Universidade Federal
de São João del-Rei

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
SISTEMAS OPERACIONAIS

TRABALHO PRÁTICO 3

Diogo Augusto Martins Honorato

Guilherme Garcia de Oliveira

Leonardo da Silva Vieira

São João del-Rei

2025

Sumário

1 Introdução

Este trabalho prático, desenvolvido na disciplina de Sistemas Operacionais , teve como objetivo implementar um simulador de sistema de arquivos baseado no padrão FAT16.

Através da simulação, buscou-se compreender os principais conceitos envolvidos na organização de sistemas de arquivos, como estrutura da FAT, clusters, diretórios e alocação de dados. O sistema é controlado via shell interativo e simula de forma realista a leitura e escrita em uma partição virtual, com persistência dos dados.

2 Implementação

A implementação foi feita em linguagem C. O sistema utiliza um arquivo binário (`fat.part`) de 4MB para representar uma partição virtual persistente.

2.1 Layout da Partição

A estrutura da partição foi projetada conforme os padrões do FAT16:

- **Cluster 0 - Boot Block:** inicializado com o byte 0xBB para marcação.
- **Clusters 1–8 - FAT:** contém 4096 entradas de 16 bits (8192 bytes).
- **Cluster 9 - Diretório Raiz:** suporta até 32 entradas.
- **Clusters 10–4095 - Área de Dados:** armazena os dados de arquivos e subdiretórios.

2.2 Tabela FAT

A FAT (File Allocation Table) foi mantida em memória como um vetor de `uint16_t fat[4096]`. Cada entrada representa o estado de um cluster:

- `0x0000`: livre
- `0xFFFF`: fim de arquivo
- `0xFFFE`: reservado para a própria FAT
- `0xFFFD`: boot block

- valores intermediários representam encadeamentos entre clusters de um mesmo arquivo

2.3 Cluster de Diretório e Arquivo

Cada cluster pode conter um vetor de estruturas `dir_entry_t`, representando arquivos/diretórios e dados binários, no caso de arquivos. A estrutura `dir_entry_t` possui 32 bytes, com campos para nome (18 chars), atributos (arquivo ou diretório), cluster inicial e tamanho.

2.4 Comandos e Funcionalidades Implementadas

O shell interativo aceita comandos semelhantes ao de um sistema real. Entre os principais comandos e suas funcionalidades, destacam-se:

- `init`: formata o sistema criando estrutura inicial da FAT e diretório raiz.
- `load`: carrega uma partição já existente de disco.
- `mkdir <caminho>`: cria diretórios em qualquer nível hierárquico.
- `create <caminho>`: cria arquivos vazios.
- `write / append <dados> <arquivo>`: escreve ou adiciona conteúdo, realocando clusters se necessário.
- `read <arquivo>`: lê o conteúdo real do arquivo da partição.
- `unlink <caminho>`: remove arquivos ou diretórios vazios, liberando clusters.
- `ls <caminho>`: lista o conteúdo do diretório em formato tabular.
- `help` e `exit`: para consulta e encerramento.

2.5 Alocação e Escrita de Dados

O sistema implementa alocação encadeada de clusters. Ao escrever em um arquivo, os clusters são alocados dinamicamente, e a FAT é atualizada para formar a cadeia. Ao sobrescrever, os clusters antigos são liberados, garantindo reutilização de espaço.

A escrita é feita em blocos de 1024 bytes, respeitando o limite de cada cluster. O processo de `append` envolve leitura do conteúdo atual, concatenação com os novos dados e regravação do arquivo completo com alocação adequada.

2.6 Remoção Segura

Para arquivos: todos os clusters são liberados na FAT. Para diretórios: apenas se estiverem vazios, evitando perda de dados acidental.

2.7 Persistência

Ao fim de qualquer operação que altera dados ou a FAT, o sistema grava imediatamente as alterações no disco, mantendo consistência entre memória e partição. Isso permite que o arquivo `fat.part` possa ser carregado futuramente com os dados preservados.

