

Phitaia File System

Ana Julia V. P. Andrade, Gilberto A. A. Pessoa, Wanderson M. de Sousa

Departamento de Ciência da Computação
Universidade Federal de Roraima (UFRR) – Boa Vista, RR – Brazil

anajuliavpac@gmail.com, gilberto.terraria@gmail.com,
wandersonmoraidesousa@gmail.com

Abstract. *This work deals with the concepts and components of a file system, aiming at the creation of the "Phitaia" system. . To ensure correct operation on Linux, it is crucial to have elements: MBR, boot block, superblock, files, directories, FUSE and inodes. Tests were done, but the incompatibility of the code with the kernel made implementation difficult. Reports of other people experiencing similar issues were found. With the acquired knowledge, the way to be followed for the creation of the "Phitaia" system is understood.*

Resumo. *Este trabalho aborda sobre os conceitos e componentes de um sistema de arquivos, visando a criação do sistema "Phitaia". É feita uma definição de Sistemas de arquivos, falado sobre seu funcionamento e funcionalidades, e é respondida uma pergunta que explicará a motivação deste trabalho. Para garantir o correto funcionamento no Linux, é crucial contar com elementos: MBR, bloco de inicialização, superbloco, arquivos, diretórios, FUSE e inodes. Testes foram feitos, porém a incompatibilidade do código com o kernel dificultaram a implementação. Relatos de outras pessoas enfrentando problemas semelhantes foram encontrados. Com o conhecimento adquirido, compreende-se o caminho a ser seguido para a criação do sistema "Phitaia".*

1. Introdução

Este trabalho discute os fundamentos e componentes de um sistema de arquivos no Linux, com o objetivo de explorar o processo de criação do sistema de arquivos para a elaboração de um, chamado "Phitaia". O sistema de arquivos possui características como organização de dados, gerenciamento de espaço de armazenamento, controle de acesso e permissões, segurança, integridade dos dados e metadados. A criação de um sistema de arquivos ocorre durante a formatação do dispositivo de armazenamento, e os componentes incluem o MBR, bloco de inicialização, superbloco, arquivos, diretórios, Filesystem in Userspace (FUSE) e i-nodes. No geral, esses componentes trabalham juntos para garantir o funcionamento adequado e eficiente do sistema de arquivos no Linux. Foi feita a realização de testes no sistema de arquivos indicados pelo docente da disciplina de Sistema Operacionais. Esses testes foram feitos usando diferentes versões do Linux, mas as incompatibilidades com o kernel mais recente e a falta de suporte dificultaram o processo. Outras pessoas também enfrentaram problemas semelhantes, relatando dificuldades com o comando "mount". Embora tenha havido relatos de sucesso em versões mais recentes do kernel, os códigos utilizados não foram compartilhados. Devido à complexidade do projeto e à falta de atualização do código, a implementação do sistema de arquivos não foi bem-sucedida. Com o conhecimento

obtido foi possível compreender que caminho deve-se ser trilhado para a criação do “Phitaia”.

2. Sistema de arquivos

Um sistema de arquivos é idealizado para armazenar dados de forma não volátil, possuem características obrigatórias como um namespace (uma metodologia de organização), metadados (fornece a base lógica), API (fornece acesso a chamadas de sistemas que manipulam arquivos e diretórios), segurança (garante que usuários acessem somente seus arquivos) e um software (sistema de arquivos virtual do linux e os drivers de dispositivo específicos) para implementar o sistema de arquivos.

O Linux possui um sistema de arquivos virtual (VFS - Virtual File System) que fornece uma camada de abstração entre os sistemas de arquivos específicos e as chamadas de sistema feitas pelos aplicativos. Ele permite que diferentes sistemas de arquivos sejam suportados no Linux, como ext4, NTFS, FAT32, entre outros. O VFS também define uma série de estruturas e funções que permitem que os drivers de sistema de arquivos interajam com o kernel. Ele gerencia as operações comuns de sistema de arquivos, como leitura, gravação, exclusão, criação de diretórios e muito mais específicos correspondentes.

2.1 Funcionamento e Utilidade

Um sistema de arquivos fornece a estrutura e o conjunto de recursos que permitem aos usuários e ao sistema operacional gerenciar arquivos e diretórios com eficiência. Os recursos do sistema de arquivos são projetados para garantir a integridade dos dados, fornecer uma hierarquia lógica para organizar arquivos e fornecer funções para acessar e manipular dados armazenados. Algumas das principais características do sistema de arquivos incluem:

Organização de dados: organizar os dados em estruturas de dados hierárquicos contendo arquivos, pastas e diretórios. Podendo conter dentro de diretórios outros diretórios. Essa organização facilita o registro, localização e recuperação dos dados armazenados e permite que o sistema crie uma estrutura organizacional para que o sistema operacional possa fazer a leitura do ambiente.

Gerenciamento do espaço de armazenamento: rastrear quais áreas do dispositivo de armazenamento estão ocupadas e quais estão disponíveis para uso. Quando os arquivos são criados, modificados ou excluídos, o sistema aloca ou libera espaço no dispositivo de armazenamento conforme necessário.

Controle de acesso e permissões: gerenciar o acesso a dados e diretórios, permitindo que usuários e aplicativos leiam, gravem ou modifiquem dados de acordo com suas permissões.

Segurança: responsável por aplicar diferentes níveis de segurança para proteger os dados e garantir a privacidade, como criptografia.

Integridade dos dados e recuperação de erros: monitorar a integridade dos dados e fornecer mecanismos de recuperação de erros para garantir dados armazenados consistentes e confiáveis. faz isso por meio de técnicas como registro ou copy-on-write, que rastreiam alterações e permitem que versões anteriores sejam restauradas em caso de falha ou corrupção de dados.

Metadados: administrar os metadados, que são informações adicionais sobre os arquivos e diretórios, como datas de criação e modificação, tamanho, permissões de acesso e propriedades.

2.2 Qual a diferença do Sistema de arquivos padrão do linux para o sistema de arquivos implementado?

A diferença entre o sistema de arquivos padrão do Linux e um sistema de arquivos implementado como um sistema de brinquedo reside principalmente na complexidade, recursos e finalidade.

O sistema de arquivos padrão do Linux, como o Ext4, é um sistema de arquivos maduro, estável e altamente otimizado. Ele foi desenvolvido e aprimorado ao longo de muitos anos para atender às necessidades gerais de um sistema operacional moderno. Ele suporta recursos avançados, como journaling (registro de transações), alocação eficiente de espaço em disco, controle de acesso baseado em permissões e integridade de dados.

Um sistema de arquivos implementado como um sistema de brinquedo é projetado com fins educacionais, de aprendizado ou experimentação. Esses sistemas de arquivos geralmente são simplificados e não possuem todos os recursos e otimizações encontrados nos sistemas de arquivos padrão. Eles podem ser criados como projetos pessoais ou acadêmicos para entender os conceitos básicos de um sistema de arquivos, como estrutura de diretórios, alocação de espaço em disco, manipulação de arquivos e metadados.

Além disso, o sistema de arquivos padrão do Linux é amplamente testado, utilizado é suportado pela comunidade e pela indústria. Ele é desenvolvido por especialistas em sistemas de arquivos e passa por um processo rigoroso de revisão e atualização contínua para garantir sua estabilidade e confiabilidade. Já um sistema de arquivos implementado como um sistema de brinquedo pode não ter a mesma robustez e confiabilidade devido ao seu propósito mais limitado e à falta de recursos e testes extensivos.

Porém, desenvolvedores que preenchem seu código com implementações `ioctl()` ou `/proc`s arquivos geralmente são incentivados a criar um sistema de arquivos virtual autônomo. Os sistemas de arquivos tornam a interface explícita e visível no espaço do usuário; eles também facilitam a escrita de scripts que executam funções administrativas. Mas escrever um sistema de arquivos Linux pode ser uma tarefa intimidante. Um desenvolvedor que passou algum tempo apenas se familiarizando com a interface do driver pode ser perdoado por se recusar a aprender a API do VFS também.

Ao implementar um sistema de arquivos, é possível adaptar o sistema às necessidades específicas, como otimização para determinado tipo de aplicação, suporte a recursos avançados, melhor desempenho, escalabilidade ou qualquer outro requisito específico.

3. Criação de um Sistema de arquivos Linux

A criação de um sistema de arquivos é um processo fundamental para preparar um dispositivo de armazenamento para o armazenamento e gerenciamento eficiente dos dados. Durante esse processo, são estabelecidas estruturas de dados e configurações que permitirão ao sistema operacional organizar, acessar e manipular os arquivos e diretórios presentes no dispositivo.

O processo de criação de um sistema de arquivos geralmente ocorre durante a formatação do dispositivo de armazenamento. A formatação pode ser realizada por meio de utilitários fornecidos pelo sistema operacional ou por meio de ferramentas específicas de formatação. Vamos explorar os principais passos envolvidos na criação de um sistema de arquivos.

Para garantir o correto funcionamento de um sistema de arquivos no Linux, é crucial contar com elementos essenciais que desempenham papéis fundamentais. A seguir, será apresentada uma descrição mais detalhada dos itens necessários para esse propósito.

