UNIVERSIDADE DO VALE DO ITAJAÍ
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

Sistema Operacionais

JALLISON ALFREDO JIMENEZ
MATHEUS VOLTOLINI

ITAJAÍ 2025

Sumário

Resumo do Projeto

O objetivo do projeto foi desenvolver um sistema cliente-servidor utilizando a linguagem C++ para simular operações básicas de um banco de dados (INSERT, DELETE, SELECT e UPDATE). O sistema utiliza comunicação entre processos via FIFO (Named Pipes) e múltiplas threads no servidor para o processamento paralelo de requisições. As informações dos registros são armazenadas em uma lista duplamente encadeada.

1.Contexto e Explicação da Aplicação

O projeto foi proposto no contexto de estudos sobre comunicação entre processos (IPC) e programação concorrente. Em sistemas reais, servidores devem ser capazes de lidar com múltiplos clientes simultaneamente, garantindo integridade dos dados e tempo de resposta eficiente. A solução adotada simula um banco de dados simples, permitindo testar esses conceitos em um ambiente controlado com ferramentas da linguagem C++ e bibliotecas POSIX.

1.2 Códigos Importantes da Implementação

Para o correto funcionamento da aplicação, é necessário abrir dois terminais. Um será utilizado para o **servidor**, e outro para o **cliente**. Os códigos devem ser compilados e executados da seguinte maneira:

Terminal 1 - Servidor

```
g++ -o servidor Servidor.cpp -pthread
./servidor
```

Terminal 2 - Cliente

```
g++ -o cliente Cliente.cpp
./cliente
```

1.2.1 Servidor (Resumo)

Responsável por Interpretar o comando e executar

```
pthread_mutex_t mutex_write = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t mutex_id = PTHREAD_MUTEX_INITIALIZER;

void* processa_requisicao(void* arg) {
    string comando = *(string*)arg;
    delete (string*)arg;

    // Interpreta comando e executa INSERT, DELETE, SELECT ou UPDATE
}
```

1.2.2 Cliente (Resumo)

```
int fd = open("requisicoes_fifo", O_WRONLY);
string comando = "INSERT,ProdutoX";
write(fd, comando.c_str(), comando.size() + 1);
```

1.2.3 Lista Duplamente Encadeada

```
struct No {
    int id;
    char nome[50];
    No* eloA;
    No* eloP;
};
```

2. Resultado Implementação

Nessa situação foi gerado uma sequência de simulação feita no código, no qual vai ser salvo no arquivo que é feito a lista encadeada

```
[INSERT] Nome inserido: Arroz com ID 1
[INSERT] Nome inserido: Feijão com ID 2
[SELECT] ID: 1, Nome: Arroz
[UPDATE] ID 1 alterado para: Arroz Integral
[DELETE] ID 2 removido.
```

Resultado:

ID	Nome
1	Arroz Integral

3. Análise e Discussão dos Resultados

A implementação demonstrou eficiência na separação entre cliente e servidor, possibilitando múltiplas operações simultâneas por meio do uso de threads. O controle de concorrência com mutexes foi fundamental para garantir a integridade dos dados armazenados na lista. A abordagem com FIFO simplificou a comunicação, embora em aplicações reais o uso de sockets seja mais comum. O projeto permitiu compreender conceitos como exclusão mútua, sincronização e manipulação de estruturas dinâmicas em C++.