Relatório da implementação do Algoritmo do Banqueiro

Lucas Felipe Barbosa

Introdução

O algoritmo do banqueiro é uma técnica muito utilizada em sistemas operacionais para evitar deadlocks em ambientes com múltiplos processos concorrentes que compartilham recursos. Esse algoritmo realiza a verificação se um sistema permanece em um estado seguro a cada alocação de recursos, para que não ocorram deadlocks ou outros problemas.

O trabalho prático desenvolvido nessa disciplina implementa uma versão multithread desse algoritmo, simulando um cenário em que clientes solicitam e liberam recursos de forma concorrente, com controle de acesso através de mutex para evitar condições de corrida.

Desenvolvimento

Por fins de familiaridade já que os exemplos de algoritmos mostrados em sala de aula comumente são nessa linguagem, o programa foi desenvolvido em C. A estrutura do código segue os requisitos do algoritmo do banqueiro:

Estruturas de Dados:

disponivel[]: Armazena os recursos disponíveis.

maximo[][]: Define a demanda máxima de cada cliente.

alocado[][]: Registra os recursos já alocados.

necessidade[][]: Calcula a necessidade remanescente (maximo - alocado).

Funcionalidades Principais:

verifica_seguranca(): Determina se o sistema está em um estado seguro após a simulação de uma alocação.

solicitar_recursos(): Atende ou nega pedidos com base na segurança.

liberar_recursos(): Devolve recursos ao banco.

Controle de Concorrência:

Um mutex global (mutex_banco) protege as estruturas de dados compartilhadas.

Cada thread (cliente) opera em um loop, solicitando e liberando recursos aleatoriamente.

Inicialização:

O programa recebe os recursos disponíveis via linha de comando (Exemplo: ./banqueiro 10 5 7).

As matrizes maximo e necessidade são preenchidas aleatoriamente.

Resultados

O programa foi testado com 5 clientes e 3 tipos de recursos, gerando os seguintes comportamentos:

1. Pedidos atendidos: os recursos são alocados, já que o ssistema permanece seguro;
2. Pedidos negados: pedido é recusado devido à alocação ter levado a um estado inseguro;
3. Liberação de recursos: devolução de recursos após uso, aumentando a disponibilidade.

Exemplo de Saída:

Cliente 0 tentando solicitar...
Cliente 0: Pedido ATENDIDO.
Cliente 0 usando recursos...
Cliente 0 liberando recursos...
Cliente 1: Recursos indisponíveis, esperando...

O uso de mutex garantiu a ausência de condições de corrida, e o algoritmo de segurança preveniu deadlocks.

Conclusão

A implementação do algoritmo feita demonstrou a eficácia do Algoritmo do Banqueiro em gerenciar recursos compartilhados de forma segura e sem deadlocks. A combinação de threads, mutex e verificação de segurança atendeu aos requisitos do problema, simulando um ambiente concorrente realista.