# CSE344 System Programming-Midterm Project Report

Yağız Hakkı Aydın
1901042612

## Client Side

1. The client code begins by setting up the necessary variables and functions, including including the required libraries, defining buffer sizes, and declaring file descriptors and paths for the client and server FIFOs.

2. The `checkClientProgramArguments()` function is used to validate the command-line arguments provided to the client program. It checks the number and format of arguments to ensure they are valid.

3. The `handleCtrlC()` function is a signal handler that is triggered when the client receives a kill signal (SIGINT), typically when the user presses Ctrl+C. It sets the buffer to "killClient" to notify the server about the client's intention to terminate.

4. The `main()` function is the entry point for the client code. It sets up the signal handler, gets the client's current path, opens the server FIFO, and sends the client's path and PID to the server for identification.

5. Within the main loop, the client waits for incoming responses from the server. If there are any responses, they are printed to the console. If the response indicates that the server has forked with a new server FIFO, the client updates its server FIFO path and opens the new FIFO.

6. If there are no incoming responses, the client prompts the user for a command. The `handleClientComment()` function takes the user's input and processes it based on the command. The available commands include "help," "list," "download," "upload," "killServer," "quit," and "readF."

7. For commands such as "list," "download," "upload," "killServer," and "quit," the client prepares the corresponding request by populating the buffer and writes it to the server FIFO.

8. The "readF" command requires additional parameters, such as the filename and line number. The client extracts these parameters, populates the buffer with the command and parameters, and writes it to the server FIFO.

9. The `printHelpComment()` function handles the "help" command and displays information about the available commands and their usage.

10. The `connectServer()` function is responsible for establishing the client FIFO and connecting to the server. It generates a unique client FIFO path based on the client's PID, creates the FIFO using `mkfifo()`, and opens the FIFO for reading.

11. The `setServerFifoPath()` function sets the server FIFO path based on the server's PID.

12. Finally, the client continues the main loop, alternating between waiting for server responses and prompting the user for commands until it receives a termination signal or the program is manually exited.

# Server Side

1. The server code begins by setting up the necessary variables and functions, including including the required libraries, defining buffer sizes, and declaring file descriptors and paths for the server FIFO and client FIFOs.

2. The `handleSignal()` function is a signal handler that is triggered when the server receives a kill signal (SIGINT or SIGTERM). It cleans up resources, closes file descriptors, and removes the server FIFO.

3. The `main()` function is the entry point for the server code. It sets up the signal handler, creates the server FIFO using `mkfifo()`, and opens the FIFO for reading.

4. Within the main loop, the server waits for incoming requests from clients. If there are any requests, they are processed by the `handleClientRequest()` function.

5. The `handleClientRequest()` function reads the request from the client FIFO, parses the command and parameters, and performs the corresponding actions based on the command. This includes handling commands such as "help," "list," "download," "upload," "killServer," "quit," and "readF."

6. For commands such as "help," "list," "download," "upload," and "killServer," the server prepares the response by populating the buffer and writes it to the client FIFO.

7. The "readF" command requires additional parameters, such as the filename and line number. The server retrieves the requested file, reads the specified line, and prepares the response with the content of the line. It then writes the response to the client FIFO.

8. The `sendResponse()` function handles writing the response to the client FIFO. It ensures that the entire response is written by repeatedly calling `write()` until all the data has been sent.

9. The `extractClientPID()` function extracts the client's PID from the received request. It is used for identification and communication between the server and client.

