

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TPHCM

KHOA: CÔNG NGHỆ THÔNG TIN



BÁO CÁO CUỐI KỲ

MÔN: LẬP TRÌNH HỆ THỐNG

XÂY DỰNG HỆ THỐNG TẠP TIN FUSE

Sinh viên thực hiện:

1. Nguyễn Thắng Lợi - 22162023

2. Lê Anh Khoa – 22162016

3. Mai Thị Quỳnh Như - 22162

TP. Hồ Chí Minh, tháng 04 năm 2025

Mục Lục

PHẦN 1: PHẦN Ở ĐẦU.....	3
1.1. Tóm tắt.....	3
1.2. Tóm lược những nghiên cứu trong và ngoài nước liên quan đến đề tài.....	3
1.3. Mục tiêu đề tài.....	3
1.4. Đối tượng và phạm vi nghiên cứu.....	4
1.5. Phương pháp nghiên cứu.....	4
1.6. Nội dung đề tài.....	4
PHẦN 2: PHẦN NỘI DUNG.....	6
 CHƯƠNG 1: TỔNG QUAN.....	6
1.1. Bối cảnh và sự cần thiết của hệ thống tệp truy cập từ xa.....	6
1.2. Các công trình liên quan.....	6
1.3. Giới thiệu giải pháp RemoteFS.....	7
 CHƯƠNG 2: CƠ SỞ LÝ THUYẾT.....	8
2.1. FUSE (Filesystem in Userspace).....	8
2.2. SSH (Secure Shell).....	10
 CHƯƠNG 3: MỤC TIÊU THIẾT KẾ.....	13
3.1. Phân tích các yêu cầu thiết kế.....	13
3.2. Các mục tiêu thiết kế.....	14
3.3. Phạm vi dự án.....	14
 CHƯƠNG 4: ỨNG DỤNG.....	16
4.1. Sơ đồ khái ứng dụng.....	16
4.2. Kiến trúc và nguyên lý hoạt động.....	17

4.3 Quy trình phát triển.....	18
PHẦN 3: PHẦN KẾT LUẬN.....	20
3.1. Kết quả đạt được.....	20
3.2. Ưu nhược điểm.....	22
3.3. Hướng phát triển của đề tài.....	23
TÀI LIỆU THAM KHẢO.....	24

PHẦN 1: PHẦN MỞ ĐẦU

1.1. Tóm tắt

Báo cáo này mô tả dự án RemoteFS, một ứng dụng hệ thống tệp trong không gian người dùng (user-space filesystem) được phát triển dựa trên khuôn khổ FUSE (Filesystem in Userspace) và giao thức SSH. Ý tưởng cốt lõi là cho phép người dùng gắn kết (mount) một thư mục cụ thể trên máy chủ từ xa vào một điểm trong hệ thống tệp cục bộ của họ, tạo ra một giao diện liền mạch để tương tác với các tệp từ xa như thể chúng đang ở trên máy tính của chính họ. Nội dung báo cáo bao gồm cơ sở lý thuyết về FUSE và SSH, phân tích yêu cầu kỹ thuật, thiết kế kiến trúc, quy trình phát triển ứng dụng, các kết quả đã đạt được, cũng như những ưu nhược điểm và định hướng phát triển cho đề tài.

1.2. Tóm lược những nghiên cứu trong và ngoài nước liên quan đến đề tài

Việc truy cập và quản lý tài nguyên từ xa là một lĩnh vực quan trọng trong khoa học máy tính và mạng máy tính. Các phương pháp truyền thống cho phép truy cập tệp từ xa bao gồm:

- **Truyền tệp thủ công:** Sử dụng các công cụ như scp (Secure Copy Protocol) hoặc sftp (SSH File Transfer Protocol). Các phương pháp này yêu cầu người dùng thực hiện các lệnh rõ ràng để sao chép hoặc di chuyển tệp.
- **Hệ thống tệp mạng truyền thống:** NFS (Network File System) cho môi trường Unix/Linux và SMB/CIFS (Server Message Block/Common Internet Internet Filesystem) cho môi trường Windows là các giao thức phức tạp được triển khai ở cấp độ kernel hoặc dưới dạng dịch vụ hệ thống, yêu cầu cấu hình đáng kể trên cả máy chủ và máy khách.
- **Các hệ thống tệp dựa trên FUSE:** FUSE đã mở ra khả năng phát triển các hệ thống tệp tùy chỉnh trong không gian người dùng một cách an toàn và dễ dàng hơn. Nhiều dự án đã sử dụng FUSE để tạo ra các hệ thống tệp ảo (ví dụ: loopbackfs, unionfs) hoặc truy cập dữ liệu từ các nguồn khác nhau (ví dụ: Google Drive FS, S3FS, SSHFS).

Dự án RemoteFS nằm trong nhóm các hệ thống tệp dựa trên FUSE, cụ thể là tập trung vào việc sử dụng SSH làm kênh truyền tải và xác thực, tương tự như SSHFS nhưng có thể có cách tiếp cận triển khai khác hoặc tập trung vào các khía cạnh cụ thể của việc ánh xạ FUSE calls sang SSH commands. Việc nghiên cứu các công cụ và dự án hiện có giúp hiểu rõ các phương pháp đã được thử nghiệm và các thách thức thường gặp khi xây dựng hệ thống tệp từ xa qua mạng.

1.3. Mục tiêu đề tài

Mục tiêu chính của đề tài là:

- Nghiên cứu và hiểu rõ cơ chế hoạt động của khuôn khổ FUSE.
- Nghiên cứu cách sử dụng giao thức SSH và các thư viện hỗ trợ (ví dụ: libssh) để thực thi lệnh từ xa và truyền dữ liệu.
- Xây dựng một ứng dụng (daemon) trong không gian người dùng triển khai các hàm callback của FUSE.
- Ánh xạ các yêu cầu hệ thống tệp từ FUSE sang các thao tác tương ứng được thực hiện qua kết nối SSH đến máy chủ từ xa.
- Cho phép người dùng gắn kết một thư mục remote và thao tác với các tệp bên trong nó thông qua các công cụ hệ thống tệp cục bộ.

1.4. Đối tượng và phạm vi nghiên cứu

- **Đối tượng nghiên cứu:** Khuôn khổ FUSE, giao thức SSH, các thư viện lập trình cho FUSE (libfuse) và SSH (libssh), cùng với mã nguồn và kiến trúc của ứng dụng RemoteFS.
- **Phạm vi nghiên cứu:** Đề tài tập trung vào việc triển khai các thao tác hệ thống tệp cơ bản (đọc, ghi, liệt kê, lấy thông tin, tạo, xóa) trên một thư mục remote thông qua kết nối SSH sử dụng FUSE. Đề tài không đi sâu vào các tính năng nâng cao của hệ thống tệp (ví dụ: quản lý quyền hạn phức tạp, symlinks/hardlinks, extended attributes, file locking) hoặc tối ưu hóa hiệu năng chuyên sâu vượt ra ngoài phạm vi cơ bản.

1.5. Phương pháp nghiên cứu

- **Nghiên cứu lý thuyết:** Tìm hiểu sâu về cấu trúc và API của FUSE thông qua tài liệu và bài nghiên cứu [1]. Nghiên cứu về giao thức SSH và cách thực hiện các tác vụ tệp thông qua SSH (ví dụ: exec commands, SFTP).
- **Phân tích mã nguồn:** Nghiên cứu mã nguồn dự án RemoteFS tại kho GitHub [2] để hiểu cách các khái niệm lý thuyết (FUSE, SSH) được áp dụng trong thực tế, cấu trúc module và cách các thao tác được triển khai.
- **Phát triển phần mềm:** Xây dựng và triển khai ứng dụng RemoteFS bằng ngôn ngữ C, sử dụng libfuse và libssh. Quá trình này bao gồm viết mã, biên dịch, kiểm thử và gỡ lỗi.
- **Kiểm thử thực nghiệm:** Thực hiện các bài kiểm thử để xác nhận các chức năng đã triển khai hoạt động chính xác, đánh giá hiệu năng cơ bản và xác định các hạn chế.

1.6. Nội dung đề tài

Báo cáo này được tổ chức theo bố cục sau:

- **Phần 1: Phần mở đầu:** Giới thiệu tổng quan về đề tài, bối cảnh nghiên cứu, mục tiêu, phạm vi và phương pháp nghiên cứu.

- **Phần 2: Phần nội dung:** Trình bày chi tiết các khía cạnh kỹ thuật của dự án.
 - Chương 1: Giới thiệu vấn đề cần giải quyết.
 - Chương 2: Phân tích các yêu cầu chức năng của hệ thống.
 - Chương 3: Trình bày các kiến thức nền tảng liên quan (FUSE, SSH).
 - Chương 4: Mô tả kiến trúc, các module, quy trình xây dựng và phân tích khó khăn của ứng dụng RemoteFS.
- **Phần 3: Phần kết luận:** Tóm tắt kết quả đạt được, thảo luận về ưu nhược điểm và đưa ra các hướng phát triển trong tương lai.

PHẦN 2: PHẦN NỘI DUNG

CHƯƠNG 1: TỔNG QUAN

1.1. *Bối cảnh và sự cần thiết của hệ thống tệp truy cập từ xa.*

Trong môi trường làm việc và học tập hiện đại, nhu cầu truy cập và xử lý dữ liệu lưu trữ trên các máy chủ từ xa ngày càng trở nên phổ biến. Người dùng thường xuyên phải làm việc với các tài nguyên không nằm trên máy tính cá nhân của họ, từ các máy chủ phát triển, máy chủ lưu trữ dữ liệu chung cho đến các môi trường điện toán đám mây. Việc truy cập và quản lý các tài nguyên từ xa này một cách hiệu quả và liền mạch là một yêu cầu thiết yếu. Các phương pháp truy cập truyền thống thường đòi hỏi người dùng phải thực hiện nhiều thao tác thủ công hoặc đối mặt với sự phức tạp trong cấu hình, gây gián đoạn luồng công việc và giảm năng suất. Do đó, việc phát triển các giải pháp cho phép tương tác với hệ thống tệp từ xa một cách tự nhiên, như thế chúng là một phần của hệ thống tệp cục bộ, là vô cùng cần thiết.

1.2. *Các công trình liên quan*

Để giải quyết nhu cầu truy cập tệp từ xa, nhiều giải pháp đã được phát triển và ứng dụng rộng rãi, mỗi giải pháp có những ưu và nhược điểm riêng:

- **Truyền tệp thủ công:** Các công cụ như scp (Secure Copy Protocol) hay sftp (SSH File Transfer Protocol) cho phép sao chép tệp an toàn qua mạng. Tuy nhiên, chúng yêu cầu người dùng phải thực hiện các lệnh một cách tường minh cho mỗi lần truyền dữ liệu và không cung cấp khả năng truy cập trực tiếp, liền mạch vào cấu trúc thư mục từ xa như một hệ thống tệp thông thường.
- **Hệ thống tệp mạng truyền thống:** NFS (Network File System) và SMB/CIFS (Server Message Block/Common Internet File System) là các giao thức phổ biến, thường được triển khai ở cấp độ kernel hoặc dưới dạng dịch vụ hệ thống. Chúng cung cấp khả năng truy cập tệp từ xa một cách minh bạch hơn, nhưng việc cài đặt và cấu hình thường phức tạp, đòi hỏi quyền quản trị và sự tương thích giữa máy chủ và máy khách.
- **Hệ thống tệp dựa trên FUSE:** Sự ra đời của FUSE (Filesystem in Userspace) đã mở ra một hướng tiếp cận mới, cho phép phát triển các hệ thống tệp trong không gian người dùng một cách linh hoạt và an toàn hơn. Nhiều dự án đã tận dụng FUSE để tạo ra các hệ thống tệp ảo hoặc kết nối đến các nguồn dữ liệu đa dạng. Một ví dụ tiêu biểu và liên quan trực tiếp là SSHFS, một hệ thống tệp FUSE sử dụng SFTP qua SSH để mount các thư mục từ xa. Các dự án như vậy chứng minh tính khả thi và hiệu quả của việc sử dụng FUSE để xây dựng cầu nối giữa hệ thống tệp cục bộ và các nguồn dữ liệu từ xa. Việc nghiên cứu các giải pháp này giúp hiểu rõ hơn về các kỹ thuật đã được áp dụng và những thách thức cần giải quyết.

1.3. Giới thiệu giải pháp RemoteFS

Dự án RemoteFS được đề xuất như một giải pháp xây dựng hệ thống tệp truy cập từ xa, dựa trên sự kết hợp giữa khuôn khổ FUSE và giao thức SSH. Ý tưởng cốt lõi là phát triển một ứng dụng hệ thống tệp trong không gian người dùng, cho phép người dùng "gắn kết" (mount) một thư mục trên máy chủ SSH từ xa vào một điểm mount trên hệ thống tệp cục bộ của họ.

Khi được mount, người dùng có thể tương tác với các tệp và thư mục từ xa thông qua các công cụ và ứng dụng hệ thống tệp thông thường (ví dụ: ls, cd, cat, trình quản lý tệp đồ họa) như thể chúng đang tồn tại trên máy tính cục bộ. RemoteFS tận dụng tính linh hoạt của FUSE để xử lý các yêu cầu hệ thống tệp ở user-space và dựa vào tính bảo mật, khả năng xác thực mạnh mẽ cùng cơ chế truyền dữ liệu mã hóa của giao thức SSH để đảm bảo an toàn cho việc truy cập tài nguyên từ xa. Mục tiêu chính là cung cấp một giao diện liền mạch, an toàn và dễ sử dụng để làm việc với các hệ thống tệp từ xa qua mạng.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. FUSE (*Filesystem in Userspace*)

2.1.1. Khái niệm và kiến trúc

FUSE (Filesystem in Userspace) là một khuôn khổ phần mềm dành cho các hệ điều hành tương tự Unix (như Linux, macOS, FreeBSD), cho phép những người dùng không có đặc quyền (non-privileged users) tạo ra các hệ thống tệp của riêng họ mà không cần phải viết mã ở cấp độ hạt nhân (kernel-space). Thay vào đó, mã hệ thống tệp được chạy trong không gian người dùng (user-space). Điều này mang lại nhiều lợi ích, bao gồm quy trình phát triển đơn giản và nhanh chóng hơn, khả năng sử dụng các thư viện user-space tiêu chuẩn, tăng cường tính ổn định (lỗi trong mã FUSE filesystem thường không làm sập toàn bộ hệ thống) và không yêu cầu sửa đổi hay biên dịch lại kernel.

Kiến trúc của FUSE bao gồm ba thành phần chính:

1. **Kernel Module (fuse.ko):** Đây là thành phần hoạt động trong kernel-space. Nó đăng ký một hệ thống tệp ảo mới với VFS (Virtual File System) của kernel. Khi có một thao tác tệp diễn ra trên một mount point của FUSE, VFS sẽ chuyển yêu cầu đó đến module fuse.ko. Module này không tự xử lý yêu cầu mà đóng vai trò trung gian, chuyển tiếp yêu cầu từ kernel đến tiến trình user-space tương ứng thông qua một thiết bị đặc biệt (/dev/fuse). Nó cũng nhận phản hồi từ tiến trình user-space và trả về cho VFS.
2. **Thư viện User-space (libfuse):** Đây là thư viện chạy trong user-space, cung cấp một API cấp cao để các nhà phát triển xây dựng hệ thống tệp của họ. libfuse đảm nhiệm việc giao tiếp với kernel module qua /dev/fuse, xử lý việc nhận yêu cầu từ kernel, gọi các hàm callback do người dùng định nghĩa, và gửi trả kết quả về lại kernel. Nó đơn giản hóa đáng kể quá trình phát triển FUSE filesystem.
3. **User-space Daemon (Tiến trình hệ thống tệp):** Đây là chương trình do nhà phát triển viết, chạy như một tiến trình nền (daemon) trong user-space. Tiến trình này liên kết với libfuse và chứa logic thực sự của hệ thống tệp. Nó triển khai một tập hợp các hàm (gọi là callbacks hoặc operations) tương ứng với các thao tác hệ thống tệp tiêu chuẩn (như đọc, ghi, tạo thư mục, lấy thông tin tệp, v.v.). Khi libfuse nhận được yêu cầu từ kernel sẽ gọi hàm callback tương ứng trong daemon này để xử lý.

2.1.2. Luồng hoạt động

Luồng xử lý một yêu cầu hệ thống tệp trong FUSE diễn ra như sau:

- Một ứng dụng người dùng (ví dụ: lệnh ls, cat, hoặc một trình quản lý tệp) thực hiện một thao tác trên một tệp hoặc thư mục nằm trong mount point của FUSE filesystem (ví dụ: gọi hàm readdir() để liệt kê thư mục).

- Lời gọi hàm hệ thống này được chuyển đến tầng VFS của kernel.
- VFS xác định rằng thao tác này thuộc về một FUSE filesystem và chuyển tiếp yêu cầu đến FUSE kernel module (fuse.ko).
- Kernel module fuse.ko đóng gói yêu cầu này và ghi nó vào file descriptor của thiết bị /dev/fuse đang được lắng nghe bởi tiến trình FUSE daemon tương ứng.
- Thư viện libfuse trong tiến trình FUSE daemon đọc yêu cầu từ file descriptor /dev/fuse.
- libfuse giải mã yêu cầu và gọi đến hàm callback cụ thể mà nhà phát triển đã đăng ký để xử lý loại thao tác đó (ví dụ: hàm my_readdir).
- Hàm callback trong user-space daemon thực thi logic cần thiết để đáp ứng yêu cầu (ví dụ: trong RemoteFS, nó có thể gửi một lệnh ls qua SSH đến máy chủ từ xa).
- Sau khi xử lý xong, hàm callback trả kết quả (dữ liệu hoặc mã lỗi) về cho libfuse.
- libfuse đóng gói phản hồi và ghi nó trở lại file descriptor /dev/fuse.
- FUSE kernel module đọc phản hồi này.
- Kernel module chuyển phản hồi về lại cho VFS.
- VFS trả kết quả cuối cùng về cho ứng dụng người dùng ban đầu.

Toàn bộ quá trình này diễn ra minh bạch đối với ứng dụng người dùng, như thể là đang tương tác với một hệ thống tệp thông thường.

2.1.3. Các hàm callback quan trọng

Để triển khai một FUSE filesystem, nhà phát triển cần định nghĩa một cấu trúc kiểu **struct fuse_operations**. Cấu trúc này chứa các con trỏ hàm, mỗi con trỏ trỏ đến một hàm callback do nhà phát triển viết để xử lý một loại thao tác hệ thống tệp cụ thể. Thư viện libfuse sẽ gọi các hàm này khi nhận được yêu cầu tương ứng từ kernel.

Một số hàm callback quan trọng thường được triển khai bao gồm:

- **getattr:** Lấy thuộc tính của tệp hoặc thư mục (kích thước, quyền, thời gian sửa đổi, v.v.). Thường được gọi rất thường xuyên.
- **readdir:** Đọc nội dung của một thư mục (liệt kê các tệp và thư mục con).
- **open:** Mở một tệp. Được gọi trước các thao tác đọc (read) hoặc ghi (write).
- **read:** Đọc dữ liệu từ một tệp đang mở.

- **write:** Ghi dữ liệu vào một tệp đang mở.
- **release:** Đóng một tệp đang mở (giải phóng tài nguyên).
- **mkdir:** Tạo một thư mục mới.
- **rmdir:** Xóa một thư mục rỗng.
- **unlink:** Xóa một tệp tin.
- **rename:** Đổi tên hoặc di chuyển một tệp/thư mục.
- **create:** Tạo và mở một tệp mới.
- **statfs:** Lấy thông tin về toàn bộ hệ thống tệp tin (dung lượng trống, kích thước block, v.v.).

Trong dự án RemoteFS, các hàm callback này được triển khai để ánh xạ các yêu cầu hệ thống tệp từ FUSE thành các thao tác tương ứng thực hiện qua kết nối SSH đến máy chủ từ xa. Ví dụ, hàm readdir có thể được triển khai bằng cách gửi lệnh ls -la qua SSH và phân tích kết quả trả về.

2.2. SSH (*Secure Shell*)

2.2.1. Tổng quan

SSH (*Secure Shell*) là một giao thức mạng mã được thiết kế để cung cấp kênh giao tiếp an toàn qua một mạng không an toàn (như Internet). Mục đích chính của SSH là thay thế các giao thức đăng nhập từ xa không an toàn như Telnet hay rlogin và các giao thức truyền tệp không an toàn như FTP. SSH đảm bảo tính **bảo mật (confidentiality)** và **toàn vẹn (integrity)** của dữ liệu trao đổi giữa hai máy tính bằng cách sử dụng mã hóa mạnh, đồng thời cung cấp cơ chế **xác thực (authentication)** mạnh mẽ cho cả client và server.

Kiến trúc của SSH dựa trên mô hình client-server:

- **SSH Client:** Là chương trình người dùng khởi tạo kết nối đến máy chủ SSH.
- **SSH Server (sshd):** Là một tiến trình nền (daemon) chạy trên máy chủ, lắng nghe các kết nối SSH đến, xử lý xác thực và thực thi các yêu cầu từ client.

Để đảm bảo an toàn, SSH sử dụng một loạt các kỹ thuật mã:

- **Thiết lập kênh an toàn:** Khi kết nối được thiết lập, client và server thương lượng các thuật toán mã hóa và thực hiện một quy trình trao đổi khóa (ví dụ: Diffie-Hellman) để tạo ra một khóa phiên bí mật (session key). Khóa này sau đó được sử dụng với một thuật toán mã hóa đối xứng (như AES, ChaCha20) để mã hóa toàn bộ dữ liệu truyền đi sau đó, đảm bảo tính bảo mật. Các mã xác thực

thông điệp (MAC) cũng được sử dụng để đảm bảo tính toàn vẹn dữ liệu, chống lại việc sửa đổi trên đường truyền.

- **Xác thực Server:** Client xác thực định danh của server bằng cách kiểm tra khóa công khai của server (host key). Lần đầu kết nối, client thường lưu lại host key này và kiểm tra lại trong các lần kết nối sau để chống lại tấn công man-in-the-middle.
- **Xác thực Client:** Server xác thực định danh của người dùng muốn kết nối. Các phương thức xác thực phổ biến bao gồm:
 - Xác thực bằng mật khẩu (password authentication).
 - Xác thực bằng khóa công khai (public-key authentication): An toàn hơn mật khẩu, người dùng tạo một cặp khóa (private key và public key), public key được đặt trên server, và client chứng minh sở hữu private key tương ứng mà không cần gửi private key qua mạng.

