УДК 004.4 http://doi.org/10.15407/pp2024.02-03.334

С.В. Поперешняк, С.В. Панченко, О. Д. Федорченко, С. А. Ільїн

ОСОБЛИВОСТІ ОБРОБКИ ТА ЗБЕРЕЖЕННЯ ДАНИХ ЗА ДОПОМОГОЮ ВІРТУАЛЬНОЇ ФАЙЛОВОЇ СИСТЕМИ

В роботі було розроблено та розглянуто програмну систему, спрямовану на створення віртуальної файлової системи (ВФС) на основі FUSE, яка надає зручний та гнучкий інтерфейс для взаємодії з файловою структурою. Основною метою цього проєкту є пришвидшення обробки файлів за допомогою кастомізації операцій файлової системи та створення інструменту, який дозволяє користувачам легко маніпулювати віртуальною файловою структурою, використовуючи консольний інтерфейс. Розробка призначена для обробки та збереження даних на операційних системах сімейства Unix. Функціональні завдання програмної системи сфокусовані на створенні віртуальної файлової системи (ВФС) на основі FUSE з метою надання користувачам зручного інтерфейсу для взаємодії з файловою структурою. Головною метою є забезпечення функціональних можливостей для створення та видалення об'єктів, отримання детальної інформації про атрибути файлів, читання та запису вмісту файлів, роботи із символьними посиланнями, а також навігації та взаємодії з директоріями. Це включає в себе можливість створення нових файлів та директорій, видалення об'єктів, отримання детальної інформації про атрибути файлів, читання та редагування їх вмісту, а також роботу із символьними посиланнями. Подальшою метою є створення зручного та функціонального інструменту для користувачів, який може бути використаний для проведення експериментів, навчання або вирішення конкретних завдань, забезпечуючи гнучкість та ефективність взаємодії з файловою системою на основі FUSE через консольний інтерфейс.

Ключові слова: віртуальна файлова система, UNIX, FUSE, ядро ОС, файл, директорія, soft-посилання, статичний поліморфізм, гібридні системи, живучість, бази даних, багатокритеріальна оптимізація, центр обробки даних

S.Popereshnyak, S. Panchenko, O. Fedorchenko, S. Ilyin

FEATURES OF DATA PROCESSING AND STORAGE USING THE VIRTUAL FILE SYSTEM

The work developed and considered a software system aimed at creating a virtual file system (VFS) based on FUSE, which provides a convenient and flexible interface for interacting with the file structure. The main goal of this project is to speed up file processing by customizing file system operations and creating a tool that allows users to easily manipulate the virtual file structure using a console interface. The development is intended for data processing and storage on operating systems of the Unix family. The functional tasks of the software system are focused on creating a virtual file system (VFS) based on FUSE with the aim of providing users with a convenient interface for interacting with the file structure. The main goal is to provide functionality for creating and deleting objects, obtaining detailed information about file attributes, reading and writing file contents, working with symbolic links, and navigating and interacting with directories. This includes the ability to create new files and directories, delete objects, get detailed information about file attributes, read and edit their contents, and work with symbolic links. The further goal is to create a convenient and functional tool for users, which can be used for conducting experiments, training or solving specific tasks, while providing flexibility and efficiency of interaction with the FUSE-based file system through the console interface.

Key words: virtual file system, UNIX, FUSE, OS kernel, file, directory, soft-link, static polymorphism, hybrid systems, survivability, databases, multicriteria optimization, data center.

Вступ

З урахуванням швидкого розвитку сучасних технологій та зростання обсягів даних, виникає необхідність вдосконалення існуючих віртуальних файлових систем або розроблення нових, які відповідають сучасним вимогам ефективності, безпеки та гнучкості. Актуальність написання нової файлової системи полягає у здатності відпові-

сти на виклики, пов'язані з розширеним обсягом даних, розподіленими обчисленнями, вимогами до конфіденційності та високою швидкодією обробки інформації.

