Report: Client-Server File Management System

ÇAĞRI YILDIZ 1901042630

CSE_344_MIDTERM

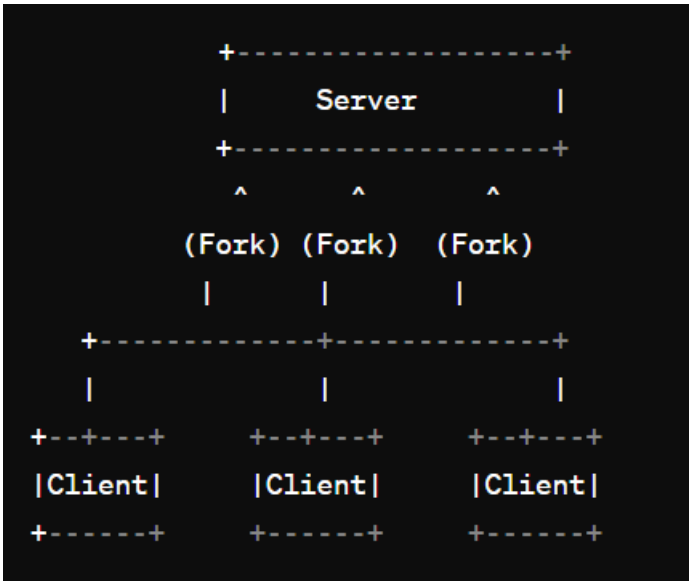
Introduction

The project presents a client-server file management system that allows multiple clients to communicate with a server through named pipes (FIFOs). Clients can execute various commands on the server, such as listing directory contents, reading and writing files, uploading/downloading files, and archiving the server directory. This system showcases fundamental concepts like Inter-Process Communication (IPC), concurrent client handling using forked processes, and resource synchronization through semaphores.

System Architecture

- **Server:** A central server manages all client requests via a named FIFO. It uses semaphores for synchronization and handles clients concurrently via forked child processes.
- **Client**: Each client has a unique FIFO, through which it communicates with the server. Clients submit commands to the server, and read the server's responses via their private FIFO.
- **Named Pipes (FIFOs):**
  Server FIFO: Receives client connection requests.
  Client FIFO: Unique for each client, used to read responses

Diagram:



Design Decisions

- **Client Connection Management:**
  The server dynamically accepts up to a specified number of clients.
  Each client is given a unique FIFO based on its PID for private communication.
- **Concurrency and Synchronization:**
  Concurrency: Each client request is handled by a separate forked process.
  Synchronization: Semaphores are used to protect critical sections involving file operations and client count management.
- **Command Processing:**
  The server can't processes commands like list, readF, writeT, etc., and sends appropriate responses back to clients.
  Command execution is modularized into individual functions for ease of maintenance.
- **Error Handling and Cleanup:**
  Signal handlers ensure graceful server shutdown.
  FIFO files are unlinked and semaphores are destroyed during cleanup.

Implementation Details

- **Server Implementation (server.c):**
  Initializes semaphores and sets up a signal handler for cleanup.
  Creates a server FIFO and listens for client connection requests.
  Forks a new process for each client to handle commands concurrently.
  Supported commands include (but won't work):
  list: Lists all files in the directory.
  readF: Reads a specific line or the entire file.
  writeT: Writes to a specific line or appends to a file.
  upload: Receives a file from the client.
  download: Sends a file to the client.
  archServer: Creates a tar archive of the server's directory.
  killServer: Gracefully shuts down the server.
- **Client Implementation (client.c):**
  Creates a private FIFO and notifies the server via the server FIFO.
  Sends commands to the server and reads responses through its private FIFO.
  Supported commands include all those processed by the server.

- **Makefile:**
  Provides targets to compile both the server and the client.
  Includes a clean target to remove compiled binaries.