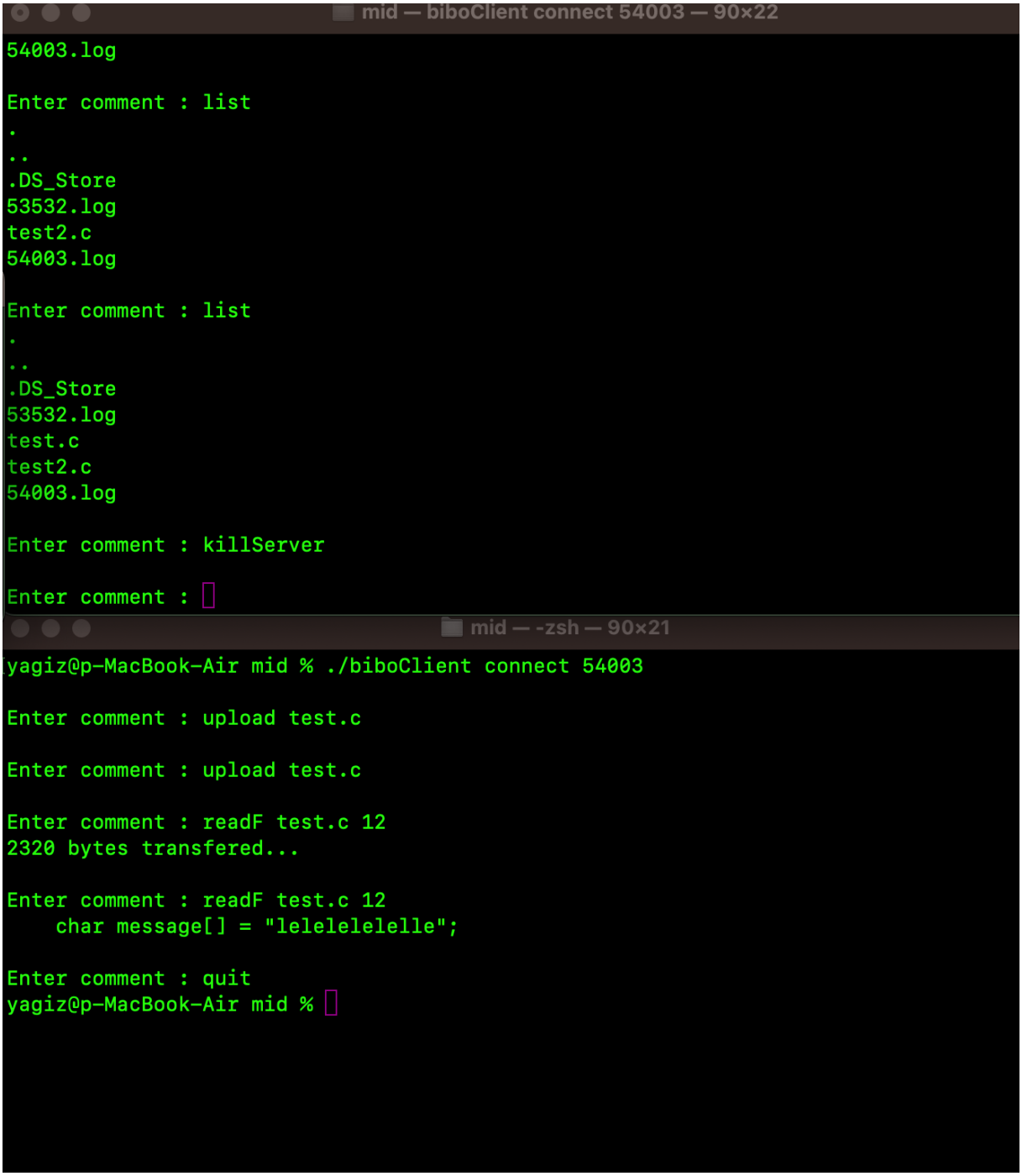
10. The `generateServerFifoPath()` function generates a unique server FIFO path based on the server's PID. This allows the server to create a new FIFO when a client requests forking.

11. The server continues the main loop, waiting for incoming requests from clients, until it receives a termination signal or the program is manually exited.

12. When the server receives a termination signal, the `handleSignal()` function is called to clean up resources and exit gracefully.

# Test Screenshot


mid — -zsh — 88×12
yagiz@p-MacBook-Air mid % ./biboServer /Users/yagiz/desktop/server 12
Server started PID 54003
Client PID 54005 connected as "client01"
Client PID 54007 connected as "client01"
s---> /Users/yagiz/Desktop/mid/test.c
d---> /Users/yagiz/desktop/server/test.c
2320 bytes transfered...
line>     char message[] = "lelelelelelle";
line>     char message[] = "lelelelelelle";
Client with PID 54007 quits...Server Killed by client




From those screenshots,it can be seen that two clients are connected at the same time and perform user commands.No download tested because it's not necessary to show;because it uses exact same file transfer function to perform.