Після розгляду успішних аналогів важливо відзначити, що багато з них характеризуються закритим кодом чи вимагають від розробників власної реалізації операцій, що часто може обмежувати гнучкість та розширюваність. У цьому контексті власнорозроблена файлова система, заснована на FUSE бібліотеці, виділяється серед аналогів, пропонуючи низку вагомих переваг. Вона не лише гнучка завдяки стандартному інтерфейсу FUSE, а й має відкритий код, що сприяє активній участі розробників і прискорює виявлення та виправлення помилок. Крім того, забезпечення конфіденційності, висока швидкодія, цілісність даних у багатопотоковому середовищі, сумісність з Unix-подібними системами та можливість розширення функціональності за допомогою кастомізації операцій роблять її важливим рішенням для сучасних вимог до файлових систем.

Мета роботи – є пришвидшення обробки файлів за допомогою кастомізації операцій файлової системи та створення інструменту, який дозволяє користувачам легко маніпулювати віртуальною файловою структурою, використовуючи консольний інтерфейс.

У даній роботі буде розглянуто концепцію віртуальних файлових систем, проведений аналіз існуючих рішень, та обґрунтована необхідність розробки нової файлової системи, яка відповідає вимогам сучасності та враховує актуальні тенденції в області обробки та збереження даних.

Віртуальна файлова система та галузі практичного застосування

Файлові системи пропонують загальний інтерфейс для програм для доступу до даних. Хоча мікроядра реалізують файлові системи в просторі користувача [1], більшість файлових систем є частиною монолітних ядер [2]. Реалізації ядра дозволяють уникнути великих накладних витрат на передачу повідомлень мікроядер і демонов простору користувача [3]. Проте в останні роки популярність файлових систем простору користувача зросла з чотирьох причин.

(1) Кілька файлових систем, які можна стекувати, додають спеціалізовані функції до існуючих файлових систем (наприклад, дедуплікацію та стиснення [4]).

(2) В академічному середовищі та науково-дослідних установах ця структура дозволила швидко експериментувати та створювати прототипи нових підходів [5, 6].

(3) Кілька існуючих файлових систем на рівні ядра було перенесено в простір користувача (наприклад, ZFS [7], NTFS [8]).

(4) Більше компаній покладаються на впровадження простору користувача: GPFS і LTFS від IBM, CASL від Nimble Storage, HDFS від Apache, файлова система Google, GlusterFS від RedHat, Data DDFS домену [9] тощо.

Збільшення складності файлових систем є фактором, що сприяє зростанню популярності файлових систем у просторі користувача. Код простору користувача легше розробляти, переносити та підтримувати. Помилки ядра можуть вивести з ладу цілі системи, тоді як помилки в просторі користувача є більш обмеженими. Багато бібліотек і мов програмування доступні в просторі користувача на різних платформах.

Віртуальна файлова система (ВФС)— це програмна або апаратна конструкція, яка надає інтерфейс для роботи з файловою системою. Це може бути шар абстракції між програмним забезпеченням та фізичними носіями даних, який дозволяє звертатися до файлів і папок, незалежно від їхнього розташування чи форматування зберігання.

ВФС може забезпечувати деякі додаткові функції, такі як шифрування, контроль доступу, кешування, компресія, відображення віддалених ресурсів тощо. Вона дозволяє програмам взаємодіяти з файловою системою, не враховуючи конкретні деталі роботи фізичних пристроїв чи мережевих ресурсів.

Основні переваги віртуальних файлових систем:

- абстракція від апаратних рішень;
- управління різноманітністю джерел даних;
- додаткові функціональні можливості;
- підтримка віддалених ресурсів;
- зручність розробки та тестування.

Віртуальна файлова система перехоплює системні виклики, пов'язані з операціями файлової системи. Ці виклики генеруються операційною системою або програмами, що намагаються взаємодіяти з файловою системою. ВФС взаємодіє з ядром операційної системи для обробки системних викликів і отримання доступу до ресурсів. Вона реєструється у ядрі як обробник файлових операцій та може взаємодіяти з іншими компонентами операційної системи. Розглянемо схему роботи на прикладі рисунка 1 за електронним джерелом [10].

Реалізація віртуальної файлової системи може зустрічати ряд проблем, з якими розробники повинні враховувати, створюючи інтерфейс.

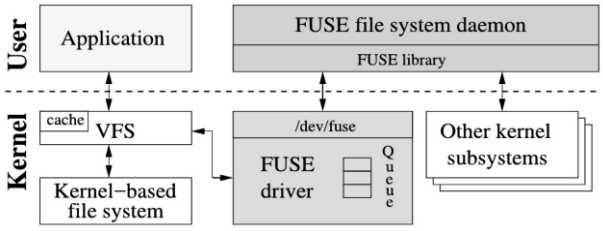

Рис. 1. Схема роботи файлової системи

Деякі з основних проблем включають:

- продуктивність: віртуальні файлові системи можуть впливати на продуктивність, оскільки вони додають додатковий шар абстракції; ефективність роботи з файлами і папками повинна бути належним чином оптимізована;
- безпека: забезпечення безпеки даних і запобігання можливим атакам на віртуальну файлову систему є важливим завданням; це включає управління доступом, шифрування інформації та інші заходи безпеки;
- сумісність: важливо враховувати сумісність віртуальної файлової системи з різними операційними системами і програмами; розробка таких систем повинна враховувати різні стандарти та протоколи;
- відновлення та відміна змін: якщо стається помилка чи відмова в роботі, важливо мати ефективні механізми відновлення та відміни змін для запобігання втраті даних або пошкодженню файлової системи;
- масштабованість: в роботі з великою кількістю файлів та об'ємом даних, важливо забезпечити ефективну масштабованість віртуальної файлової системи.

Особливості обробки та збереження даних за допомогою віртуальної файлової системи можуть бути ключовим елементом для підвищення живучості гібридних систем керування базами даних в умовах руйнівних зовнішніх впливів. Використання віртуальних файлових систем може забезпечити гнучкість і надійність зберігання даних, що стає особливо важливим у сценаріях, коли системи керування базами даних мають витримувати різні види навантаження, такі як кібератаки, природні катастрофи або технічні неполадки.

Віртуальні файлові системи можуть дозволяти ефективно реплікувати та резервувати дані, забезпечуючи їхню доступність в умовах аварійного відновлення. Наприклад, можливість автоматичного створення резервних копій даних на віртуальних дисків, розподілених по різних фізичних пристроях або хмарних сервісах, може забезпечити швидке відновлення системи та мінімізацію втрат інформації в разі непередбачених ситуацій.

Крім того, використання віртуальних файлових систем може дозволити гібридним системам керування базами даних виконувати операції резервного копіювання та міграції даних без перерв у роботі, що сприяє підвищенню їхньої живучості та доступності.

Отже, використання віртуальних файлових систем може стати важливим елементом стратегії забезпечення живучості гібридних систем керування базами даних в умовах негативних зовнішніх впливів.

Також обробка та збереження даних за допомогою віртуальної файлової системи грають важливу роль у багатокритері-

альній оптимізації балансування навантаження центрів обробки даних при варіабельності топології інтранет-мережі.

Віртуальні файлові системи можуть допомагати ефективно керувати розподілом даних між центрами обробки даних, забезпечуючи оптимальну доступність та швидкодію у разі зміни топології інтранет-мережі. Наприклад, вони можуть автоматично розподіляти навантаження залежно від доступної пропускної здатності та ресурсів кожного центру обробки даних.

Крім того, використання віртуальних файлових систем може допомогти забезпечити надійне зберігання даних у різних локаціях, що може бути важливим для забезпечення резервного копіювання та відновлення даних у випадку виникнення аварійних ситуацій.

Отже, інтеграція віртуальних файлових систем у системи керування даними може сприяти оптимізації балансування навантаження центрів обробки даних за умов варіабельності топології інтранет-мережі.