4. Componentes de um Sistema de arquivos

Um sistema de arquivos é um conjunto de estruturas e componentes que permitem a organização, armazenamento e gerenciamento dos dados em um dispositivo de armazenamento, como um disco rígido, SSD, pen drive ou cartão de memória. Ele fornece uma estrutura lógica para a organização dos arquivos e diretórios, além de oferecer recursos para acessar, manipular e proteger os dados armazenados. Os componentes de um sistema de arquivos desempenham papéis fundamentais no gerenciamento eficiente e confiável dos dados. Alguns dos principais componentes incluem:

4.1 MBR

O MBR (Master Boot Record) é uma abreviação para Registro Mestre de Inicialização que é uma parte essencial de um disco de armazenamento que contém as informações de inicialização e a tabela de partição. O mesmo está localizado no primeiro setor (o setor 0) de um disco rígido ou de uma unidade de armazenamento, ele contém um pequeno programa de inicialização chamado de código de bootstrap, juntamente com a tabela de partição. A tabela de partição lista as informações sobre as partições presentes no disco, como seus tamanhos, tipos de sistema de arquivos e outros detalhes relevantes. O MBR também inclui um pequeno espaço reservado para a assinatura de inicialização, que é um valor de verificação de 2 bytes, essa assinatura é usada pelo processo de inicialização para verificar se o MBR é válido e pode ser executado.

No contexto do sistema de arquivos Linux, o MBR é usado principalmente para sistemas de arquivos baseados em BIOS (Basic Input/Output System), ele é frequentemente usado em discos rígidos tradicionais (HDDs) e em algumas unidades de estado sólido (SSDs) que usam a interface de disco MBR.

4.2 Bloco de inicialização

Um bloco de inicialização é uma unidade lógica de código ou script executado durante o processo de inicialização de um sistema operacional. Esses blocos de inicialização são responsáveis por executar várias tarefas durante o processo de inicialização, como configuração de hardware, inicialização de serviços e execução de scripts personalizados.

Ele está localizado no primeiro setor de um disco ou partição formatada com um sistema de arquivos específico, como FAT (File Allocation Table) ou NTFS (New Technology File System) no Windows ou ext4 no Linux. Algumas informações comuns no bloco de inicialização incluem:

1. Código de inicialização (bootstrap code): É um pequeno programa executável armazenado no bloco de inicialização. Esse código é carregado na memória durante a inicialização do sistema operacional e é responsável por iniciar o sistema de arquivos.
2. Informações de inicialização: O bloco de inicialização pode conter informações sobre o sistema de arquivos, como o tipo de sistema de arquivos utilizado, parâmetros de configuração e outras informações relevantes para a inicialização do sistema de arquivos.
3. Tabela de partição: Em alguns sistemas de arquivos, especialmente aqueles que suportam partições, o bloco de inicialização pode conter uma tabela de partição. Essa tabela descreve as partições presentes no disco, incluindo informações sobre o tamanho, tipo e localização de cada partição.

4.3 Superbloco

Outro bloco essencial é o Super Bloco que trata-se de uma estrutura de dados importante que armazena informações fundamentais sobre o sistema de arquivos, cada sistema de arquivos linux possui um super bloco, que é uma área reservada no início do dispositivo de armazenamento como um disco rígido ou uma partição. O super bloco é lido pelo sistema operacional durante a inicialização do sistema de arquivos para obter informações críticas sobre sua estrutura e localização dos metadados. Essas informações são usadas para permitir o acesso e a manipulação correta dos arquivos e diretórios no sistema de arquivos. São elas: o tipo de sistema de arquivos, tamanho de bloco, contagem de blocos e inodes, além de ponteiros para os blocos de dados relacionados ao sistema de arquivos. O sistema é dividido em blocos, vale ressaltar que o tamanho do bloco afeta a eficiência do armazenamento e operações de leitura/gravação.

Cada sistema de arquivos possui seu próprio formato de super bloco, com campos específicos para armazenar as informações mencionadas acima. O formato e a localização exata do super bloco podem variar de acordo com o tipo de sistema de arquivos usado.

4.4 Arquivos e diretórios

Um arquivo é uma unidade básica de armazenamento no Linux. Ele contém dados, como texto, código executável, imagens, vídeos, etc. os arquivos podem ser acessados, lidos, gravados e executados, dependendo das permissões definidas.

Um diretório, também conhecido como pasta, é uma estrutura que armazena arquivos e outros diretórios. Ele serve como um contêiner para organizar e agrupar arquivos relacionados. Os diretórios também podem conter outros diretórios, permitindo a criação de uma hierarquia de diretórios. A estrutura de diretórios no Linux é organizada em uma árvore, com o diretório-raiz (root directory) que é o diretório de nível mais alto em uma estrutura hierárquica de diretórios. O diretório-raiz é a base de toda a estrutura do sistema de arquivos Linux e contém todos os outros diretórios e arquivos. Todos os caminhos absolutos no sistema de arquivos começam a partir do diretório-raiz.

A manipulação é efetuada pelo gerenciamento de arquivos que realiza as tarefas no sistema operacional Linux, como armazenar e recuperar dados, instalar e executar aplicativos, gerenciar permissões de acesso, entre outras atividades relacionadas ao gerenciamento de arquivos e diretórios.

4.5 Filesystem in Userspace

A biblioteca "libfuse" (Filesystem in Userspace) é uma biblioteca amplamente difundida para a criação de sistemas de arquivos em espaço de usuário no Linux, essa biblioteca permite que os desenvolvedores criem sistemas de arquivos personalizados

que podem ser montados e acessados como qualquer outro sistema de arquivos no sistema operacional. Ele oferece uma interface de programação de aplicativos que permite que os desenvolvedores implementem suas próprias operações de sistema de arquivos.

