CSE 344 System Programming
Midterm Report

Ömer Kaan Uslu

1801042642

Introduction

In this project, I designed a server client file management system which allows clients to perform various file operations such as listing files, reading files, writing to files, uploading files, and downloading files. The system consists of a server program and multiple client programs that communicate using a named pipe for interprocess communication.

System Architecture

The system follows a client-server architecture, where multiple client programs connect to a central server program. The server program handles client connections, receives commands from clients, performs the requested file operations, and sends the appropriate responses back to the clients. Each client program can independently execute commands and interact with the server.

Design Decisions

Communication Protocol: The named pipe (FIFO) was chosen as the communication protocol between clients and the server.

Signal Handling: Both the server and client programs implement signal handlers for SIGTERM and SIGINT signals. These signal handlers gracefully handle termination requests and perform operations before exiting.

Shared Resource Access and Synchronization: To ensure mutual exclusion and prevent concurrent access to shared resources (such as the log file and critical sections), a semaphore mechanism is implemented. The semaphore guarantees that only one client can access the critical section at a time, preventing race conditions and conflicts.

File Operations: The system supports various file operations, including listing files, reading files, writing to files, uploading files, and downloading files. The commands are sent by clients to the server, which processes them accordingly and provides appropriate responses.

Implementation Details

Server

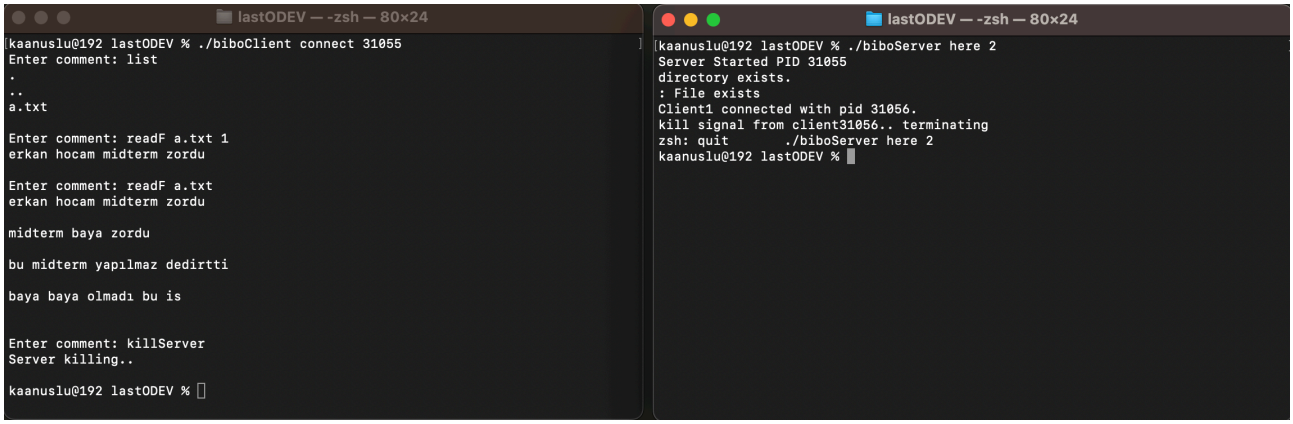
The server program does handle multiple client connections concurrently and processing commands to send back response.
It listens for client connection from fifo at first.
When a new client comes, program creates a new process with fork to handle clients concurrently.
Each client process id is used for creating a new fifo to communicate with client.
The *serveClient* function is used to serve client after those operations. This method serves client to get clients requests and send response them seperately for each command. In this method, there is *sem_post* and *sem_wait* calls to guard synchronization. The semaphore ensures that only one client can access the critical section at a time, avoiding conflicts and data corruption.
Server log file is created and the server completes its lifetime.

Client

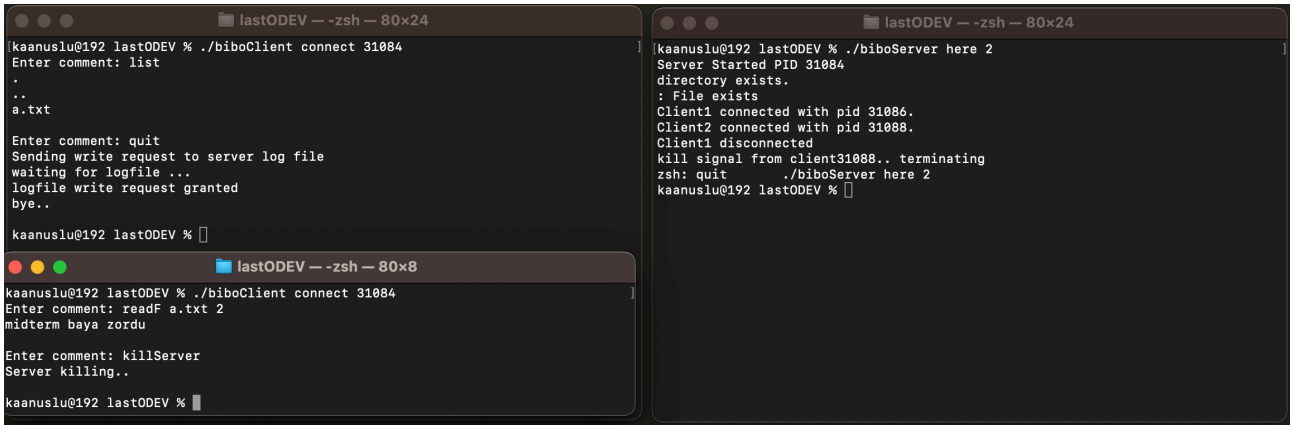
The client program does handle communication with server. It does open fifo, and sends its process id to server.
Program takes command input and sends it to server without doing anything about this string. The response directly comes from server without parsing it on client side etc.
After sending the command, client listens for response and displays the response.
If killServer or quit commands entered, the while loop ends and program sends this commands to server in order to process this commands in server side.

Test Results

With one client:



With two client:



With three client: