Daniel Victor Silva Bonfim
Elder Henrique Alves Correia
Winicius Abilio de Britto
Yagho Junior Petini

Implementação de Sistema de Arquivos EXT2

Relatório técnico de atividade prática solicitado pelo professor Rodrigo Campiolo na disciplina de Sistemas Operacionais do Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Universidade Tecnológica Federal do Paraná – UTFPR
Departamento Acadêmico de Computação – DACOM
Bacharelado em Ciência da Computação – BCC

Campo Mourão
Julho / 2025

Resumo

Este relatório técnico apresenta o desenvolvimento de um shell interativo capaz de manipular diretamente imagens de sistemas de arquivos EXT2. O projeto foi implementado em linguagem C, sem o uso de bibliotecas externas para sistemas de arquivos, utilizando acesso direto aos dados binários da imagem. Foram implementadas funcionalidades para leitura, navegação e modificação da estrutura interna do EXT2, incluindo comandos como `ls`, `cd`, `mkdir`, `touch`, `rm`, `cp` e `rename`. As operações envolvem o controle de inodes, blocos, bitmaps e diretórios, bem como a manipulação de indireções simples e duplas em blocos. O resultado é um utilitário que simula comandos Unix diretamente sobre a imagem EXT2, proporcionando uma compreensão prática das estruturas de baixo nível dos sistemas de arquivos.

Palavras-chave: EXT2. sistemas de arquivos. shell interativo. manipulação de imagens. inodes. blocos.

Sumário

1 Introdução

Os sistemas de arquivos são componentes fundamentais em qualquer sistema operacional, pois são responsáveis pela organização, armazenamento e recuperação eficiente dos dados em dispositivos de armazenamento. Dentre as diversas opções disponíveis, o sistema EXT2 (Second Extended File System) destaca-se por sua simplicidade e robustez, sendo amplamente utilizado em ambientes Linux.

Este relatório apresenta o desenvolvimento de um shell interativo para manipulação direta de imagens do sistema de arquivos EXT2, utilizando a linguagem C. O objetivo principal do projeto foi proporcionar uma experiência prática no entendimento das estruturas internas de sistemas de arquivos, como superbloco, inodes, blocos de dados, diretórios e bitmaps, além de permitir a leitura e modificação dessas estruturas de forma controlada, sem depender de comandos externos ou bibliotecas de alto nível.

Por meio da implementação de comandos básicos como `ls`, `cd`, `mkdir`, `touch` e `rm`, foi possível explorar conceitos como superbloco, inodes, bitmaps e blocos de dados, além de compreender melhor a organização e os limites do EXT2.

Nas seções seguintes, detalharemos a descrição da atividade, os métodos empregados, os resultados obtidos, eventuais bugs conhecidos, a divisão das atividades entre os participantes, e as conclusões sobre o projeto.

2 Descrição da Atividade

A atividade consistiu na implementação de um shell interativo, em linguagem C, para manipulação de imagens de sistemas de arquivos EXT2 (`.img`). O shell funciona inteiramente sobre a imagem binária do sistema, realizando leitura e escrita direta nos blocos da imagem por meio de ponteiros de arquivos e cálculos de deslocamento, sem uso de chamadas externas ou bibliotecas específicas para EXT2.

comandos similares aos disponíveis em sistemas Unix, operando exclusivamente sobre os dados internos da imagem EXT2, a partir do diretório raiz (`/`).

As principais funcionalidades desenvolvidas foram:

- Leitura e interpretação de estruturas internas do EXT2: superbloco, descritores de grupo, tabela de inodes, bitmaps e diretórios.
- Implementação de comandos de leitura, sem alteração da imagem: `info`, `ls`, `pwd`, `cd`, `attr` e `cat`.
- Implementação de comandos de escrita, com alteração direta na imagem: `touch`, `mkdir`, `rm`, `rmdir`, `rename`.

- Implementação do comando `cp`, que realiza a cópia de arquivos da imagem EXT2 para o sistema de arquivos do host. O comando `mv` foi tratado como um alias para `rename`, dentro da imagem.
- Restrição ao uso de blocos de tamanho fixo (1024 bytes), tratamento apenas de diretórios com um único bloco de dados e suporte apenas a arquivos com tamanho máximo de 64 MiB.

Todas as operações foram realizadas diretamente sobre os dados binários da imagem, sem uso de bibliotecas externas para manipulação EXT2 e sem chamadas ao sistema como `system()` ou `exec()`.

3 Métodos

3.1 Estruturas de dados

Para representar as estruturas internas do sistema de arquivos EXT2, utilizou-se o arquivo `ext2_fs.h`, que define as principais estruturas, tais como o superbloco (`ext2_super_block`), descriptors de grupo, inodes (`ext2_inode`) e entradas de diretório (`ext2_dir_entry_2`). Estas estruturas refletem diretamente o layout do sistema EXT2 na imagem de disco.

3.2 Inicialização

A função `ext2_init()`, implementada em `ext2_lib.c`, é responsável por abrir a imagem do sistema de arquivos no modo leitura e escrita binário (`"r+b"`). Em seguida, realiza a leitura do superbloco e dos descriptors de grupo para carregar as informações essenciais ao funcionamento do sistema. O tamanho do bloco é calculado a partir do campo `s_log_block_size` do superbloco, conforme a fórmula: `block_size = 1024 << sb.s_log_block_size`.

3.3 Operações de entrada/saída em blocos e inodes

Foram implementadas funções específicas para leitura e escrita de blocos e inodes, que permitem o acesso direto à imagem do sistema de arquivos:

- `read_block()` e `write_block()` realizam, respectivamente, a leitura e escrita de blocos de dados na imagem.
- `get_inode()` e `write_inode()` permitem acesso e atualização das estruturas de inode.

- O gerenciamento de recursos livres no sistema é feito por meio das funções `alloc_inode()`, `free_inode_resource()`, `alloc_block()` e `free_block_resource()`, que manipulam os bitmaps de inodes e blocos, respectivamente.

3.4 Manipulação de diretórios

Para a navegação e modificação da estrutura de diretórios, destacam-se as seguintes funções:

- `search_directory()` e `find_inode_by_path()` que realizam a busca de entradas de diretório por nome e caminho completo.
- `add_dir_entry()` e `remove_dir_entry()` para inserção e remoção de entradas, com o devido ajuste do tamanho das entradas adjacentes por meio do campo `rec_len`.

3.5 Gerenciamento de Blocos Indiretos

Para lidar com arquivos de maior tamanho, foi implementado no projeto, suporte à liberação de blocos indiretos simples e duplos. A função `free_indirect_blocks()` trata os níveis de indireção recursivamente, lendo blocos de ponteiros e liberando os blocos referenciados. A função `free_all_blocks()`, por sua vez, é responsável por liberar todos os blocos associados a um inode, incluindo os 12 blocos diretos e os blocos indiretos de níveis 1 e 2.

3.6 Shell interativo

O shell, implementado no arquivo `ext2_shell.c`, roda em loop, lendo comandos do usuário, processando os argumentos e chamando as funções específicas definidas em `commands.c`. Ele mantém o estado atual, incluindo o inode e caminho de trabalho, permitindo comandos de navegação como `cd` e `pwd`. Comandos como `cp`, `touch`, `mkdir`, `rm`, `rmdir` e `rename` operam diretamente sobre a imagem do sistema de arquivos EXT2 por meio das funções implementadas.

4 Resultados e Discussão

4.1 Listagem Inicial do Diretório Raiz

Comando

```
ext2shell:[/] $ ls
```

Ação

Listar o conteúdo do diretório raiz (/).

Saída

```
inode: 1797277984
record length: 28526
name length: 119
file type: 32
```

Análise

O diretório raiz contém uma entrada de arquivo com um nome longo e dados incomuns. Isso pode ser um resquício de dados ou um arquivo de exemplo presente na imagem.

4.2 Criação de Diretório

Comando

```
ext2shell:[/] $ mkdir pasta
```

Ação

Criar um novo diretório chamado `pasta` no diretório atual (/).