Ao usá-la, os sistemas de arquivos podem ser implementados em nível de usuário, sem a necessidade de modificar o kernel do Linux. Isso torna o desenvolvimento de sistemas de arquivos mais flexível, pois não requer privilégios de superusuário para montar e usar os sistemas de arquivos.

A biblioteca fornece as funcionalidades necessárias para lidar com a comunicação entre o kernel e o sistema de arquivos em espaço de usuário. Ela gerencia as chamadas de sistema relacionadas a arquivos, diretórios e metadados, permitindo que os sistemas de arquivos em espaço de usuário se comportem de maneira semelhante aos sistemas de arquivos tradicionais. Através da libfuse, os desenvolvedores podem implementar sistemas de arquivos com diferentes funcionalidades e comportamentos, criando soluções personalizadas para necessidades específicas. Essa biblioteca facilita a criação de sistemas de arquivos virtuais, sistemas de arquivos criptografados, sistemas de arquivos em rede e muitos outros tipos de sistemas de arquivos personalizados.

4.6 I-nodes

Os "i-nodes"(índices de nós) são uma estrutura muito importante no sistema de arquivos do Linux, eles são usados para representar metadados de arquivos e diretórios, como: permissões, propriedade, tamanho, data de criação e modificação, entre outros.

Cada arquivo e diretório em um sistema de arquivos Linux é associado a um i-node único que fornece uma maneira eficiente de organizar e acessar as informações sobre os arquivos em um sistema de arquivos. Eles são gerenciados pelo kernel do Linux e são invisíveis para a maioria dos usuários. É importante mencionar que cada sistema de arquivos tem um número limitado de i-nodes disponíveis, e esse limite é definido durante a criação do sistema de arquivos. Portanto, em sistemas com muitos arquivos pequenos, é possível atingir o limite de i-nodes antes de atingir o limite de espaço em disco.

5. Implementação, testes e erros

O Sistema de arquivos que foi alvo de tentativas de implementação foi o indicado no link a seguir: <https://www.ic.unicamp.br/~islene/2s2013-mo806/vfs/gogislenefs/> . Esse sistema de arquivos foi feito na linguagem C em 2013, foi testado na versão kernel 5.19, GNU Make 4.2, GCC 11.3 no Ubuntu 22.04 LTS "Jammy Jellyfish". Além desse sistema foi feita a tentativa de implementar outros códigos, mas também não foi obtido sucesso nessas tentativas. A seguir veja alguns dos passos feitos para a tentativa de implementação.

5.1 Códigos usados no terminal

Foi usada uma sequência de códigos no terminal do Linux, que possui o Kernel de versão 5.19. Dentro dos arquivos foram colocados os códigos em linguagem C.

Veja abaixo os códigos:

```
sudo apt remove --purge linux-headers-* (Remove os pacotes de cabeçalhos do kernel instalados no sistema. Os pacotes de cabeçalhos são necessários para compilar módulos do kernel.)
```

```
sudo apt autoremove && sudo apt autoclean (Executa uma limpeza do sistema, removendo pacotes desnecessários e limpando arquivos temporários.)
```

```
sudo apt install linux-headers-generic (Instala os pacotes de cabeçalhos do kernel genérico, necessários para compilar módulos do kernel.)
```

```
$ sudo apt install git build-essential kernel-package fakeroot libncurses5-dev libssl-dev ccache (Instala um conjunto de pacotes e ferramentas de desenvolvimento necessários para compilar o kernel e módulos.)
```

```
$ apt-get install build-essential linux-headers-`uname -r` (instala os pacotes essenciais de compilação e os cabeçalhos do kernel necessários para compilar programas no sistema operacional com base na versão do kernel atual)
```

```
$ mkdir fs (Cria um diretório chamado "fs" no diretório atual.)
```

```
$ cd fs (Navega para o diretório "fs".)
```

```
$ nano Makefile (Abre o arquivo "Makefile" no editor de texto "nano" para editar as configurações de compilação do módulo.)
```

```
$ nano islene.h (Abre os arquivos "islene.h")
```

```
$ nano file.c (Abre os arquivos "file.c")
```

```
$ nano inode.c (Abre os arquivos "inode.c")
```

```
$ nano namei.c (Abre os arquivos "namei.c")
```

```
$ make (Executa o comando "make", que compila o código-fonte do módulo e gera o arquivo binário "islene.ko".)
```

```
$ dd if=/dev/zero of=rep bs=1k count=4 (Esse comando usa o utilitário dd para criar um arquivo chamado "rep" com tamanho de 4 kilobytes. Ele preenche o arquivo com zeros, pois o arquivo de origem (if) é "/dev/zero", que é uma fonte infinita de bytes nulos.)
```

```
$ mkdir -p mnt (Cria um diretório chamado "mnt" no diretório atual. O uso da opção -p garante que o comando crie o diretório pai se ele ainda não existir.)
```


`$ mount -t islene -o loop rep mnt` (Esse comando monta o arquivo "rep" como um sistema de arquivos no diretório "mnt". A opção "-t islene" indica o tipo de sistema de arquivos a ser usado, que é "islene" neste caso. A opção -o loop permite que o arquivo seja tratado como um dispositivo de loopback. O arquivo "rep" é montado no diretório "mnt" para que seu conteúdo seja acessível.)

