.

2. Apresentação do algoritmo implementado:

O algoritmo está dividido em três classes:

1. App.java
2. ExclusaoMutua.java
3. Master.java

Agora, falaremos um pouco sobre como cada uma das classes mencionadas acima funcionam.

A classe **App** cria e gerencia múltiplas threads e uma thread mestre em Java.

Funcionalidades:

- Cria um array de threads "exclusaoMutua" com tamanho definido.
- Inicia cada thread do array, passando o índice como argumento.
- Cria e inicia a thread "master".

A classe **ExclusaoMutua** representa um thread individual que deseja acessar a sessão crítica. Ela funciona da seguinte forma:

1. **Solicita acesso à região crítica:**
    - Procura um servidor disponível (master) via multicast.
    - Envia uma mensagem ao mestre solicitando permissão para acessar a região crítica.
2. **Aguarda permissão:**
    - Espera a resposta do mestre para saber se foi selecionada para acessar a região crítica.
3. **Acessa a região crítica (se selecionada):**
    - Registra o acesso do thread na região crítica em um arquivo de texto.
    - Envia uma mensagem final ao mestre informando o término do acesso.

A classe **Master** gerencia o acesso à região crítica e coordena a comunicação com os threads. Ela utiliza *multicast* para comunicar-se de forma eficiente com todas as threads. Ela funciona da seguinte forma:

1. **Anuncia disponibilidade:**
    - Envia periodicamente mensagens em multicast informando que o servidor está disponível para receber solicitações de acesso à região crítica.
2. **Coleta solicitações:**
    - Espera por solicitações enviadas pelas threads via multicast.
    - Armazena os IDs das threads que solicitaram acesso.
3. **Seleciona uma thread:**
    - Seleciona aleatoriamente uma thread da lista de solicitações pendentes.
4. **Concede acesso:**
    - Envia uma mensagem em multicast para a thread selecionada, concedendo permissão para acessar a região crítica.
5. **Espera término de acesso:**
    - Espera a thread selecionada enviar uma mensagem finalizando o seu acesso à região crítica.