Успішність та актуальність розробки віртуальної файлової системи

Успішність віртуальних файлових систем визначається за кількома ключовими критеріями. Перш за все, це **продуктивність**, виміряна швидкодією та ефективністю роботи під високими навантаженнями. **Надійність** вказує на стійкість та можливість відновлення даних у разі виникнення збоїв. **Масштабованість** визначає здатність системи працювати ефективно при збільшенні обсягу даних. **Безпека** включає захист від несанкціонованого доступу та забезпечення конфіденційності. Інші важливі аспекти включають сумісність з іншими системами, здатність до інновацій та користувацький досвід, а також вартість власності, яка визначає витрати на утримання та розвиток файлової системи. Всі ці критерії враховуються для визначення та оцінки успішності віртуальних файлових систем у сучасному IT-середовищі.

Розробка власної ВФС, реалізованої на основі FUSE, може бути актуальною з ряду причин:

- гнучкість та розширюваність: FUSE дозволяє створювати віртуальні файлові системи в просторі користувача без необхідності зміни ядра операційної системи; це забезпечує гнучкість та розширюваність у розробці, дозволяючи легко адаптувати ВФС до конкретних потреб проєкту;
- сумісність з різними ОС: віртуальні файлові системи, розроблені за допомогою FUSE, можуть легко переноситися між різними операційними системами, оскільки FUSE підтримується на багатьох платформах (Linux, macOS, FreeBSD, інші); це робить розробку більш універсальною та ефективною;
- розвиток функціональності: розробка на основі FUSE дозволяє вам створювати власні файлові системи з розширеними функціональними можливостями, які можуть відповідати конкретним потребам користувачів чи проєкту; це особливо корисно у випадках, коли існуючі ВФС не влаштовують за вимогами;
- експериментація та навчання: розробка ВФС на основі FUSE може слугувати відмінною можливістю для навчання та експериментації в царині системного програмування; розуміння принципів роботи файлових систем та їхній вплив на взаємодію з операційною системою може бути корисним для розробників.

Програмна система, що розробляється, спрямована на створення віртуальної файлової системи (ВФС) на основі FUSE, яка надає зручний та гнучкий інтерфейс для взаємодії з файловою структурою. Основною метою цього проєкту є створення інструменту, який дозволяє користувачам легко маніпулювати віртуальною файловою структурою, використовуючи консольний інтерфейс.

Функціональні завдання та мета програмної системи сфокусовані на створенні віртуальної файлової системи (ВФС) на основі FUSE з метою надання користу-

вачам зручного інтерфейсу для взаємодії з файловою структурою. Головною метою є забезпечення функціональних можливостей для створення та видалення об'єктів, отримання детальної інформації про атрибути файлів, читання та запису вмісту файлів, роботи з символьними посиланнями, а також навігації та взаємодії з директоріями. Це включає в себе можливість створення нових файлів та директорій, видалення об'єктів, отримання детальної інформації про атрибути файлів, читання та редагування їх вмісту, а також роботу із символьними посиланнями. Подальшою метою є створення зручного та функціонального інструменту для користувачів, який може бути використаний для проведення експериментів, навчання або вирішення конкретних завдань, водночас забезпечуючи гнучкість та ефективність взаємодії з файловою системою на основі FUSE через консольний інтерфейс.

Розробка може бути застосована під час розроблення інших видів програмного забезпечення та у повсякденній взаємодії з файлами.

Цільова аудиторія для віртуальної файлової системи на основі FUSE включає розробників програмного забезпечення, системних адміністраторів, та інших IT-професіоналів, які використовують та інтегрують файлові системи у своїх проєктах. Також, система має бути пристосована для кінцевих користувачів, які використовують віртуальну файлову систему для зручного управління та взаємодії з файлами.