`$ insmod islene.ko` (Carrega o módulo do kernel "islene.ko" usando o comando "insmod". Esse comando insere o módulo no kernel, permitindo que ele seja executado e utilizado pelo sistema operacional.)

`$ cat counter0` (O comando cat é usado para exibir o conteúdo de um arquivo no terminal. Nesse caso, ele exibe o conteúdo do arquivo "counter0" no diretório "mnt".)

5.2 Testes e resultados alcançados

Foram feitos diversos testes com códigos diferentes para tentar implementar o sistema de arquivos, porém nenhuma das tentativas foram concluídas com sucesso. Os testes foram feitos no Linux que estava instalado no sistema por meio de dual boot e também foram feitos em diferentes versões da distro Linux pela VirtualBox, inclusive em versões diferentes.

Um dos problemas analisados foi na construção do sistema de arquivos, seu código(<https://www.ic.unicamp.br/~islene/2s2013-mo806/vfs/gogislenefs/>) é de 2013 e foi feito e pensado para o kernel 3.1.x. Ao tentar baixar uma versão antiga do linux pela virtualbox, foi percebida a falta de suporte, não era compatível usar o firefox para acessar os sites e conseguir copiar e colar o código. Foi feita também a tentativa no código (<https://lwn.net/Articles/13379/>) feito por Jonathan Corbet em 2002. Inclusive em um fórum(<https://lwn.net/Articles/13325/>) a respeito da criação de um sistema de arquivos é possível ver que outras pessoas também tiveram complicação e não conseguiram implementar completamente. Algumas pessoas relataram que tiveram problema na hora de usar o comando "mount", que foi o mesmo caso enfrentado nas tentativas feitas neste trabalho.

Um dos participantes da discussão relatou ter conseguido implantar na versão do kernel 4.2 e indicou um link que levava para o github, onde apresentava o código(<https://gist.github.com/RadNi/9d8a074e6264c1664b97b8eee11b1d2a>). Nos comentários do sistema de arquivo compartilhado pelo "RadNi" ou Amirhossein Khajehpour, como indicado no github, é possível ver o comentário de Charlie-Green que relatou ter conseguido implementar o sistema de arquivos no kernel 5.0, porém apesar dele dizer que postaria seu código que foi baseado no do "RadNi", ele não postou. Portanto entendemos que para executar um sistema de arquivos que foi feito a algum tempo atrás é preciso fazer adaptações no código para assim obter sucesso. Por ser algo um pouco complexo este projeto acabou não contemplando a atualização do código e por tanto não obteve sucesso na implementação.



Figura 1. Imagem do Fórum relatos de funcionamento.

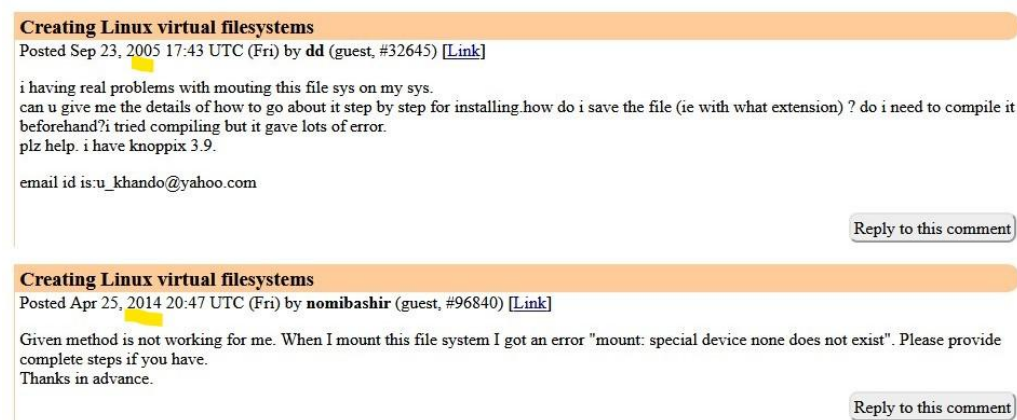


Figura 2. Imagem do Fórum com relatos de erros.