2.2.2. Các phương thức tương tác

SSH không chỉ dùng để đăng nhập tương tác vào shell từ xa mà còn cung cấp các phương thức tương tác khác, đặc biệt hữu ích cho các ứng dụng tự động hóa và tích hợp như RemoteFS:

- **Thực thi lệnh từ xa:** SSH cho phép client gửi một lệnh cụ thể đến server để thực thi mà không cần mở một phiên shell tương tác đầy đủ. Ví dụ: `ssh user@host 'ls -l /path/to/dir'`. Server sẽ thực thi lệnh và gửi kết quả (stdout, stderr) trở lại client. Cơ chế này là nền tảng để RemoteFS thực hiện các thao tác không liên quan trực tiếp đến truyền nội dung file, ví dụ như liệt kê thư mục (ls), tạo thư mục (mkdir), xóa (rm, rmdir), lấy thông tin tệp (stat), đổi tên (mv), v.v.
- **SFTP (SSH File Transfer Protocol):** Đây là một giao thức mạng chạy trên nền SSH, cung cấp một tập hợp các thao tác chuẩn hóa để truy cập, truyền tải và quản lý tệp tin trên hệ thống từ xa một cách an toàn. SFTP hoạt động như một hệ thống con (subsystem) của SSH. Nó cung cấp các lệnh như đọc tệp, ghi tệp, tạo tệp, xóa tệp, liệt kê thư mục, lấy thuộc tính tệp,... một cách có cấu trúc hơn so với việc chỉ thực thi các lệnh shell cơ bản như cat hay cp. Trong RemoteFS, SFTP là phương thức hiệu quả và đáng tin cậy để triển khai các hàm FUSE liên quan đến việc đọc và ghi nội dung tệp (read, write, create).

2.2.3. Thư viện hỗ trợ: Giới thiệu về thư viện được sử dụng (ví dụ: libssh).

Để tích hợp chức năng SSH vào một ứng dụng như RemoteFS, thay vì tự triển khai lại toàn bộ giao thức phức tạp, các nhà phát triển thường sử dụng các thư viện hỗ trợ. Trong dự án này, thư viện được sử dụng là libssh.

libssh là một thư viện đa nền tảng viết bằng ngôn ngữ C, giúp triển khai các chức năng của giao thức SSH (phiên bản 1 và 2) từ phía client hoặc server. Nó cung cấp một API cho phép các lập trình viên dễ dàng:

- Thiết lập và quản lý các kết nối SSH.
- Thực hiện các phương thức xác thực khác nhau (mật khẩu, public key).
- Mở các kênh SSH (channels).
- Thực thi các lệnh từ xa trên server.
- Truyền tệp tin sử dụng giao thức SFTP.
- Thiết lập chuyển tiếp cổng (port forwarding).

Việc sử dụng libssh trong RemoteFS giúp trừu tượng hóa các chi tiết cấp thấp của giao thức SSH, cho phép tập trung vào logic chính là ánh xạ các thao tác FUSE thành các lời gọi hàm tương ứng của libssh để tương tác với máy chủ SSH từ xa.

CHƯƠNG 3: MỤC TIÊU THIẾT KẾ

3.1. Phân tích các yêu cầu thiết kế

Việc thiết kế RemoteFS cần đáp ứng các yêu cầu chức năng và phi chức năng cơ bản để đảm bảo hệ thống hoạt động hiệu quả và đúng mục đích.

Yêu cầu về mặt kỹ thuật:

- **Gắn kết (Mounting):** Hệ thống phải cho phép người dùng gắn kết một thư mục cụ thể trên máy chủ SSH từ xa vào một điểm gắn kết (mount point) trên hệ thống tệp cục bộ.
- **Ngắt kết nối (Unmounting):** Cung cấp cơ chế để ngắt kết nối thư mục từ xa một cách an toàn.
- **Thao tác tệp cơ bản:** Hỗ trợ các hoạt động cơ bản trên tệp tin bao gồm đọc (read), ghi (write), tạo mới (create), xóa (unlink), lấy thông tin thuộc tính (getattr).
- **Thao tác thư mục cơ bản:** Hỗ trợ các hoạt động cơ bản trên thư mục bao gồm liệt kê nội dung (readdir), tạo mới (mkdir), xóa (rmdir), lấy thông tin thuộc tính (getattr).
- **Truy cập trong suốt:** Sau khi gắn kết, việc truy cập các tệp và thư mục từ xa phải diễn ra một cách tự nhiên thông qua các công cụ và ứng dụng hệ thống tệp cục bộ mà không yêu cầu người dùng thực hiện các lệnh đặc biệt cho từng thao tác.

Yêu cầu về mặt tổ chức:

- **Bảo mật:** Toàn bộ quá trình giao tiếp với máy chủ từ xa (xác thực, truyền dữ liệu, thực thi lệnh) phải được bảo mật. RemoteFS cần tận dụng tối đa các cơ chế bảo mật sẵn có của giao thức SSH.
- **Tính khả dụng cơ bản:** Hệ thống phải hoạt động ổn định trong điều kiện kết nối mạng và máy chủ SSH hoạt động bình thường. Cần có cơ chế xử lý lỗi cơ bản khi kết nối SSH gặp sự cố.
- **Tính dễ sử dụng:** Việc gắn kết và sử dụng hệ thống tệp từ xa cần tương đối đơn giản cho người dùng đã quen thuộc với dòng lệnh và SSH.
- **Hiệu năng chấp nhận được:** Mặc dù không phải là mục tiêu chính, hiệu năng của các thao tác tệp cần đủ tốt để không gây khó chịu cho người dùng trong các tác vụ thông thường. Không yêu cầu tối ưu hóa hiệu năng chuyên sâu ở giai đoạn này.

3.2. Các mục tiêu thiết kế

Dựa trên các yêu cầu đã phân tích, các mục tiêu thiết kế cụ thể cho RemoteFS được đặt ra như sau:

1. **Xây dựng trên nền FUSE:** Thiết kế và triển khai một tiến trình daemon trong không gian người dùng (user-space) sử dụng thư viện libfuse để đăng ký và xử lý các yêu cầu từ VFS (Virtual File System) của hệ điều hành.
2. **Sử dụng SSH làm phương tiện vận chuyển:** Tận dụng thư viện libssh để thiết lập, xác thực và duy trì các kết nối SSH đến máy chủ từ xa một cách an toàn và hiệu quả.
3. **Ánh xạ thao tác FUSE sang SSH/SFTP:** Thiết kế cơ chế ánh xạ rõ ràng và hợp lý giữa các hàm callback trong struct fuse_operations (ví dụ: getattr, readdir, read, write, mkdir) với các hành động tương ứng trên máy chủ từ xa thông qua việc thực thi lệnh SSH hoặc sử dụng giao thức SFTP.
4. **Đảm bảo chức năng cốt lõi:** Tập trung vào việc triển khai chính xác và đầy đủ các chức năng hệ thống tệp cơ bản đã đề ra trong yêu cầu chức năng.
5. **Ưu tiên tính bảo mật và đơn giản:** Thiết kế cần ưu tiên việc sử dụng các tính năng bảo mật của SSH và giữ cho kiến trúc tổng thể đơn giản, dễ hiểu và dễ bảo trì, thay vì cố gắng tích hợp các tính năng phức tạp hoặc tối ưu hóa hiệu năng quá sớm.

