Sistemas Operacionais
Trabalho Prático 2 - Documentação

Alunos: **Ana Clara Medina Fonseca**
Lucas Borges Grandolpho

1 INTRODUÇÃO

No universo da computação, um sistema de arquivos é um conjunto de estruturas que permitem ao sistema operacional controlar o acesso ao disco rígido. O desenvolvimento dos sistemas de arquivos é importantíssimo no campo computacional uma vez que sem eles as informações colocadas em um meio de armazenamento seriam dadas em grandes massas de dados, sem alguma maneira de distinguir os locais onde uma determinada parte da informação termina e outra se inicia.

O objetivo deste trabalho prático foi criar um simulador de um sistema de arquivos simplificado, baseado na tabela de alocação de 16 bits (FAT) e realizar algumas operações sobre este.

2 IMPLEMENTAÇÃO

Para a implementação, foram criadas estruturas e rotinas que auxiliam na resolução do problema proposto. Estes elementos serão detalhados a seguir.

2.1 Arquivos *fat.c / fat.h*

No arquivo *fat* foram implementadas algumas estruturas auxiliares para representar o sistema de arquivos. São elas:

- *struct dir_entry_t*: Estrutura da entrada de diretório com 32 bytes cada.
- *uint16_t fat[4096]*: Vetor que representa os 8 clusters da tabela FAT, com um total de 8192 bytes (4096 entradas de 16 bits).
- *struct data_cluster*: Estrutura que representa os diretórios (incluindo *root*), consistindo de 32 entradas de diretório com 32 bytes cada, totalizando 1024 bytes.

2.1.1 Listagem das Rotinas

As funções implementadas neste arquivo serão detalhadas abaixo:

- *int init( )*: Função que inicializa o sistema de arquivos com as estruturas de dados. Será criado o arquivo *fat.part* (ou recriado) com as partições *boot block*, FAT, *root dir* e *data clusters*, sendo cada uma delas preenchidas com os valores adequados.
- *int load( )*: Função que carrega o arquivo *fat.part* que representa os sistema de arquivos. Caso o arquivo *fat.part* ainda não existe, nada será feito.
- *int mkdir(char *nome, int dir)*: Função que cria um novo diretório. Ela procurará a primeira entrada e *cluster* vazios para realizar a criação (ou também se não há espaços vazios, e neste caso a função é encerrada).
- *int ls(char *nome)*: Função que lista os elementos presentes em um diretório.
- *int create(char *nome)*: Função que cria um novo arquivo no diretório. Funciona de maneira análoga a função *mkdir*.
- *int read(char *nome)*: Função que lê o conteúdo de um arquivo.
- *int append(char *nome, char *str, int app)*: Função que anexa dados ao final de um arquivo. Ela procurará onde está o final do arquivo atual para iniciar a escrita.
- *int write(char *nome, char *str)*: Função que escreve dados em um arquivo novo ou sobrescreve em um arquivo existente. Funciona de maneira similar a função *append*, mas neste caso ela não fará a procura pelo fim do arquivo, uma vez que deverá escrever a partir do início dele.
- *int unlink(char *nome)*: Função que exclui um arquivo ou um diretório. Para o caso da exclusão de diretŕorio, ele deverá estar vazio.
- *int entradaVazia( )*: Função que verifica se há e qual é a primeira entrada vazia em um diretório.
- *int clusterVazio( )*: Função que verifica se há e qual é o primeiro *cluster* vazio na FAT.