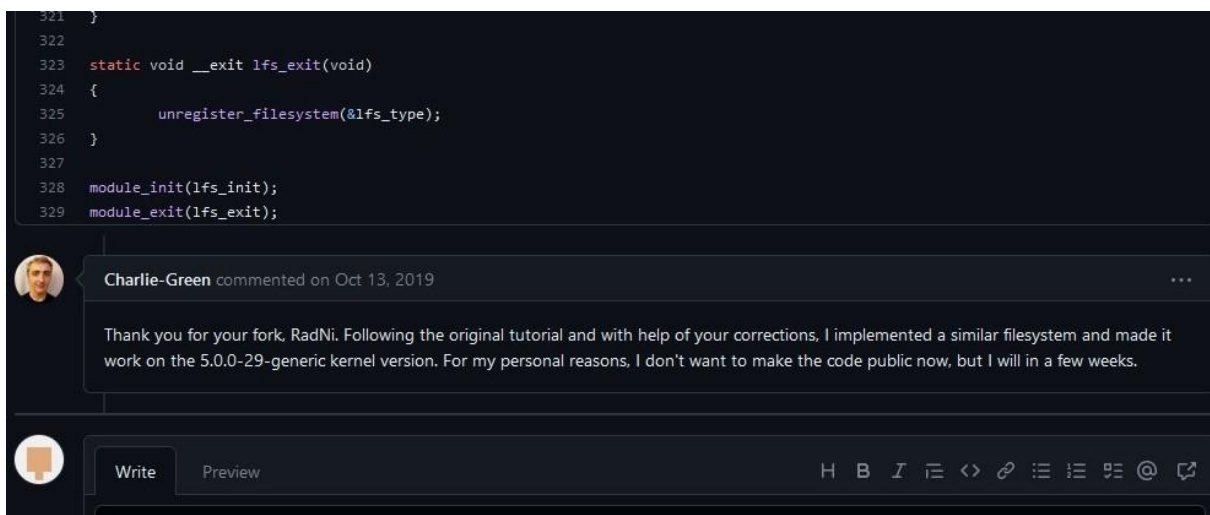
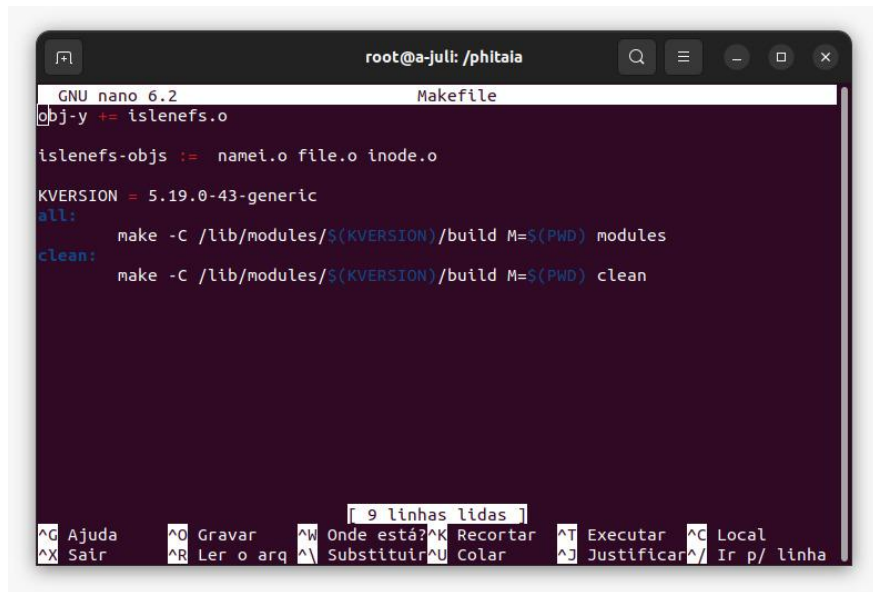


Figura 3. Imagem do GitHub com relato de funcionamento no kernel 5.0

5.3 Imagens da implementação



A terminal window titled 'root@a-juli: /phitaia' showing the GNU nano 6.2 editor editing a file named 'Makefile'. The content of the Makefile is as follows:

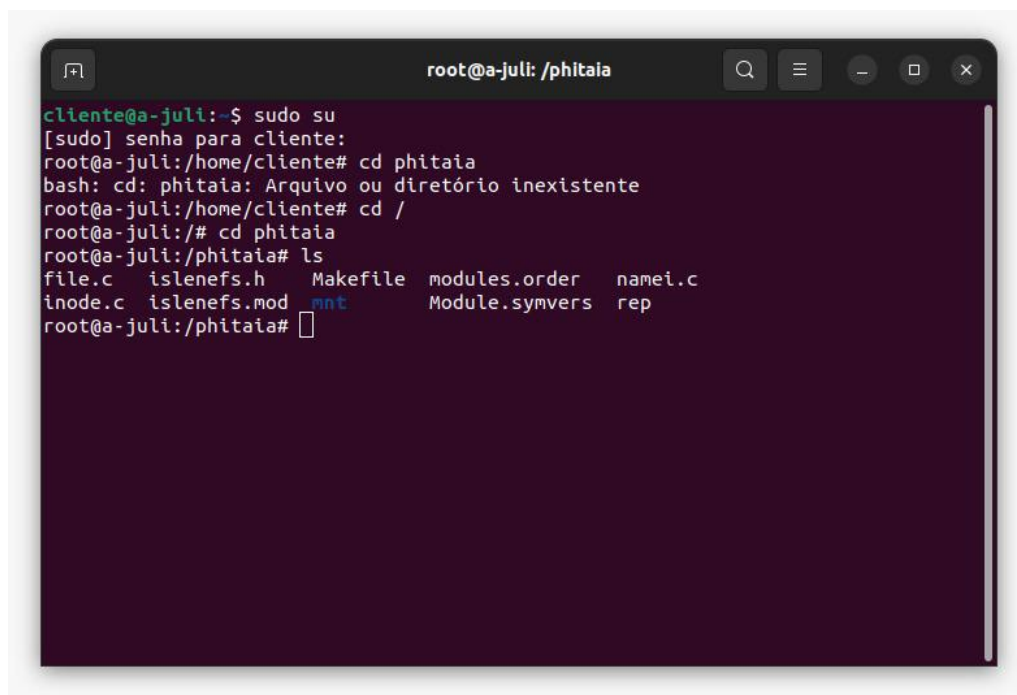
```
obj-y += islenefs.o

islenefs-objs := namei.o file.o inode.o

KVERSION = 5.19.0-43-generic
all:
    make -C /lib/modules/$(KVERSION)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(KVERSION)/build M=$(PWD) clean
```

The bottom status bar of the nano editor shows '9 linhas lidas' and various keyboard shortcuts like '^G Ajuda', '^O Gravar', '^W Onde está?', etc.

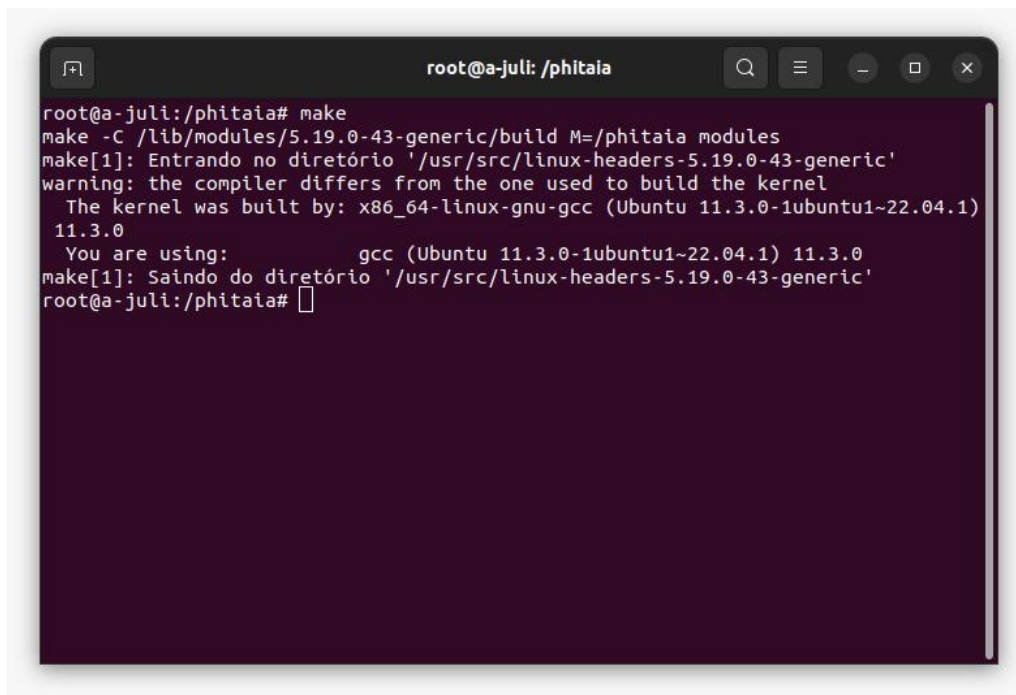
Figura 4. Imagem do arquivo “Makefile” no terminal



A terminal window titled 'root@a-juli: /phitaia' showing a sequence of commands and their outputs:

```
cliente@a-juli:~$ sudo su
[sudo] senha para cliente:
root@a-juli:/home/cliente# cd phitaia
bash: cd: phitaia: Arquivo ou diretório inexistente
root@a-juli:/home/cliente# cd /
root@a-juli:/# cd phitaia
root@a-juli:/phitaia# ls
file.c  islenefs.h  Makefile  modules.order  namei.c
inode.c islenefs.mod mnt       Module.symvers rep
root@a-juli:/phitaia#
```