3.3. Phạm vi dự án

Để đảm bảo tính khả thi và tập trung vào các mục tiêu cốt lõi, phạm vi của dự án RemoteFS được xác định như sau:

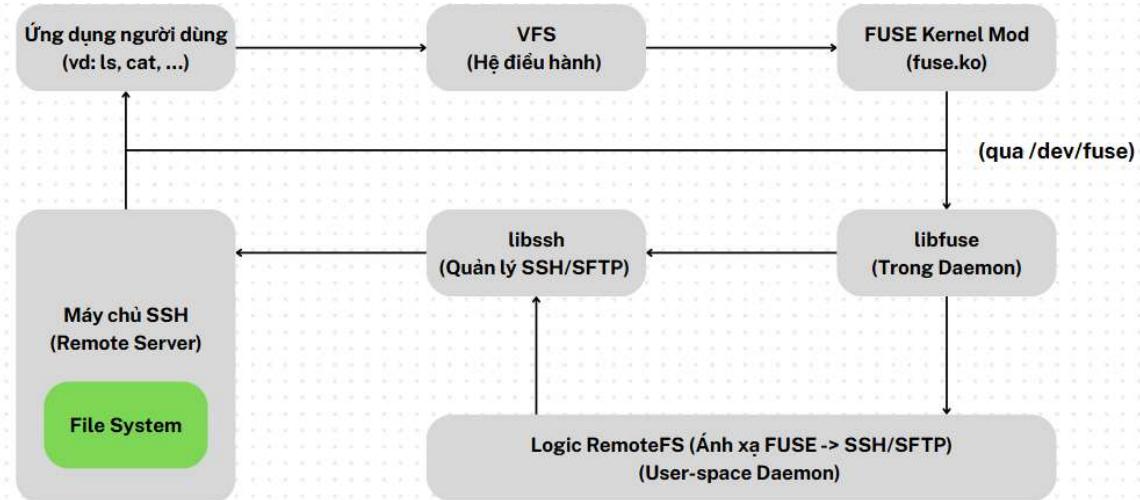
- **Bao gồm (In Scope):**
 - Triển khai các hàm callback FUSE cần thiết cho các thao tác hệ thống tệp cơ bản: getattr, readdir, open, read, write, release, mkdir, rmdir, unlink, create, statfs. Có thể xem xét rename.
 - Cho phép người dùng chỉ định thông tin kết nối SSH (host, user, port) và thư mục từ xa cần mount khi khởi chạy ứng dụng.
 - Sử dụng các phương thức xác thực SSH cơ bản được hỗ trợ bởi libssh (ví dụ: mật khẩu, có thể mở rộng public key).
 - Xử lý các lỗi cơ bản liên quan đến kết nối SSH và thao tác tệp từ xa.
 - Hoạt động trên môi trường Linux hỗ trợ FUSE.
- **Không bao gồm (Out of Scope):**

- Các tính năng hệ thống tệp nâng cao như liên kết tượng trưng (symbolic links), liên kết cứng (hard links), thuộc tính mở rộng (extended attributes), quản lý danh sách kiểm soát truy cập (ACLs), khóa tệp (file locking).
- Các cơ chế tối ưu hóa hiệu năng phύt tệp như bộ đệm dữ liệu (data caching) hoặc bộ đệm metadata (metadata caching) phía client.
- Hỗ trợ các giao thức truy cập từ xa khác ngoài SSH.
- Giao diện người dùng đồ họa (GUI) để quản lý việc mount.
- Xử lý tình huống lỗi mạng phύt tệp hoặc cơ chế tự động kết nối lại nâng cao.
- Quản lý quyền hạn (permissions) phύt tệp vượt ra ngoài những gì hệ thống tệp từ xa và SSH cung cấp mặc định.

CHƯƠNG 4: ỨNG DỤNG

4.1. Sơ đồ khái niệm ứng dụng

Để hình dung rõ hơn về luồng hoạt động của RemoteFS, chúng ta có thể mô tả qua sơ đồ khái niệm tương tác giữa các thành phần chính như sau:



- Ứng dụng người dùng:** Khởi tạo yêu cầu thao tác tệp (ví dụ: ls /mnt/remote).
- VFS (Virtual File System):** Tiếp nhận yêu cầu và chuyển đến module FUSE của kernel vì /mnt/remote là điểm mount FUSE.
- FUSE Kernel Module:** Nhận yêu cầu từ VFS và chuyển tiếp qua thiết bị /dev/fuse đến tiến trình RemoteFS daemon đang chạy trong user-space.
- libfuse:** Trong tiến trình daemon, libfuse đọc yêu cầu từ /dev/fuse và gọi hàm callback tương ứng đã được đăng ký trong logic của RemoteFS.
- Logic RemoteFS (Daemon):** Hàm callback được gọi (ví dụ: remote_readdir). Logic này sẽ:
 - Xác định hành động cần thực hiện trên máy chủ từ xa.
 - Sử dụng libssh để gửi lệnh SSH (ví dụ: ls -la remote_path) hoặc thực hiện thao tác SFTP tương ứng.
- libssh:** Thiết lập, quản lý kết nối SSH và thực thi lệnh hoặc thao tác SFTP được yêu cầu bởi logic RemoteFS.
- Máy chủ SSH:** Nhận yêu cầu qua kết nối SSH, thực thi lệnh hoặc thao tác SFTP trên hệ thống tệp từ xa.

- Kết quả từ máy chủ SSH được gửi trả lại qua libssh, Logic RemoteFS, libfuse, FUSE Kernel Module, VFS và cuối cùng đến ứng dụng người dùng ban đầu.

4.2. Kiến trúc và nguyên lý hoạt động

Về kiến trúc:

- Nhân chính (Core):** Tiến trình RemoteFS daemon được xây dựng dựa trên libfuse. Nó khởi tạo FUSE, đăng ký các hàm callback trong struct fuse_operations, và bắt đầu vòng lặp chính (fuse_main) để lắng nghe yêu cầu từ kernel.
- Module xử lý FUSE Operations:** Đây là phần cốt lõi, chứa việc triển khai các hàm callback (như remote_getattr, remote_readdir, remote_read, remote_write, v.v.). Mỗi hàm chịu trách nhiệm xử lý một loại yêu cầu VFS cụ thể.
- Module quản lý kết nối SSH:** Sử dụng libssh, module này chịu trách nhiệm thiết lập kết nối SSH ban đầu khi mount, quản lý phiên (session) SSH, xử lý xác thực, và đóng kết nối khi unmount. Nó cung cấp các hàm tiện ích để thực thi lệnh và thao tác SFTP qua kết nối đã thiết lập.
- Module ánh xạ và thực thi:** Module này chứa logic để chuyển đổi yêu cầu FUSE (với các tham số như đường dẫn, cờ, dữ liệu) thành các lệnh shell phù hợp hoặc các lời gọi hàm SFTP của libssh. Ví dụ, mkdir("/path/newdir") trong FUSE sẽ được ánh xạ thành việc thực thi lệnh mkdir /remote/path/newdir qua SSH. read("/path/file") sẽ được ánh xạ thành các thao tác mở tệp và đọc dữ liệu qua SFTP. Cấu trúc chi tiết của các module và cách chúng tương tác có thể được tham khảo trực tiếp từ mã nguồn của dự án.

Về nguyên lý hoạt động:

- Khởi tạo:** Khi người dùng chạy lệnh mount (ví dụ: remotesfs user@host:/remote/dir /mnt/remote), tiến trình RemoteFS daemon khởi động. Nó phân tích các tham số dòng lệnh, sử dụng libssh để thiết lập kết nối SSH đến user@host, và sau đó gọi fuse_main với cấu trúc fuse_operations đã định nghĩa và điểm mount /mnt/remote. fuse_main sẽ đăng ký filesystem với kernel và đi vào vòng lặp chờ yêu cầu.
- Xử lý yêu cầu:** Khi có thao tác trên /mnt/remote, luồng xử lý như mô tả ở mục 4.1 diễn ra. Bên trong daemon, libfuse gọi hàm callback tương ứng.
- Thực thi từ xa:** Hàm callback sử dụng module quản lý kết nối SSH để gửi yêu cầu đến server.
 - Đối với các thao tác metadata hoặc cấu trúc thư mục (như getattr, readdir, mkdir, rmdir, unlink, rename), việc thực thi lệnh shell tương ứng qua SSH (ví dụ: stat, ls, mkdir, rmdir, rm, mv) thường là cách tiếp cận đơn giản. Kết

quả trả về từ lệnh (stdout, stderr, exit code) sẽ được phân tích để tạo phản hồi cho FUSE.

