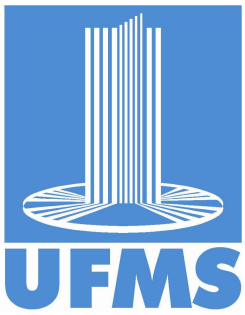

Universidade Federal do Mato Grosso do Sul
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
SISTEMAS OPERACIONAIS

RELATÓRIO DO TRABALHO PRÁTICO DA DISCIPLINA DE SISTEMAS OPERACIONAIS: IMPLEMENTAÇÃO DE UM SISTEMA DE ARQUIVOS UTILIZANDO O FUSE

ISABELA BUZZO GALVÃO

GISLAYNE GARABINI DAMASCENO

LAURA BARAUNA LUDGERO

LUISA CARLA FERREIRA

Campo Grande - MS
2023

1 INTRODUÇÃO

Os sistemas de arquivos desempenham um papel essencial nos sistemas operacionais, permitindo o armazenamento, gerenciamento de segurança, acesso e manipulação de dados no armazenamento. Contudo, sua implementação é complexa e demanda amplo conhecimento devido à sua sofisticação e à necessidade de interação direta com o núcleo do sistema operacional.

O FUSE (Filesystem in Userspace) é uma interface de software que possibilita a criação de um sistema de arquivos a nível de usuário, sem necessidade de alterar o código do núcleo do sistema operacional, o que tornando-o uma ferramenta facilitadora e flexível desse processo. O sistema de arquivos opera diretamente no terminal e é operado por linha de comando, traduzindo as chamadas de sistema para as operações implementadas pelo usuário. Vale ressaltar que o FUSE oferece suporte para o desenvolvimento em diversas linguagens, sendo que, neste relatório, foi implementado utilizando a linguagem C.

2 METODOLOGIA

O trabalho foi realizado pelo grupo de maneira que cada integrante implementasse uma função específica do sistema de arquivos. Para facilitar a comunicação, utilizamos um grupo de mensagens para tirarmos dúvidas, compartilhar conteúdo sobre operação do FUSE e discutir o andamento do trabalho.

Utilizamos um repositório no GitHub para gerenciar o código fonte do projeto e cada funcionalidade foi desenvolvida em uma branch separada.

3 FUNCIONAMENTO DO SISTEMA DE ARQUIVOS IMPLEMENTADO

O sistema de arquivos funciona a partir do diretório representado pela variável *'diretório_raiz'*, que contém informações sobre os diretórios subsequentes. Cada arquivo é representado pela estrutura *MyFile*, incluindo seu nome, conteúdo, tamanho e timestamp. Os diretórios são representados pela estrutura *MyDirectory*, contendo seu nome, variável int para número de arquivos e diretórios e uma lista de seus arquivos e subdiretórios.

As operações básicas implementadas são:

- ***myfs_read***: realiza a leitura de um arquivo. A função recebe como parâmetros o caminho do arquivo (path), um buffer (buffer) para armazenar os dados lidos, o tamanho (size) a ser lido, o deslocamento (offset) e informações do arquivo (struct fuse_file_info *fi). A função procura o arquivo especificado dentro da lista de arquivos do diretório. A variável path, que representa o caminho do arquivo a ser lido, é utilizada no loop acrescida de

1 para o caractere \ ser ignorado. Se o arquivo é encontrado e o deslocamento (offset) somado ao tamanho (size) a ser lido é menor do que o tamanho real do arquivo (diretorio_raiz.arquivos[i].size), calcula o novo tamanho a ser lido (size) para evitar leituras além do final do arquivo. Se o arquivo é encontrado, a função retorna size, do contrário, retorna -ENOENT, indicando erro.

- ***myfs_create***: cria um novo arquivo no sistema de arquivos. A função recebe como parâmetros o nome do arquivo (`MyFS`), o modo ((`mode_t mode`) e informações do arquivo ((`struct fuse_file_info *fi`). A função percorre a lista de arquivos do diretório (diretorio_raiz.arquivos) verificando se há algum arquivo com o mesmo nome. Se encontrar , retorna o erro -EEXIST. O novo arquivo é criado na próxima posição disponível no array de arquivos diretorio_raiz.arquivos. Então, ocorre a atribuição de dados ao diretório, incluindo o tratamento para que o caractere \ seja ignorado. Após isso, o contador de diretórios é incrementado.