2.8 Testes Automatizados

Foi implementado um script test_fat16.sh que simula uma sequência completa de comandos e verifica a criação, leitura, modificação e remoção de arquivos e diretórios. Isso garante uma cobertura básica de funcionalidades e ajuda na detecção precoce de erros em alterações futuras.

3 Resultados e Discussão

O projeto desenvolvido é capaz de executar corretamente todas as operações previstas para arquivos e diretórios. A validação do sistema foi realizada por meio de testes automatizados e demonstrações práticas. O script test_fat16.sh executa uma sequência completa de comandos como inicialização, criação, escrita, leitura, listagem, remoção e carregamento. Sua execução bem-sucedida confirmou o funcionamento correto e integrado das funcionalidades.

A demonstração prática de um cenário típico também comprovou o comportamento esperado. Ao executar comandos como init, mkdir /docs, create /docs/a.txt e write "teste"/docs/a.txt, foi possível observar a formatação da partição, a criação correta das entradas de diretório e a leitura do conteúdo gravado. O comando ls / exibiu a presença

do diretório docs, enquanto ls /docs mostrou o arquivo a.txt com as informações corretas. Por fim, read /docs/a.txt retornou "teste", validando a leitura sequencial.

A persistência do sistema foi verificada com o encerramento e reinicialização do simulador. O comando load restaurou corretamente o estado anterior, com todos os arquivos e diretórios intactos, comprovando que as alterações são gravadas de forma consistente no arquivo fat.part.

A implementação segue os princípios do sistema FAT16, utilizando alocação encadeada e mantendo a tabela FAT em memória para gerenciar o estado dos clusters. As operações de escrita e remoção manipulam essa estrutura de forma atômica, garantindo a integridade dos dados.

Em relação à eficiência, a operação append foi implementada com simplicidade, por meio da leitura completa do conteúdo, concatenação em memória e reescrita do arquivo. Embora funcional, essa abordagem é ineficiente para arquivos grandes. Uma possível melhoria seria localizar o último cluster e encadear novos diretamente a partir dele.

O sistema também apresenta limitações da própria especificação, como o limite de 18 caracteres para nomes de arquivos e até 32 entradas por diretório. Isso ocorre devido ao uso fixo de um cluster de 1024 bytes por diretório. Uma melhoria viável seria permitir diretórios com múltiplos clusters encadeados, como ocorre com os arquivos.

Por fim, destaca-se a confiabilidade do simulador, exemplificada pela verificação de diretórios vazios antes da exclusão com unlink, evitando perda de dados e seguindo práticas comuns em sistemas reais.

Em resumo, os resultados demonstram que o simulador cumpre seus objetivos e serve como ferramenta eficaz para o estudo da implementação e do funcionamento de sistemas de arquivos baseados em FAT.

4 Conclusão

A implementação do simulador FAT16 proporcionou uma imersão profunda na estrutura e funcionamento dos sistemas de arquivos, revelando a complexidade e a engenharia por trás da organização de dados digitais. A capacidade do sistema de replicar, com precisão e consistência, operações cruciais como a criação de diretórios e arquivos, alocação de dados, leitura e escrita com múltiplos clusters e remoção controlada pela FAT validou a compreensão teórica e prática dos conceitos envolvidos.

O rigoroso processo de testes automatizados foi fundamental para confirmar o comportamento esperado em cenários diversos, atestando a estabilidade e robustez do código em todas as condições testadas. Esse projeto não apenas consolidou o conhecimento sobre as operações de baixo nível de um sistema de arquivos, mas também destacou a importância de uma arquitetura bem definida e de metodologias de validação eficientes no desenvolvimento de software.

A experiência com o simulador FAT16 pavimenta o caminho para futuras explorações em sistemas de arquivos mais complexos, como NTFS ou EXT4, e oferece uma base sólida para o desenvolvimento de ferramentas de recuperação de dados ou otimização de armazenamento.

5 Referências

TANENBAUM, A. S. Sistemas operacionais modernos. Rio De Janeiro (Rj): Prentice-Hall Do Brasil, 2010.