- Đối với các thao tác đọc/ghi dữ liệu (read, write, create), sử dụng SFTP qua libssh thường hiệu quả và đáng tin cậy hơn. libssh cung cấp các hàm để mở tệp SFTP, đọc/ghi dữ liệu theo offset và kích thước, và đóng tệp.
4. **Trả kết quả:** Kết quả (thành công, lỗi, dữ liệu đọc được) được hàm callback trả về cho libfuse, sau đó đi ngược lại qua kernel để đến ứng dụng người dùng.

4.3 Quy trình phát triển

Quy trình phát triển ứng dụng RemoteFS tuân theo các bước cơ bản của việc phát triển phần mềm, kết hợp với các đặc thù của việc xây dựng FUSE filesystem:

1. **Thiết lập môi trường:** Chuẩn bị môi trường phát triển trên Linux, cài đặt các gói cần thiết bao gồm trình biên dịch C (GCC), công cụ build (Make), thư viện libfuse (và header files - thường là gói libfuse-dev hoặc tương tự), thư viện libssh (và header files - gói libssh-dev hoặc tương tự).
2. **Nghiên cứu và Thiết kế:** Nghiên cứu kỹ API của libfuse và libssh. Thiết kế cấu trúc cơ bản cho ứng dụng daemon, xác định các hàm callback FUSE cần triển khai và cách ánh xạ chúng sang các thao tác SSH/SFTP.
3. **Triển khai (Coding):**
 - Viết mã nguồn bằng ngôn ngữ C.
 - Triển khai khung sườn chính của ứng dụng (xử lý tham số dòng lệnh, gọi fuse_main).
 - Triển khai module quản lý kết nối SSH sử dụng libssh (kết nối, xác thực).
 - Triển khai từng hàm callback trong struct fuse_operations. Bắt đầu với các hàm cơ bản như getattr, readdir, sau đó đến open, read, write, mkdir, unlink, v.v..
 - Viết các hàm tiện ích để thực thi lệnh SSH và thao tác SFTP.
 - Xử lý lỗi cơ bản từ libfuse và libssh.
4. **Biên dịch:** Sử dụng gcc và make để biên dịch mã nguồn, liên kết với thư viện libfuse và libssh. Ví dụ lệnh biên dịch cơ bản: gcc remotefs.c -o remotefs \$(pkg-config --cflags --libs fuse libssh).
5. **Kiểm thử (Testing):** Đây là bước quan trọng và lặp đi lặp lại.

- Tạo một thư mục mount point cục bộ (ví dụ: mkdir /tmp/myremote).
 - Chạy RemoteFS để mount thư mục từ xa (ví dụ: ./remotefs user@host:/remote/path /tmp/myremote). Có thể chạy ở chế độ foreground và debug (-f -d) để xem log từ FUSE.
 - Sử dụng các lệnh Linux tiêu chuẩn để kiểm tra chức năng: ls -l /tmp/myremote, cd /tmp/myremote, cat /tmp/myremote/file.txt, echo "test" > /tmp/myremote/newfile.txt, mkdir /tmp/myremote/newdir, rm /tmp/myremote/newfile.txt, rmdir /tmp/myremote/newdir.
 - Kiểm tra các trường hợp biên (tệp lớn, tên tệp đặc biệt, thư mục rỗng/không rỗng, lỗi kết nối).
 - Unmount filesystem: fusermount -u /tmp/myremote.
6. **Gỡ lỗi (Debugging):** Sử dụng trình gỡ lỗi như gdb để phân tích lỗi và crash. Sử dụng các tùy chọn debug của FUSE (-d) và libssh (nếu có) để xem thông tin chi tiết về hoạt động bên trong. Phân tích mã nguồn và logic để tìm và sửa lỗi.
7. **Lặp lại:** Quay lại bước 3, 5, 6 để sửa lỗi, cải thiện hoặc thêm tính năng mới.

PHẦN 3: PHẦN KẾT LUẬN

3.1. Kết quả đạt được

Dự án đã thành công trong việc nghiên cứu và xây dựng RemoteFS, một hệ thống tệp hoạt động trong không gian người dùng dựa trên khuôn khổ FUSE và giao thức SSH. Kết quả chính đạt được bao gồm:

- Xây dựng thành công Daemon RemoteFS:** Đã phát triển được một ứng dụng daemon sử dụng libfuse và libssh có khả năng hoạt động như một cầu nối giữa hệ thống tệp cục bộ và thư mục trên máy chủ SSH từ xa.
- Chức năng Gắn kết (Mounting):** RemoteFS cho phép người dùng gắn kết một thư mục được chỉ định trên máy chủ SSH vào một điểm mount trên máy tính cục bộ, cung cấp một giao diện truy cập thống nhất.

```
[root@kali]~[~/Uni/System Programming/FinalTerm_Project/remote_proc_fuse]
# ./bin/remotefs ~/remote_fs_folder -o host=103.130.211.150 -o port=10033 -o user=sysadmin -o remotepath=/home/ute/test -o pass=vi
[INFO] src/main.c:223: Mounting remote filesystem:
[INFO] src/main.c:224:   Remote Host: 103.130.211.150
[INFO] src/main.c:225:   Remote Port: 10033
[INFO] src/main.c:226:   Remote User: sysadmin
[INFO] src/main.c:231:   Auth Method: Password [Provided]
[INFO] src/main.c:233:   Remote Path: /home/ute/test
[INFO] src/main.c:259:   Mount Point: /root/remote_fs_folder
[INFO] src/mount_config.c:92: Saved mount point: /root/remote_fs_folder -> /home/ute/test
[INFO] src/main.c:265: Saved mount point info for tools
```

```
[root@kali]~[~/Uni/System Programming/FinalTerm_Project/remote_proc_fuse]
# cd ~/remote_fs_folder

[root@kali]~/remote_fs_folder]
# ls
test.txt
```

- Hỗ trợ Thao tác Cơ bản:** Đã triển khai các hàm callback FUSE cần thiết để hỗ trợ các thao tác hệ thống tệp cơ bản, bao gồm:

- Liệt kê thư mục**

```
[root@kali]~/remote_fs_folder]
# ls -la
total 5
drwxr-xr-x  2 kali kali 4096 Apr 24 14:36 .
drwx----- 40 root root 4096 Apr 25 16:12 ..
-rw-----  1 kali kali    0 Apr 24 12:57 .abc.txt.swp
-rw-----  1 kali kali    0 Apr 24 12:57 .fuse_hidden0000000b00000001
-rw-----  1 kali kali    0 Apr 23 23:26 .fuse_hidden0000001f00000001
-rw-r--r--  1 kali kali    5 Apr 24 14:36 test.txt

[root@kali]~/remote_fs_folder]
# date
Fri Apr 25 04:18:50 PM +07 2025
```

```
sysadmin@vinhhat:/home/ute/test$ date
Fri 25 Apr 2025 09:18:00 AM UTC
sysadmin@vinhhat:/home/ute/test$ ls -la
total 12
drwxr-xr-x  2 sysadmin sysadmin 4096 Apr 24 07:36 .
drwxrwxrwx 10 ute      ute      4096 Apr 24 06:08 ..
-rw-----  1 sysadmin sysadmin     0 Apr 24 05:57 .abc.txt.swp
-rw-----  1 sysadmin sysadmin     0 Apr 24 05:57 .fuse_hidden0000000b00000001
-rw-----  1 sysadmin sysadmin     0 Apr 23 16:26 .fuse_hidden0000001f00000001
-rw-r--r--  1 sysadmin sysadmin    5 Apr 24 07:36 test.txt
sysadmin@vinhhat:/home/ute/test$ |
```