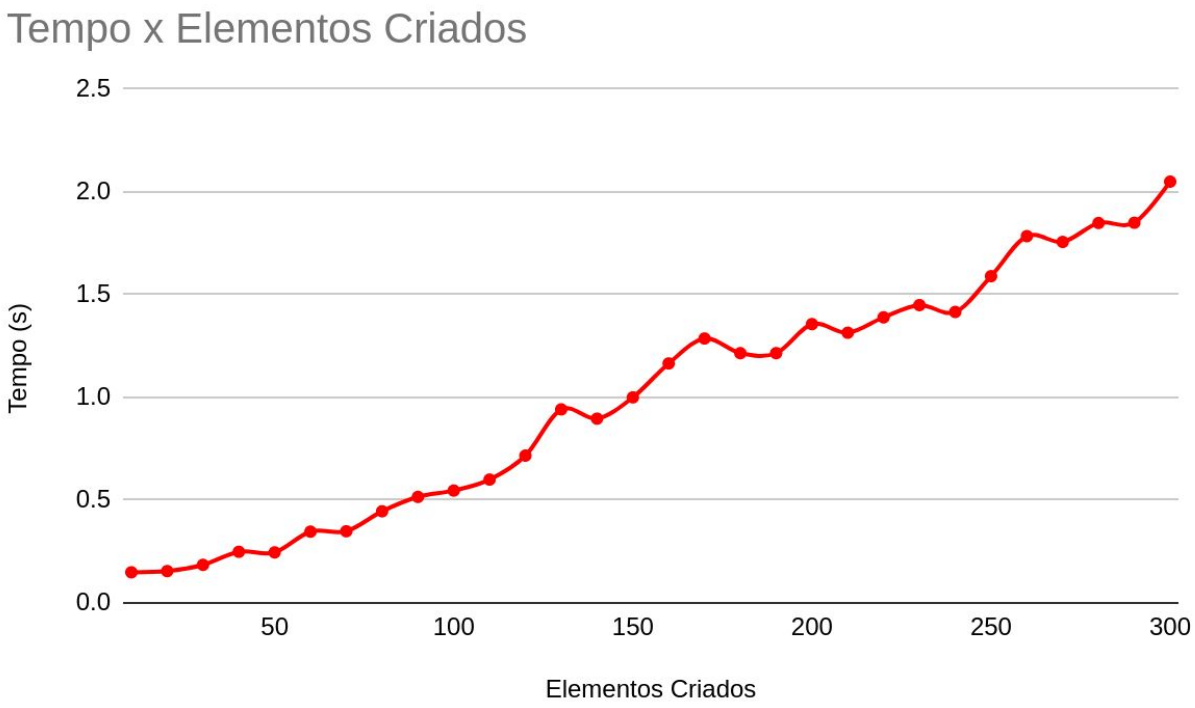
- *int salvaNoArquivo(FILE *arq)*: Função que salva o arquivo após serem realizadas alterações nele.

2.2 Arquivo *main.c*

O arquivo *main.c* tem como função básica o recebimento e interpretação de comandos fornecidos pelo usuário, executando as rotinas adequadas ao que foi solicitado.

3 RESULTADOS E DISCUSSÕES

Para um melhor entendimento do funcionamento do algoritmo, foram feitos testes onde criavam-se arquivos e diretórios de forma automatizada, apenas variando de forma crescente o número de criações realizadas. Observando o tempo médio de cada execução, obteve-se o seguinte gráfico:



4 CONCLUSÃO

Observando os resultados obtidos nos testes, é possível notar que o tempo cresce de uma maneira aproximadamente linear. Isto corrobora com os resultados esperados visto que não há elementos no algoritmo que justificariam um crescimento de outra forma.

De forma geral, a simulação funcionou de forma adequada e relativamente próximo ao esperado, atendendo aos requisitos do problema proposto.

5 REFERÊNCIAS BIBLIOGRÁFICAS

1. TANENBAUM, Andrew S.; BOS, Herbert. *Sistemas Operacionais Modernos*. 4. ed. Amsterdã: Pearson, 2016.
2. MORIMOTO, Carlos E. *Sistemas de Arquivo*. Disponível em: <https://www.hardware.com.br/termos/sistema-de-arquivos>
3. DALEY, R. C.; NEUMANN, P. G. *A General-Purpose File System For Secondary Storage*. Massachusetts. Disponível em: <https://www.multicians.org/fjcc4.html>