Архітектура програмного забезпечення

В розробці віртуальної файлової системи, використання паттернів проєктування стає важливим етапом для забезпечення ефективності, читабельності та легкості управління кодом. Один із таких патернів – Model-View-Controller (MVC) – дозволяє відокремити представлення, логіку та дані, роблячи архітектуру більш модульною та гнучкою.

Модель у віртуальній файловій системі представляє основні концепції, такі як звичайні файли, директорії та soft-посилання. Кожен із цих елементів відповідає реальній структурі файлової системи та забезпечує базовий функціонал для взаємодії з користувачем.

Представлення вирішує, як інформація буде відображатися для користувача. Контролер виконує роль посередника між користувачем та моделлю. Він приймає команди від користувача через консоль та визначає, як ці команди мають бути оброблені. Після цього він взаємодіє з моделлю для виконання необхідних операцій.

Для кращого розуміння архітектури побудуємо UML-діаграму компонентів. Розглянемо діаграму компонентів віртуальної файлової системи (Рис. 2.).

Архітектура, побудована за MVC паттерном, дозволяє зберігати код чистим, легко його розширювати та модифікувати. Розподіл обов'язків між моделлю, представленням та контролером полегшує роботу

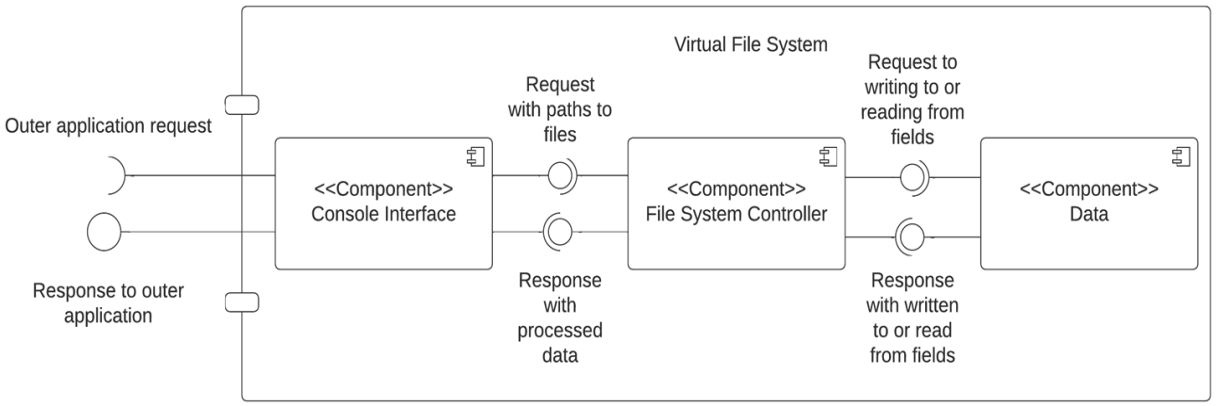

Рис. 2. Діаграма компонентів ВФС

над проєктом, забезпечуючи оптимальну організацію та легкість управління.

Опис структур даних

Детальний огляд реалізації класів і їхніх методів дозволить виявити сильні та слабкі сторони структури даних, аналізувати оптимальність вибраних рішень та вирішувати проблеми, які можуть виникнути під час реалізації віртуальної файлової системи.

Класи-моделі, "Composite" паттерн, std::variant

Розглянемо звичайний Composite паттерн [15]. Цей шаблон проєктування належить до структурних патернів і дозволяє клієнтам однаково обробляти окремі об'єкти та їхні композиції (групи об'єктів). У цьому шаблоні створюється ієрархія класів, що представляють як окремі об'єкти, так і їхні композиції, де обидва типи об'єктів реалізують спільний інтерфейс. Таким чином, клієнт може взаємодіяти з кожним об'єктом (примітивним або складним) через спільний інтерфейс, що спрощує обробку об'єктів в ієрархії.

