Mini-Rsync – Detailed System Explanation

Overview

This document explains the architecture, components, and workflow behind the Mini■Rsync project. It is designed for interviews and resume discussion.

1. Architecture

The system consists of:

- Sync Server: Receives files, writes safely via temporary files, atomic rename.

- Sync Client: Watches a directory, computes SHA-256, sends files to server.

- Thread Pool: Allows parallel file transfers.

- Watcher: Detects file changes via inotify (Linux).

- Networking: TCP-based custom protocol.

- Integrity: SHA-256 to detect real modifications.

2. Workflow

1. Client starts and scans directory.

2. For each file, SHA-256 is computed.

3. Client connects to the server using TCP.

4. Sends header: filename length → filename → file size.

5. Sends file content in 8KB chunks.

6. Server writes content to .tmp.

7. After transfer, server renames tmp → final.

8. Logs event.

3. Key Concepts Demonstrated

- POSIX networking (socket, connect, bind, listen, accept)

- File handling (open, read, write)

- Atomic operations (rename)

- Thread synchronization (mutex, condition_variable)
- Parallel processing with thread pools
- Linux event-based filesystem monitoring (inotify)
- Hashing for integrity validation

4. What Was Added in Enhancement Stage (Requested Features)

- Clean folder structure + Makefile
- SHA-256 hashing via OpenSSL
- Thread pool optimization
- Robust README
- Educational code comments
- Server-side safe write flow

Next Improvements Possible:

- TLS using OpenSSL
- File delta transfer
- Resume support
- Compression using zlib

This PDF is a beginner-friendly explanation of the project for resume and interview use.