Figura 5. Imagem dos repositórios e arquivos criados e gerados

A terminal window titled 'root@a-juli: /phitaia' with standard window controls. The terminal output shows the execution of 'make' in the directory '/lib/modules/5.19.0-43-generic/build M=/phitaia modules'. It displays a warning about the compiler and the kernel version, and then shows the directory being exited. The prompt returns to 'root@a-juli: /phitaia#'.

```
root@a-juli:/phitaia# make
make -C /lib/modules/5.19.0-43-generic/build M=/phitaia modules
make[1]: Entrando no diretório '/usr/src/linux-headers-5.19.0-43-generic'
warning: the compiler differs from the one used to build the kernel
The kernel was built by: x86_64-linux-gnu-gcc (Ubuntu 11.3.0-1ubuntu1~22.04.1)
11.3.0
You are using: gcc (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0
make[1]: Saindo do diretório '/usr/src/linux-headers-5.19.0-43-generic'
root@a-juli:/phitaia#
```

Figura 6. Imagem do terminal após executar o comando “make”

6. Conclusão

Este trabalho aborda conceitos, funcionalidade, construção e implementação de sistemas de arquivos em um ambiente Linux. Um sistema de arquivos é uma estrutura para armazenamento não volátil de dados com recursos obrigatórios, como namespaces, metadados, APIs, segurança e o software que implementa o sistema. O Linux usa um sistema de arquivos virtual (VFS), que permite suportar diferentes sistemas de arquivos, como ext4, NTFS e FAT32. O VFS fornece uma camada de abstração entre um sistema de arquivos específico e as chamadas de sistema feitas pelos aplicativos.

Um sistema de arquivos fornece a capacidade de organizar dados em hierarquias, gerenciar espaço de armazenamento, controlar acesso e permissões, garantir a segurança dos dados, manter a integridade e recuperar-se de erros. Ele também gerencia metadados, que são informações adicionais sobre arquivos e diretórios.

No Linux, a implementação de um sistema de arquivos pode ser personalizada de acordo com necessidades específicas. Os componentes essenciais de um sistema de arquivos incluem o MBR que contém informações de inicialização e a tabela de partição. O bloco de inicialização é executado durante a inicialização do sistema operacional e contém código de inicialização e informações sobre o sistema de arquivos. O super bloco armazena informações críticas sobre a estrutura e localização dos metadados do sistema de arquivos. Os arquivos e diretórios são as unidades básicas de armazenamento e organização de dados. A biblioteca libfuse permite criar sistemas de arquivos personalizados em espaço de usuário no Linux. Os inodes são estruturas usadas para representar metadados de arquivos e diretórios.

Foram feitas tentativas de implementação de um sistema de arquivos baseado em um código feito em 2013. O código foi desenvolvido em C em 2013 e testado no kernel 5.19, GNU Make 4.2 e GCC 11.3 no Ubuntu 22.04 LTS "Jammy Jellyfish". Foram realizados testes em diferentes versões do Linux através do dual boot e da VirtualBox, porém nenhum dos testes foi concluído com sucesso. Houve dificuldades devido à falta de suporte para versões antigas do Linux e problemas com o comando "mount". Apesar de algumas pessoas terem relatado sucesso na implementação em versões específicas do kernel, nenhum código adicional foi disponibilizado. Concluiu-se que adaptações seriam necessárias para obter sucesso na implementação, mas devido à complexidade do projeto, não foi possível realizar as atualizações e alcançar o objetivo.

No geral, foi compreendido os conceitos para a construção de um sistema de arquivos e agora é possível identificar possíveis vulnerabilidades e erros na hora de implementar um código de um sistema de arquivos no terminal Linux. Com os conhecimentos obtidos facilitar para a próxima tentativa de implementar o Sistema de arquivos "Phitaia".

7. Referências

- Costa, L. (2010) “ Entendendo MBR e sistema de arquivos GNU/Linux”, <https://www.vivaolinux.com.br/artigo/Entendendo-MBR-e-sistema-de-arquivos-GN> U-Linux, 28 de junho de 2023.
- Net, C. ”O que é e para que serve o sistema de arquivos (File System)?”, <https://www.controle.net/faq/o-que-e-sistema-de-arquivos-file-system>, 28 de junho de 2023.
- Corbet, J. (2003) “Creating Linux virtual filesystems ”, <https://lwn.net/Articles/57369/>, 29 de junho de 2023.
- Weller, A.;_Silva, F.; Silva, F. J.; Krisman, G.;_Santiago, T.;_Sacramento, R. e Silva, K. (2013) “MO806/MC914 - Tópicos em Sistemas Operacionais”, <https://www.ic.unicamp.br/~islene/2s2013-mo806/>, 28 de junho de 2023.
- Weller, A.;_Silva, F.; Silva, F. J.; Krisman, G.;_Santiago, T.;_Sacramento, R. e Silva, K. (2013) “Index of /~islene/2s2013-mo806/vfs/gogislenefs”, <https://www.ic.unicamp.br/~islene/2s2013-mo806/vfs/gogislenefs/> , 28 de junho de 2023.
- Weller, A.;_Silva, F.; Silva, F. J.; Krisman, G.;_Santiago, T.;_Sacramento, R. e Silva, K.(2013) “Index of /~islene/2s2013-mo806/vfs/hellofs”, <https://www.ic.unicamp.br/~islene/2s2013-mo806/vfs/hellofs/>, 28 de junho de 2023.