Simulador de Sistema de Arquivos com Journaling

Autores:

- Matheus Diógenes Amorim - 2310277
- Ederson Viana Angelim - 2320329

Sobre o Projeto

Este projeto apresenta um Simulador de Sistema de Arquivos (File System) desenvolvido em Java. O objetivo principal é demonstrar a arquitetura lógica de armazenamento de dados, manipulação de diretórios e arquivos, e a implementação de um mecanismo de Journaling para garantir a consistência e auditabilidade das operações.

Metodologia

O simulador foi construído utilizando a linguagem Java, focando na aplicação de conceitos de Orientação a Objetos e Estruturas de Dados.

- **Linguagem:** Java (JDK 8+).
- **Interface:** Console/Terminal (CLI - Command Line Interface).
- **Abordagem:** O sistema simula chamadas de sistema (syscalls) através de métodos que manipulam estruturas de dados em memória.
- **Persistência:** Utiliza serialização de objetos para salvar o estado do disco (`.dat`) e manipulação de arquivos de texto para o log de journaling (`.log`).

Parte 1: Introdução ao Sistema de Arquivos com Journaling

Descrição do Sistema de Arquivos

Um Sistema de Arquivos é o componente do sistema operacional responsável por controlar como os dados são armazenados e recuperados. Ele abstrai o hardware físico (disco) em unidades lógicas conhecidas como "Arquivos" e "Diretórios". Sem ele, os dados seriam apenas um fluxo contínuo de bytes sem início ou fim definidos.

O Conceito de Journaling

O Journaling é uma técnica de tolerância a falhas. Um sistema de arquivos com journaling mantém um diário (log) circular ou incremental onde as mudanças são registradas antes (ou logo após) serem efetivadas na estrutura principal.

Isso previne a corrupção de dados em casos de desligamento abrupto ou travamento do sistema, permitindo que o SO "reproduza" (replay) o log para restaurar a consistência.

Tipos de Journaling Abordados

Na teoria de Sistemas Operacionais, existem três tipos principais:

1. **Write-Ahead Logging (WAL):** Os dados são escritos no journal antes do disco principal.
2. **Ordered Mode:** Apenas metadados são logados, mas garante-se que os dados do arquivo sejam escritos antes dos metadados.
3. **Data Journaling:** Tanto dados quanto metadados são armazenados no journal (mais seguro, porém mais lento).

Neste simulador, implementamos um log textual de operações que atua como um histórico de auditoria e persistência sequencial.

Parte 2: Arquitetura do Simulador

Estrutura de Dados

O sistema utiliza uma estrutura de Árvore (Tree) para representar a hierarquia:

- **Root:** O diretório raiz `/`.
- **Mapas (HashMaps):** Cada diretório possui mapas para armazenar seus filhos (subdiretórios e arquivos), permitindo acesso rápido pelo nome.
- **Serialização:** A classe `Directory` e `File` implementam `Serializable`, permitindo que toda a árvore de diretórios seja convertida em bytes e salva no arquivo `filesystem.dat`.

Implementação do Journaling

O mecanismo de Journaling funciona da seguinte maneira:

1. O usuário solicita uma operação (ex: criar arquivo).
2. O sistema verifica a validade da operação e atualiza a estrutura em memória.
3. Imediatamente, o sistema invoca a classe `Journal`.
4. Uma nova entrada (`JournalEntry`) contendo TIMESTAMP, OPERAÇÃO e CAMINHO é anexada ao arquivo físico `journal.log`.

Parte 3: Implementação em Java

A solução é modularizada nas seguintes classes principais:

FileSystemSimulator

- O "Kernel" do sistema.
- Centraliza a lógica de navegação (`cd`).
- Executa operações CRUD (Create, Read, Update, Delete) em arquivos e pastas.
- Comunica-se com o `Journal` para registrar atividades.

File e Directory

- As entidades do sistema.
- **Directory:** Contém `Map<String, File>` e `Map<String, Directory>`.
- **File:** Contém o conteúdo do arquivo (`String`) e metadados básicos (nome).

Journal

- O gerenciador de logs.
- Responsável por abrir o arquivo `journal.log` em modo append e escrever as entradas formatadas.

JournalEntry

- Objeto que representa uma linha do log.
- **Atributos:** `LocalDateTime timestamp`, `String operation`, `String details`.

Parte 4: Instalação e Funcionamento

Siga os passos abaixo para compilar e executar o simulador em sua máquina.

Pré-requisitos

- Java JDK instalado (versão 8 ou superior).
- Terminal (CMD, PowerShell, Bash) ou IDE (IntelliJ, Eclipse, VS Code).

Passo a Passo

1. Estrutura de Pastas

Garanta que seus arquivos estejam organizados nos pacotes corretos:

```
/src
    /filesystem
        Directory.java
        File.java
        FileSystemSimulator.java
        FileSystemStorage.java
        Journal.java
        JournalEntry.java
    /ui
        Menu.java
    Main.java
    README.md
```

2. Compilação

- Navegue até a pasta raiz do projeto (/src) e execute:

Bash

- javac filesystem/*.java ui/*.java Main.java

3. Execução

Inicie o simulador:

Bash

- java Main

Verificando os Resultados

Após utilizar o sistema, dois arquivos serão gerados na raiz do projeto:

filesystem.dat: O estado salvo do seu sistema de arquivos (binário).

journal.log: Abra este arquivo com um editor de texto para ver o histórico de todas as operações realizadas com data e hora.