- Đọc dữ liệu

The image shows two terminal windows. The left window, titled '(root㉿kali)-[~/remote_fs_folder]', contains the command '# cat test.txt' followed by the output 'hhhh'. The right window, titled 'sysadmin@vinhhat:/home/ute/test\$', contains the command 'cat test.txt' followed by the output 'hhhh'. Both windows also show the date command being run.

```
(root㉿kali)-[~/remote_fs_folder]
# cat test.txt
hhhh

(sysadmin@vinhhat:/home/ute/test$ cat test.txt
hhhh
sysadmin@vinhhat:/home/ute/test$ date
Fri 25 Apr 2025 09:20:46 AM UTC
sysadmin@vinhhat:/home/ute/test$ |
```

- Ghi dữ liệu

The image shows two terminal windows. The left window, titled '(root㉿kali)-[~/remote_fs_folder]', contains the command '# echo '123456' > abc.txt' followed by the command '# ls' with outputs 'abc.txt' and 'test.txt'. The right window, titled 'sysadmin@vinhhat:/home/ute/test\$', shows the creation of 'abc.txt' with the content '123456', followed by date commands at different times.

```
(root㉿kali)-[~/remote_fs_folder]
# echo '123456' > abc.txt
(root㉿kali)-[~/remote_fs_folder]
# ls
abc.txt  test.txt

sysadmin@vinhhat:/home/ute/test$ ls
test.txt
sysadmin@vinhhat:/home/ute/test$ date
Fri 25 Apr 2025 09:23:17 AM UTC
sysadmin@vinhhat:/home/ute/test$ ls
abc.txt  test.txt
sysadmin@vinhhat:/home/ute/test$ date
Fri 25 Apr 2025 09:23:44 AM UTC
sysadmin@vinhhat:/home/ute/test$ cat abc.txt
123456
sysadmin@vinhhat:/home/ute/test$ date
Fri 25 Apr 2025 09:24:24 AM UTC
sysadmin@vinhhat:/home/ute/test$ |
```

- Xóa tệp

The image shows two terminal windows. The left window, titled '(root㉿kali)-[~/remote_fs_folder]', contains the command '# rm *.txt' followed by the command '# ls' which shows no files. The right window, titled 'sysadmin@vinhhat:/home/ute/test\$', shows the deletion of 'abc.txt' and 'test.txt', followed by date commands at different times.

```
(root㉿kali)-[~/remote_fs_folder]
# ls
abc.txt  test.txt
(root㉿kali)-[~/remote_fs_folder]
# rm *.txt
(root㉿kali)-[~/remote_fs_folder]
# ls
(root㉿kali)-[~/remote_fs_folder]
# |
```

```
sysadmin@vinhhat:/home/ute/test$ ls
abc.txt  test.txt
sysadmin@vinhhat:/home/ute/test$ date
Fri 25 Apr 2025 09:25:52 AM UTC
sysadmin@vinhhat:/home/ute/test$ ls
sysadmin@vinhhat:/home/ute/test$ date
Fri 25 Apr 2025 09:26:28 AM UTC
sysadmin@vinhhat:/home/ute/test$ |
```

- Tạo thư mục

```

└──(root㉿kali)-[~/remote_fs_folder]
    # ls
    new

└──(root㉿kali)-[~/remote_fs_folder]
    # mkdir new

└──(root㉿kali)-[~/remote_fs_folder]
    # ls
    new

```

```

sysadmin@vinhhat:/home/ute/test$ ls
sysadmin@vinhhat:/home/ute/test$ date
Fri 25 Apr 2025 09:27:30 AM UTC
sysadmin@vinhhat:/home/ute/test$ ls; date
new
Fri 25 Apr 2025 09:27:40 AM UTC
sysadmin@vinhhat:/home/ute/test$ |

```

o Xóa thư mục

```

└──(root㉿kali)-[~/remote_fs_folder]
    # ls
    new

└──(root㉿kali)-[~/remote_fs_folder]
    # rm -r new

└──(root㉿kali)-[~/remote_fs_folder]
    # ls
    # |_

```

```

sysadmin@vinhhat:/home/ute/test$ ls; date
new
Fri 25 Apr 2025 09:28:57 AM UTC
sysadmin@vinhhat:/home/ute/test$ ls; date
Fri 25 Apr 2025 09:29:11 AM UTC
sysadmin@vinhhat:/home/ute/test$ |

```

- Ánh xạ FUSE sang SSH/SFTP:** Các yêu cầu hệ thống tệp từ FUSE được ánh xạ một cách hiệu quả thành các lệnh thực thi từ xa qua SSH hoặc các thao tác truyền tệp qua SFTP, tùy thuộc vào bản chất của yêu cầu.
- Truy cập trong suốt:** Người dùng có thể tương tác với các tệp và thư mục từ xa thông qua các công cụ dòng lệnh (ls, cat, cp, mkdir...) hoặc trình quản lý tệp đồ họa như thê chúng là tài nguyên cục bộ.
- Kiểm thử xác nhận:** Quá trình kiểm thử thực nghiệm đã xác nhận rằng các chức năng cơ bản hoạt động đúng như mong đợi, cho phép thực hiện các thao tác quản lý tệp cơ bản trên thư mục đã mount.

Nhìn chung, dự án đã đạt được mục tiêu cốt lõi là tạo ra một hệ thống tệp FUSE đơn giản nhưng hoạt động được, sử dụng SSH để truy cập tệp từ xa một cách an toàn.

3.2. Ưu nhược điểm

Ưu điểm	Nhược điểm
Linh hoạt và Dễ phát triển: Phát triển trong user-space nhờ FUSE giúp đơn giản hóa, không cần sửa kernel	Hiệu năng hạn chế: Thường chậm hơn hệ thống tệp kernel do độ trễ mạng, chi phí mã hóa SSH, và chuyển đổi ngữ cảnh
Bảo mật cao: Tận dụng và kế thừa cơ chế xác thực và mã hóa mạnh mẽ của giao thức SSH	Phụ thuộc kết nối: Hoạt động phụ thuộc hoàn toàn vào sự ổn định của kết nối mạng và máy chủ SSH từ xa.

<p>Triển khai User-space: Thường không yêu cầu quyền root để chạy, tăng tính an toàn và tiện lợi.</p> <p>Truy cập liền mạch: Cung cấp trải nghiệm truy cập tệp từ xa tự nhiên qua các công cụ hệ thống tệp chuẩn.</p> <p>Tận dụng hạ tầng có sẵn: Dễ dàng triển khai vì hầu hết các máy chủ Linux/Unix đều có sẵn SSH server.</p>	<p>Thiếu tính năng nâng cao: Chưa hỗ trợ các tính năng như symlinks, hardlinks, file locking, extended attributes, ACLs</p> <p>Xử lý lỗi còn cơ bản: Cơ chế xử lý lỗi kết nối mạng hoặc lỗi từ server SSH có thể cần cải thiện để tăng độ ổn định.</p>
--	--

3.3. Hướng phát triển của đề tài

Để nâng cao tính hữu dụng và hiệu quả của RemoteFS, có một số hướng phát triển tiềm năng trong tương lai:

- **Cải thiện hiệu năng:**
 - **Triển khai Caching:** Xây dựng cơ chế cache phía client cho cả dữ liệu (data caching, read-ahead) và metadata (attribute caching, directory entry caching) để giảm thiểu số lượng yêu cầu gửi qua mạng SSH.
 - **Tối ưu hóa giao tiếp SSH/SFTP:** Nghiên cứu các kỹ thuật để giảm overhead khi giao tiếp qua SSH, ví dụ như sử dụng các kênh SFTP hiệu quả hơn, hoặc gộp các yêu cầu nhỏ khi có thể.
- **Bổ sung tính năng:**
 - **Hỗ trợ tính năng Filesystem nâng cao:** Triển khai các hàm callback FUSE còn thiếu để hỗ trợ symbolic links (symlink, readlink), hard links (link), extended attributes (setxattr, getxattr, listxattr), file locking (lock).
 - **Quản lý quyền hạn tốt hơn:** Cải thiện việc ánh xạ và xử lý quyền POSIX giữa hệ thống cục bộ và từ xa.
- **Mở rộng và Hoàn thiện:**
 - **Hỗ trợ phương thức xác thực khác:** Tích hợp hỗ trợ cho các phương thức xác thực SSH phổ biến khác như SSH agent, GSSAPI/Kerberos.
 - **Nâng cao khả năng chịu lỗi:** Cải thiện cơ chế xử lý lỗi mạng, tự động kết nối lại (nếu khả thi và an toàn).
 - **Đóng gói và phân phối:** Tạo các gói cài đặt (ví dụ: .deb, .rpm) hoặc sử dụng các trình quản lý gói để người dùng dễ dàng cài đặt và sử dụng.