- ***myfs_write***: Escreve dados em um arquivo existente. A funçao recebe como parâmetros o caminho do arquivo (`MyFS`), um buffer contendo os dados a serem escritos (`buf`), o tamanho desses dados (`size`), o deslocamento (`offset`) e informações do arquivo (`struct fuse_file_info *fi`). A a função procura o arquivo especificado na lista de arquivos do diretório (diretorio_raiz) comparando seus nomes. Se o arquivo é encontrado, o índice (fileIndex) é atualizado para armazenar a posição do arquivo na lista, caso contrário, a função retorna `-ENOENT`. Se a soma do deslocamento e do tamanho dos novos dados ultrapassar o tamanho atual do arquivo, é feita uma realocação do espaço de memória, se a realocação não puder ser realizada, a função retorna `-ENOMEM`.

- ***myfs_unlink***: remove um arquivo do sistema. A função recebe como parâmetro o caminho do arquivo (`path`). A lista de arquivos (`diretorio_raiz.num_arquivos`) é percorrida para encontrar o arquivo especificado. Se o arquivo é encontrado, a memória associada ao conteúdo do arquivo é liberada. Em seguida, os arquivos seguintes são "movidos para frente"para preencher o espaço deixado pelo arquivo removido. Após isso, o contador de arquivos é decrementada. Se o arquivo não é encontrado, a função retorna `-ENOENT`.

- ***myfs_mkdir***: cria um novo diretório. A função recebe como parâmetros o caminho do diretório a ser criado (`path`) e o modo de permissão (`mode`). A função verifica se o nome do diretório já está sendo utilizado. Se o diretório com o mesmo nome já existir, a função retorna o erro `-EEXIST`. Em seguida, são realizadas alocações de memória para o novo diretório e arquivos e diretórios dentro do novo diretório, cada um com espaço para até 50 elementos. Posteriormente, é feita a criação do diretório.

- ***myfs_readdir***: lista os arquivos e subdiretórios de um diretório. recebe como parâmetros o caminho do diretório (`path`), um buffer (`buf`) para armazenar os nomes dos arquivos e diretórios listados, um ponteiro para a função de preenchimento (`filler`), o deslocamento

(`offset`), e informações do arquivo (`fi`). A função preenche o buffer com o nome do próprio diretório atual, utilizando o ponto (`"."`) como representação do diretório atual. Em seguida, a lista de arquivos e diretórios do diretório é percorrida e o buffer é preenchido com cada um dos elementos das listas.

- ***myfs_mkdir***: cria um novo diretório. A função `myfs_mkdir` recebe como parâmetros o caminho do diretório a ser criado (`path`) e o modo (`mode_t mode`). A função percorre os diretórios existentes (diretorio_raiz.num_diretorios) verificando se há algum diretório com o mesmo nome. Se encontrar, retorna o erro -EEXIST. É alocado espaço na memória para armazenar um novo diretório e para armazenar arquivos e subdiretórios dentro dele. A função trata o nome do novo diretório removendo o primeiro caractere da string e é inicializada a contagem de arquivos e subdiretórios do novo diretório como zero.
- ***myfs_getattr***: obtém atributos de um diretório. A função recebe como parâmetros o caminho do arquivo ou diretório (`path`) e uma estrutura `struct stat` chamada `stbuf` para armazenar informações de atributos. Se o caminho recebido corresponder ao diretório raiz, é preenchida a estrutura `stbuf` com atributos do diretório.

A montagem e utilização do código pode ser realizada da seguinte forma dentro de um ambiente Linux:

- Compile e execute o código com a marcação `-lfuse`.

```
gcc -o executável código.c -lfuse
```

- Monte o sistema de arquivos num diretório de sua escolha.

```
./executável ./diretório
```

- Navegue até o sistema.

```
cd /path/do/sistema
```

- Teste o sistema com as funções implementadas.

```
touch arquivo.txt // criação de arquivo
mkdir novo_diretório // criação de diretório
ls // exibição dos elementos do diretório atual
```

Foi também implementado um tratamento de erros para as seguinte situações:

- Na função de leitura de arquivo, se o arquivo não for encontrado ou se o deslocamento mais o tamanho a ser lido for maior que o tamanho real do arquivo, o sistema retorna um erro indicando que o arquivo não foi encontrado ou que ocorreu um erro de leitura.

- Na função de criação de arquivo, se o arquivo já existir no diretório, a função retorna um erro indicando que o arquivo já existe.
- Na função de escrita de arquivos, se não encontrar o arquivo, a função retorna um erro. Se houver falha na alocação de memória, a função retorna um erro indicando falta de memória.
- Na função de deleção de arquivos, se não encontrar o arquivo, a função retorna um erro.
- Na função de criação de diretórios, se o diretório com o nome especificado já existir, a função retorna um erro indicando que o diretório já existe.