Розглянемо UML-дігараму моделей файлової системи на прикладі віртуальної файлової системи на рисунку 3. Зліва можна побачити звичайний Composite, справа — Composite у поєднанні з TRwLock:

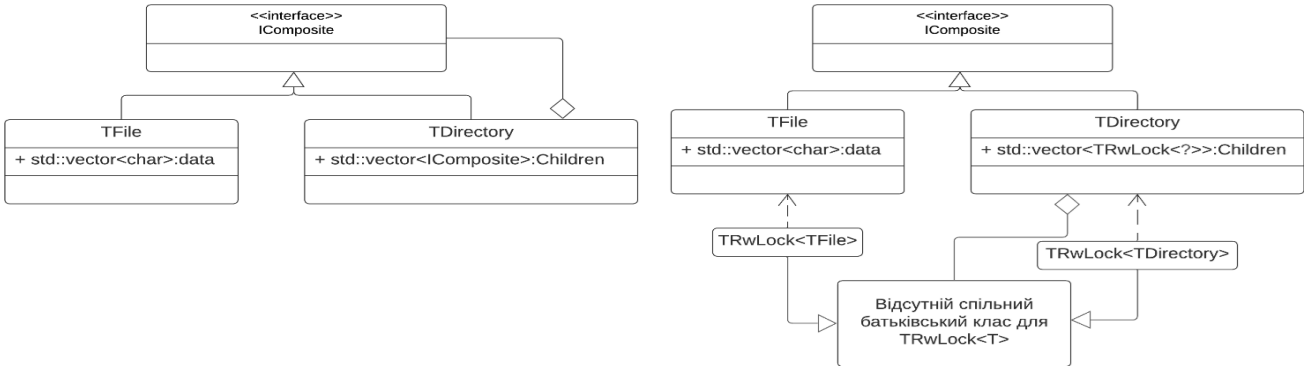

Рис. 3. Composite шаблон на прикладі ВФС

Як бачимо, TRwLock<TFile> та TRwLock<TDirectory> не мають спільного батьківського класу, а тому не можуть бути покладені разом у контейнері.

Щоб вирішити проблему, зазначену у попередньому пункті, потрібно представити певний варіантивний тип, який одночасно може представляти класи, які не мають спільного походження. Для цього у C++ був представлений std::variant.

Аналіз якості та тестування програмного забезпечення

Розглянемо детально якість та стабільність розробленого програмного продукту, а також проведемо процес тестування для впевненості в його надійності та ефективності.

Аналіз якості ПЗ

Аналіз якості ПЗ було проведено з допомогою github-застосунку CCCC (C and C++ Code Counter), який автоматично генерує звіт якості коду за різними метриками.

Загальний звіт використаних метрик можна побачити на рисунку 4.

NOM (Number of modules) — це кількість нетривіальних модулів, ідентифікованих аналізатором. Загалом кількість ідентифікованих окремих модулів для коду складає 40.

Metric	Tag	Overall
Number of modules	NOM	40
Lines of Code	LOC	1696
McCabe's Cyclomatic Number	MVG	125
Lines of Comment	COM	4

Рис. 4. Звіт якості коду CCCC

LOC (Lines of Code) — кількість непорожніх рядків вихідного коду без коментарів, підрахованих аналізатором. Загалом кількість рядків коду оцінена в 1696 рядки.

COM (Lines of Comments) — кількість рядків коментарів, визначених аналізатором. Загалом кількість рядків коментарів оцінена в 4 рядки. Ця метрика може бути досить важливою для великої кодової бази, однак у даній роботі автор намагався писати самодокументований код, за яким особа, що читає, може зрозуміти сенс написаного.

MVG (McCabe's Cyclomatic Complexity) — міра складності рішення для функцій, які складають програму. Суворе визначення цієї міри полягає в тому, що це кількість лінійно незалежних маршрутів через спрямований ациклічний граф, який відображає потік керування підпрограмою. Аналізатор підраховує це, записуючи кількість різних результатів рішення, що містяться в кожній функції, що дає гарне наближення до формально визначеної версії міри. Загалом аналізатор показує 125.