- **Đo lường và So sánh:** Thực hiện các bài đo hiệu năng chi tiết và so sánh RemoteFS với các giải pháp tương tự như SSHFS để có đánh giá khách quan hơn.

Những cải tiến này sẽ giúp RemoteFS trở thành một công cụ mạnh mẽ và đáng tin cậy hơn cho việc truy cập và quản lý tệp tin từ xa.

TÀI LIỆU THAM KHẢO

- [1] "FUSE," The Linux Kernel documentation. [Online]. Available: <https://docs.kernel.org/filesystems/fuse.html>.
- [2] "Filesystem in Userspace," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Filesystem_in_Userspace.
- [3] "libfuse/doc/kernel.txt at master," GitHub. [Online]. Available: <https://github.com/libfuse/libfuse/blob/master/doc/kernel.txt>.
- [4] "User Space Storage System Stack Modules with File Level Control," University of Connecticut SNSL. [Online]. Available: <https://snsl.engr.uconn.edu/wp-content/uploads/sites/2477/2018/06/ols10.pdf>.
- [5] "hinshun/hellofs: example fuse filesystem using containerd's mount," GitHub. [Online]. Available: <https://github.com/hinshun/hellofs>.
- [6] "rfjakob/the-fuse-wire-protocol: Document describing the filesystem in userspace wire protocol," GitHub. [Online]. Available: <https://github.com/rfjakob/the-fuse-wire-protocol>.
- [7] "libfuse/libfuse: The reference implementation of the Linux FUSE (Filesystem in Userspace) interface," GitHub. [Online]. Available: <https://github.com/libfuse/libfuse>.
- [8] "libfuse API documentation," libfuse. [Online]. Available: <https://libfuse.github.io/doxygen/>.
- [9] "libfuse - macFUSE user space library," GitHub. [Online]. Available: <https://github.com/macfuse/library>.
- [10] "FUSE (Filesystem in Userspace)," SNU Systems Software & Architecture Laboratory. [Online]. Available: <http://csl.snu.ac.kr/courses/4190.568/2024-1/fuse-tutorial.pdf>.
- [11] "RFUSE: Modernizing Userspace Filesystem Framework through Scalable Kernel-Userspace Communication," USENIX. [Online]. Available: <https://www.usenix.org/system/files/fast24-cho.pdf>.

- [12] "FUSE," GlusterFS documentation. [Online]. Available: <https://glusterdocs-beta.readthedocs.io/en/latest/overview-concepts/fuse.html>.
- [13] "Overview of the Linux Virtual File System," The Linux Kernel documentation. [Online]. Available: <https://www.kernel.org/doc/html/next/filesystems/vfs.html>.
- [14] "Filesystems in the Linux kernel," The Linux Kernel documentation. [Online]. Available: <https://docs.kernel.org/filesystems/index.html>.
- [15] "ECE566 Enterprise Storage Architecture Program: Intro to FUSE," Duke People. [Online]. Available: <https://people.duke.edu/~tkb13/courses/ece566-2020fa/homework/program-fuse.pdf>.
- [16] "Fuse I/O Modes," The Linux Kernel Archives. [Online]. Available: <https://www.kernel.org/doc/html/v5.9/filesystems/fuse-io.html>.
- [17] "libfuse," GitHub. [Online]. Available: <https://github.com/libfuse>.
- [18] "libfuse/sshfs: A network filesystem client to connect to SSH servers," GitHub. [Online]. Available: <https://github.com/libfuse/sshfs>.
- [19] "Using FUSE as an VFS," OSDev.org. [Online]. Available: <https://forum.osdev.org/viewtopic.php?t=41827>.
- [20] "FUSE (Filesystem in Userspace) on OpenSolaris | PPT," SlideShare. [Online]. Available: <https://www.slideshare.net/adorepump/fuse-filesystem-in-userspace-on-opensolaris>.
- [21] "Filesystems · libfuse/libfuse Wiki," GitHub. [Online]. Available: <https://github.com/libfuse/libfuse/wiki/filesystems>.
- [22] "Internals," JuiceFS Document Center. [Online]. Available: <https://juicefs.com/docs/community/internals/>.
- [23] "Features of data processing and storage using the virtual file system," ResearchGate. [Online]. Available: https://www.researchgate.net/publication/388979163_Features_of_data_processing_and_storage_using_the_virtual_file_system.
- [24] "XFUSE: An Infrastructure for Running Filesystem Services in User Space," USENIX. [Online]. Available: <https://www.usenix.org/system/files/atc21-huai.pdf>.
- [25] "libfuse repositories," GitHub. [Online]. Available: <https://github.com/orgs/libfuse/repositories>.
- [26] "FUSE passthrough," Android Open Source Project. [Online]. Available: <https://source.android.com/docs/core/storage/fuse-passthrough>.

- [27] "libfuse/example/hello.c at master," GitHub. [Online]. Available: <https://github.com/libfuse/libfuse/blob/master/example/hello.c>.
- [28] "Gcsfuse: A user-space file system for interacting with Google Cloud Storage," Hacker News. [Online]. Available: <https://news.ycombinator.com/item?id=37403074>.
- [29] The Linux Kernel documentation. [Online]. Available: <https://docs.kernel.org/>.
- [30] The Linux Kernel documentation. [Online]. Available: <https://www.kernel.org/doc/html/v5.8/>.
- [31] The Linux Kernel documentation. [Online]. Available: <https://www.kernel.org/doc/html/v5.9/>.
- [32] "The /proc Filesystem," The Linux Kernel documentation. [Online]. Available: <https://docs.kernel.org/filesystems/proc.html>.
- [33] "proc file system documentation," The Linux Kernel Archives. [Online]. Available: <https://www.kernel.org/doc/Documentation/filesystems/proc.txt>.
- [34] "devices.txt," The Linux Kernel Archives. [Online]. Available: <https://www.kernel.org/doc/Documentation/admin-guide/devices.txt>.
- [35] libfuse. [Online]. Available: <https://libfuse.github.io/>.
- [36] "Filesystem Architecture," Fuchsia. [Online]. Available: <https://fuchsia.dev/fuchsia-src/concepts/filesystems/filesystems>.
- [37] M. Veselý, "Virtual file system in user space," Charles University DSpace. [Online]. Available: <https://dspace.cuni.cz/bitstream/handle/20.500.11956/183081/130360154.pdf?sequence=1&isAllowed=y>.
- [38] "CSCE604227 System Programming CSCE604227 Pemrograman Sistem Week 03: FUSE: Filesystem in Userspace," VLSM. [Online]. Available: <https://docs.vlsm.org/SPSlides/sp03.pdf>.