Saída

```
Diretório 'pasta' criado.
```

Análise

O comando foi bem-sucedido. Um novo inode (nº 13) e um bloco de dados foram alocados para o diretório `pasta`.

4.3 Listagem Após Criação do Diretório

Comando

```
ext2shell:[/] $ ls
```

Saída

```
...
inode: 1797277984
...
pasta
inode: 13
record length: 28398
name length: 5
file type: 2
```

Análise

A listagem agora exibe a entrada para o novo diretório `pasta`, confirmando sua criação.

4.4 Navegação e Criação de Arquivo

Comandos

```
ext2shell:[/] $ cd pasta
ext2shell:[/pasta] $ touch arquivo
```

Ação

Entrar no diretório `pasta` e, em seguida, criar um arquivo vazio chamado `arquivo`.

Saída

```
Arquivo 'arquivo' criado.
```

Análise

A navegação para o diretório `pasta` foi bem-sucedida, como indicado pela mudança no prompt. O comando `touch` alocou um novo inode (nº 14) e adicionou uma entrada para `arquivo` dentro do diretório `pasta`.

4.5 Listagem do Conteúdo do Novo Diretório

Comando

```
ext2shell:[/pasta] $ ls
```

Saída

```
.
inode: 13
record length: 12
name length: 1
file type: 2

..
inode: 2
record length: 12
name length: 2
file type: 2

arquivo
inode: 14
record length: 1000
name length: 7
file type: 1
```

Análise

A listagem mostra as entradas padrão `.` (diretório atual, inode 13) e `..` (diretório pai, inode 2), além do `arquivo` recém-criado (inode 14).

4.6 Verificação do Caminho e Atributos

Comandos

```
ext2shell:[/pasta] $ pwd
ext2shell:[/pasta] $ attr arquivo
```

Saída

```
/pasta
Permissões UID    GID    Tamanho      Modificado em
drwxr-xr-x  0      0      1.0 KiB      06/07/2025 23:45
```

Análise

O comando `pwd` confirma a localização atual. O comando `attr` detalha que `arquivo` é um arquivo regular com 0 bytes e nenhum bloco de dados alocado, comportamento es-

perado de um arquivo criado com `touch`.

4.7 Remoção de Arquivo e Diretório

Comandos

```
ext2shell:[/pasta] $ rm arquivo
ext2shell:[/pasta] $ cd ..
ext2shell:[/] $ rmdir pasta
```

Saída

```
Arquivo 'arquivo' removido.
Diretório 'pasta' removido.
```

Análise

Primeiro, o arquivo `arquivo` foi removido, esvaziando o diretório `pasta`. Em seguida, após retornar ao diretório pai, o diretório `pasta` foi removido com sucesso. Ambas as estruturas (inode e blocos de dados) foram liberadas.

4.8 Criação e Renomeação de Arquivo

Comandos

```
ext2shell:[/] $ touch arquivo
ext2shell:[/] $ rename arquivo arquivoTeste
```

Saída

```
Arquivo 'arquivo' criado.
'arquivo' renomeado para 'arquivoTeste'.
```

Análise

Um novo arquivo foi criado (reutilizando o inode nº 13, que estava livre). A operação `rename` alterou apenas a entrada no diretório raiz para apontar para o mesmo inode, mas com um nome diferente.

4.9 Cópia de Arquivo (Imagem para Hospedeiro)

Comandos

```
ext2shell:[/] $ mkdir teste
ext2shell:[/] $ cp arquivoTeste ./teste
ext2shell:[/] $ cp arquivoTeste /home/usuario/Documentos/arquivoTeste2
```

Saída

```
Diretório 'teste' criado.
Arquivo 'arquivoTeste' copiado para './teste'.
Arquivo 'arquivoTeste' copiado para '/home/usuario/Documentos/arquivoTeste2'.
```

Análise

Um diretório `teste` foi criado na imagem. Em seguida, o arquivo `arquivoTeste` foi copiado da imagem para o sistema de arquivos hospedeiro em dois locais diferentes. É crucial notar que a cópia foi para o hospedeiro, não para o diretório `/teste` dentro da imagem, como confirmado pela seção seguinte.