Опис процесів тестування

Тестування віртуальної файлової системи було проведено з використанням фреймворку для тестування googletest. Вибір googletest був обумовлений його визнаною ефективністю, гнучкістю та набором зручних інструментів для написання та виконання тестів.

Однією з основних переваг googletest є його зручний і легкий у використанні синтаксис для написання тестів. Тестові кейси та перевірки можуть бути легко створені і організовані за допомогою зрозумілих механізмів структури тестових класів та макросів, що полегшує процес написання і управління тестами.

Ще однією важливою перевагою є можливість паралельного виконання тестів. Це забезпечує швидший оборот часу за великої кількості тестових кейсів, що сприяє ефективності та розподілу ресурсів у процесі тестування.

Покажемо оцінку продуктивності, а саме швидкість пошуку файлів за їхнім іменем. Для цього напишемо юніт-тест скрипт, що заповнить ВФС директоріями від A-Z, потім всередині кожної директорії іще раз створить директорії від A-Z, повторить цю операцію 4 рази, тобто у результаті будемо мати 28 ^ 4 = 614656 директорій. Під час тестування, скажемо ВФС знайти усі повні шляхи директорій, які мають ім’я H (рис. 5).

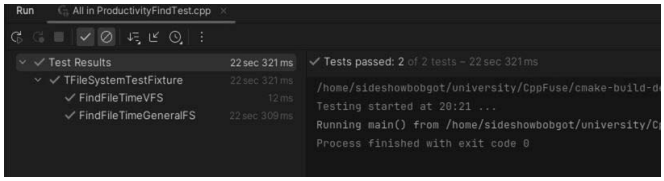

Рис. 5. Результати пошуку

Бачимо, що ВФС знайшла усі файли з іменем “H” за 12 мілісекунд, натомість файлова система Linux віднайшла їх за 22 секунди 309 мілісекунд. **Прискорення у 1859 разів.**

Висновки

У роботі розглянуто концепцію віртуальних файлових систем, проведений аналіз існуючих рішень та обґрунтована необхідність розробки нової файлової системи, яка відповідає вимогам сучасності та враховує актуальні тенденції в області обробки та збереження даних. Під час виконання роботи було створено інструмент, який дозволяє користувачам легко маніпулювати віртуальною файловою структурою, використовуючи консольний інтерфейс, що уможливлює пришвидшення обробки файлів за допомогою кастомізації операцій файлової системи.

У процесі аналізу вимог до Віртуальної Файлової Системи (ВФС) визначено ключові функціональні та нефункціональні вимоги. Серед них — висока ефективність, забезпечення безпеки та гнучкість використання. Спроєктовано основні взаємодії з файловою системою, визначено формати даних та розглянуто основні сценарії використання.

Під час проведення етапу моделювання та конструювання була розроблена архітектура ВФС. Вона включає класи та компоненти, представлені для файлів та директорій. Використано різні паттерни проєктування, такі як Visitor для забезпечення різноманітності операцій над різними типами файлів. Статичний поліморфізм реалізовано за допомогою шаблонів та CRTP паттерну.

Тестування ВФС було проведено за допомогою фреймворку googletest. Обрано

його через високу ефективність, гнучкість та можливість паралельного виконання тестів. Тестування охопило різні аспекти функціональності, забезпечивши стабільність та надійність програмного забезпечення.

На етапі впровадження були детально описані кроки для розгортання ВФС. Це включало встановлення необхідних бібліотек, створення FIFO-файлу для міжпроцесної комунікації та встановлення зв'язку з основною файловою системою. Документовано командні аргументи для легкості розгортання та налаштування.

Майбутній розвиток ВФС може включати розширення функціональності для підтримки нових операцій, покращення ефективності та роботи з великим обсягом даних. Інтеграція з іншими системами та підтримка нових типів файлів можуть стати напрямками подальшого розвитку. Постійна оптимізація та удосконалення можуть допомогти ВФС вдосконалюватися та відповідати зростаючим вимогам.