4 RESULTADOS E DISCUSSÕES

Inicialmente, a complexidade do trabalho dificultou a compreensão do objetivo central e a definição de um ponto de partida claro. A escassez de documentação sobre o uso do FUSE na internet também foi um desafo, embora os links providos pela professora tenham sido úteis. Teria sido interessante poder contar com maiores detalhes, exemplos e explicações sobre a execução do trabalho.

Apesar de nosso código compilar, suas funcionalidades não estão funcionando. Primeiramente, tivemos o seguinte erro:

```
touch: não foi possível tocar 'arq.txt': Função não implementada
```

Até descobrirmos a causa foi custoso, pois mesmo com todas as funções implementadas, o erro dizia o contrário. Porém, após adicionar todas as funções, mesmo aquelas não implementadas, na estrutura myfs_operations e declará-las sem corpo, apenas com um retorno 0, nossas funções começaram a serem reconhecidas. Alteramos algumas vezes como trataríamos as estruturas dentro do nosso código. Inicialmente, não estávamos utilizando o diretório raiz como uma estrutura de diretório, e sim uma struct própria para o root, o que estava tornando o código confuso. Também tivemos que discorrer bastante sobre como seriam os processos de criar novos arquivos e diretórios, resultando nessas seções de códigos terem sido alteradas algumas vezes com lógicas diferentes, imaginando que eles seriam a causa do mau funcionamento do nosso sistema. A parte de alocação de memória também foi dificultosa, e é o motivo atual de nosso sistema não funcionar, todas as funcionalidades apresentam algum dos erros:

```
touch: cannot touch 'srq.txt': Numerical result out of range
```

e

```
mkdir: cannot create directory 'teste': Input/output error
```

Tentamos resolver de diversas maneiras, conversando com outros grupos e alterando lógicas, porém infelizmente não tivemos sucesso.
Como sugestão, apesar das dificuldades que tivemos, o trabalho é muito interessante, seria legal se a professora pudesse descorrer um pouco sobre os erros mais comuns enfrentados pelos grupos, mesmo que após o final do semestre, como um adicional para os alunos que tivessem a curiosidade e se identificassem mais com a disciplina, visto que a experiência de interagir com o sistema operacional dessa forma trouxe maior lucidez ao conteúdo.

5 USO DE MEMÓRIA E CPU DO SISTEMA IMPLEMENTADO

Apesar de o sistema não ter sido completamente implementado por nós, é possível fazer algumas suposições diante de alguns fatos: o sistema implementado apresenta número menor de funcionalidades, maior abstração de operações devido ao uso do software FUSE, as funcionalidades que são, de fato, implementadas por nós certamente não cobrem todos os erros e outputs possíveis.
Considerando esses fatores, é plausível supor que, devido à simplificação das operações, à abstração proporcionada pelo FUSE e à implementação focada em funcionalidades essenciais, o sistema FUSE possa oferecer um desempenho superior ao sistema padrão.

6 REFERÊNCIAS

[1] Hussain Q. Mohammed. **Writing a Simple Filesystem Using FUSE in C**. 2016. Disponível em: https://www.maastaar.net/fuse/linux/filesystem/c/
2016/05/21/writing-a-simple-filesystem-using-fuse. Acessado em: 22 de novembro de 2023.

[2] Fundamentos em Sistemas de Computação - Universidade Estadual Paulista. **Sistema de Arquivos**. Disponível em: https://www.dcce.ibilce.unesp.
br/ aleardo/cursos/fsc/cap11.php. Acessado em: 19 de novembro de 2023.

[3] Colby Colllege. **A Simple Makefile Tutorial**. Disponível em: https://www.cs.colby.edu
/maxwell/courses/tutorials/maketutor/. Acessado em: 20 de novembro de 2023.

[4] Shanshank Jian. **FileSystem in Userspace**. 2018. Disponível em: https://medium.com/@jain.sm/filesystem-in-userspace-5d1b398b04e. Acessado em: 19 de novembro de 2023

[5] Shanshank Jian. **File System in User space example**. 2018. Disponível em: https://medium.com/@jain.sm/file-system-in-user-space-example-a63a21236270. Acessado em: 20 de novembro de 2023

[6] Uehara Lincoln. **Introdução ao Makefile**. 2017. Disponível em: https://embarcados.com.br/introducao-ao-makefile/. Acessado em: 22 de novembro de 2023