4.10 Verificação e Exibição de Conteúdo

Comandos

```
ext2shell:[/] $ cd teste
ext2shell:[/teste] $ ls
ext2shell:[/teste] $ cd ..
ext2shell:[/] $ cat arquivoTeste
```

Saída da listagem

```
.
inode: 14
...
..
inode: 2
...
```

Análise

A listagem do diretório `/teste` na imagem mostra que ele está vazio, confirmando que a operação `cp` não copiou arquivos internamente. O comando `cat arquivoTeste` não

produz saída, pois o arquivo foi criado com `touch` e permanece com 0 bytes de tamanho.

5 Bugs Conhecidos

Na implementação, a leitura de blocos está restrita aos doze ponteiros diretos do inode, sem rotina para interpretar blocos simples indiretos, duplos ou triplos, o que torna inacessíveis arquivos maiores que sessenta e três kibibytes, mesmo quando alocados corretamente na imagem.

O comando mv atua apenas como um alias de rename, sem remover a entrada do diretório de origem nem inseri-la no diretório de destino; dessa forma, ao mover arquivos entre pastas, ocorre erro se o diretório alvo não existe ou o arquivo permanece na localização original.

A estrutura de diretórios foi projetada para ocupar apenas um bloco de um kibibyte, sem mecanismo de expansão ou fragmentação; quando o número de entradas excede a capacidade do bloco, novas inclusões podem corromper entradas existentes ou causar falhas nas operações de escrita.

6 Divisão das Atividades

O projeto foi realizado de forma colaborativa entre todos os integrantes do grupo. Ao longo do desenvolvimento, as atividades foram divididas conforme as afinidades e preferências de cada membro. Metade do grupo concentrou-se mais na implementação do código-fonte e testes do shell interativo, enquanto os demais focaram na organização e documentação do relatório técnico. Apesar dessa divisão inicial, o grupo trabalhou de forma integrada, trocando ideias, revisando etapas e garantindo que todas as partes do projeto estivessem alinhadas e funcionais.

7 Conclusões

Neste trabalho, foi implementado um interpretador de comandos em C capaz de manipular diretamente uma imagem de sistema de arquivos EXT2, sem depender de bibliotecas externas ou chamadas de sistema, mostrando domínio das estruturas internas do superbloco, grupos de blocos, bitmaps e tabela de inodes.

O projeto cumpriu com êxito as operações essenciais de leitura de metadados, navegação por diretórios, listagem de conteúdo, exibição de arquivos, criação, remoção e renomeação simples de arquivos e pastas, validando em todos os casos a consistência da árvore de diretórios e preservando corretamente os contadores de blocos e inodes livres.

Como pontos fortes, destacam-se a organização modular do código, com funções bem delimitadas para cada comando, o tratamento básico de erros que garante respostas claras ao usuário e a aderência fiel às especificações do EXT2 para imagens de até 64MiB com blocos de 1KiB.

Apesar do escopo alcançado, a solução ainda apresenta limitações que podem ser evoluídas sem comprometer sua solidez inicial. A leitura de blocos restringe-se aos doze ponteiros diretos do inode, deixando arquivos maiores que cerca de 63KiB inacessíveis; o comando mv funciona apenas como alias de rename, sem mover efetivamente entradas entre diretórios; e a estrutura de diretórios admite somente um bloco de 1KiB, sem expansão dinâmica, o que pode afetar pastas com grande número de entradas.

Como caminhos para aprimoramento, sugere-se estender o suporte a ponteiros indiretos simples e duplos para acessar arquivos maiores, implementar o mv completo removendo e inserindo entradas entre diretórios e adotar um esquema de fragmentação dinâmica de diretórios. Essas evoluções permitirão aumentar a robustez funcional do interpretador, mantendo sua modularidade e clareza originais, tornando-o ainda mais alinhado às necessidades de estudos e aplicações práticas em sistemas de arquivos EXT2.

8 Referências

LINUX FOUNDATION. *Ext4 – Linux Kernel Documentation*. [S.l.], 2024. Disponível em: https://docs.kernel.org/filesystems/ext4/index.html. Acesso em: 06 jul. 2025.