Практична значущість розробленої системи полягає в наступному:

- Розроблена файлова система може бути застосована під час розробки інших видів програмного забезпечення та у повсякденній взаємодії з файлами.
- Цільова аудиторія для віртуальної файлової системи на основі FUSE включає розробників програмного забезпечення, системних адміністраторів, та інших IT-професіоналів, які використовують та інтегрують файлові системи у своїх проєктах.
- Систему вдало використовують системні адміністратори, які опікуються серверами із значною кількістю файлів. Їхні завдання пов'язані з необхідністю швидкого пошуку файлів, і саме це було вирішено завдяки впровадженню розробленої файлової системи. Застосування файлової системи великою мірою скоротило час пошуку файлів, сприяючи оптимізації завдань системних адміністраторів. Консольний інтерфейс системи виявився зручним та легким у використанні, що полегшує роботу користувачів.
- Розробники також здобули доступ до відкритого вихідного коду системи, що дозволяє їм налаштовувати операції з файлами згідно зі своїми потребами.

У майбутньому можливе створення NetFS – розподіленої файлової системи, яка дозволить взаємодіяти з файлами та директоріями через мережу.

Література

1. Shun Ishiguro, Jun Murakami, Yoshihiro Oyama, and Osamu Tatebe. Optimizing local file accesses for FUSE-based distributed storage. In High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion, pages 760765. IEEE, 2012.
2. R. Card, T. Ts'o, and S. Tweedie. Design and implementation of the second extended filesystem. In Proceedings to the First Dutch International Symposium on Linux, Seattle, WA, December 2004.
3. Michael Condict, Don Bolinger, Dave Mitchell, and Eamonn McManus. Microkernel Modularity with Integrated Kernel Performance. In Proceedings of the First Symposium on Operating Systems Design and Implementation, 2014.
4. Opendedup, January 2012. www.opendedup.org.
5. J. Bent, G. Gibson, G. Grider, B. McClelland, P. Nowoczynski, J. Nunez, M. Polte, and M. Wingate. Plfs: A checkpoint filesystem for parallel applications. Technical Report LA-UR 0902117, LANL, 2009. http://institute.lanl.gov/ plfs/.
6. C. Ungureanu, B. Atkin, A. Aranya, S. Gokhale, S. Rago, G.Calkowski, C.Dubnicki, and A.Bohra. HydraFS: a High-Throughput File System for the HYDRAstor Content-Addressable Storage System. In Proceedings of the FAST Conference, 2010.

7. ZFS for Linux, January 2016. www.zfs-fuse.net.
8. NTFS-3G. USENIX Association 15th USENIX Conference on File and Storage Technologies
9. K. Vangoor, V. Tarasov, E. Zadok, To FUSEorNotto FUSE: Performance of User-Space File Systems Bharath, Proceedings of the 15th USENIX Conference on File and Storage Technologies 2017, p. 59-72.
10. FUSE - The Linux Kernel Documentation. (2023). Retrieved October 7, 2023, from https://www.kernel.org/doc /html/ next/filesystems/fuse.html

Одержано: 10.04.2024
Внутрішня рецензія отримана: 17.04.2024
Зовнішня рецензія отримана: 23.04.2024

Про авторів:

[1]*Поперешняк Світлана Володимирівна*,
Кандидат фізико-математичних наук,
доцент
http://orcid.org/0000-0002-0531-9809.

[2]*Панченко Сергій Віталійович*,
бакалавр
http://orcid.org/0009-0000-8897-8182.

[3] *Федорченко Олексій Дмитрович*,
аспірант
http://orcid.org/0009-0006-2556-7574.

[4] *Ільїн Сергій Анатолійович*,
аспірант
http://orcid.org/0009-0004-3760-1464.

Місце роботи авторів:

[1, 2] Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»,
тел. +38-098-645-54-62
E-mail: spopereshnyak@gmail.com
E-mail: panchenko.serhii@lll.kpi.ua

[3,4] Інститут програмних систем НАН України