Esra Eryılmaz
171044046

CSE 344 – System Programming

Midterm Project Report

16.05.2023

General Information

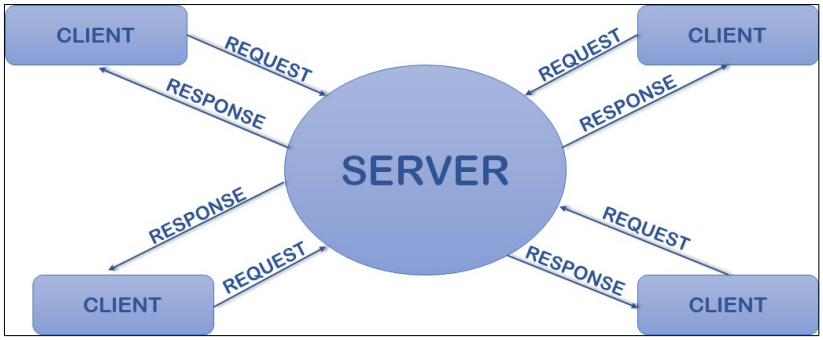
- The implementation of upload and download commands are missing. All commands are received from the client and sent to the server, but I ran out of time to implement them.
- I couldn't handle signals like Ctrl+C. (You can see the structure in the code, but I commented it out because it didn't work.) However, the "kill" request from the client is successful.
- In the server, I keep a separate temp for each connected client. But, when a client disconnects, the temps can get mixed up, and sometimes it can mistakenly display the number of a client that has disconnected. But that's not the case, you can see it from the client PIDs.
- Every time a client connects, I open a log file with the client's PID and also print the PID on the screen to ensure control.
- In the server log, I keep the server PID when the server is started. This allows clients to read it and connect.
- The "Connect/tryConnect" variable that I receive during client connection doesn't actually have any function. If the server PID is correct, it connects; otherwise, it doesn't connect.
- I try to make lots of error check in the assignment.
- I tried my best but however there are still places in the code where it may fail.

System Architecture

The system consists of two main components: a server and clients.

The server provides a service that manages access to files for multiple clients and handles their requests.

The client is a program that sends file operation requests to the server and receives responses from the server.



- **Server Architecture:**

  - The server starts in the main function.

  - The server has a signal handler. (but not working)

  - The server creates and opens a server FIFO (named pipe) for communication with clients.

  - The server defines variables necessary to manage a set of FIFOs and file operations.

  - The server enters a loop waiting for client connections.

  - Each client connection, the server creates a child process and hands over the client connection to this process.

  - The child process reads and processes client requests, and sends responses back to the client.

  - The server cleans up the server FIFO and other resources, and continues the loop.

  - The server stops accepting new connections when a user specified maximum number of clients is reached.

- **Client Architecture:**

  - The client is a program that connects to the server and sends requests for file operations.

  - The client receives user commands and uses a FIFO to communicate with the server.

  - The client parses user input to generate requests and sends them to the server.

  - The client receives responses from the server and prints them to the screen or writes them to files.

  - The client allows the user to terminate the communication with the server using special commands like "quit" or "killServer".

Design Decisions and Implementation Details

- I had created separate functions for everything while designing, which made it look organized. However, I couldn't communicate sometimes with the server . But when I did everything in the main function, there was no problem. Therefore, the readability of the code became a bit low.

- I have used some inter-process communication methods, mainly FIFOs and pipes. Although there are signals available, as I mentioned before, they do not function properly. However, there are "quit" and "kill" requests.

- **header.h**

  - It contains #define statements to define constants such as buffer sizes or maximum number of clients.

  - It includes struct definitions for data structures used in the client - server communication, such as message formats.

- **biboServer.c**

  - The server program starts by creating a server FIFO using the mkfifo system call.

  - It opens the server FIFO using the open system call, enabling it to read client requests.

  - The server initialize a counter variable, such as num_clients, to keep track of the number of connected clients.

  - It uses loop to continuously accept client connections by forking child processes for each client.

  - Inside the child process, the server opens the client FIFO using the open system call.

  - The server reads the client's request from the client FIFO and processes it accordingly.

  - It performs file operations or other tasks based on the received request.

  - The server then writes the response to the client FIFO using the write system call.

  - The server handles multiple client requests in a loop until it receives a termination signal or reaches a client limit.

- **biboClient.c**

  - The client program likely starts by creating a FIFO for receiving responses from the server, using the mkfifo system call.

  - It then opens the server FIFO using the open system call, allowing the client to write requests to the server.

  - To send a request, the client use the write system call to write data to the server FIFO.

  - After sending the request, the client reads the response from its specific FIFO(created with client's PID) using the open and read system calls.

  - The client may have a loop that allows it to send multiple requests and receive corresponding responses until it decides to stop or terminate.

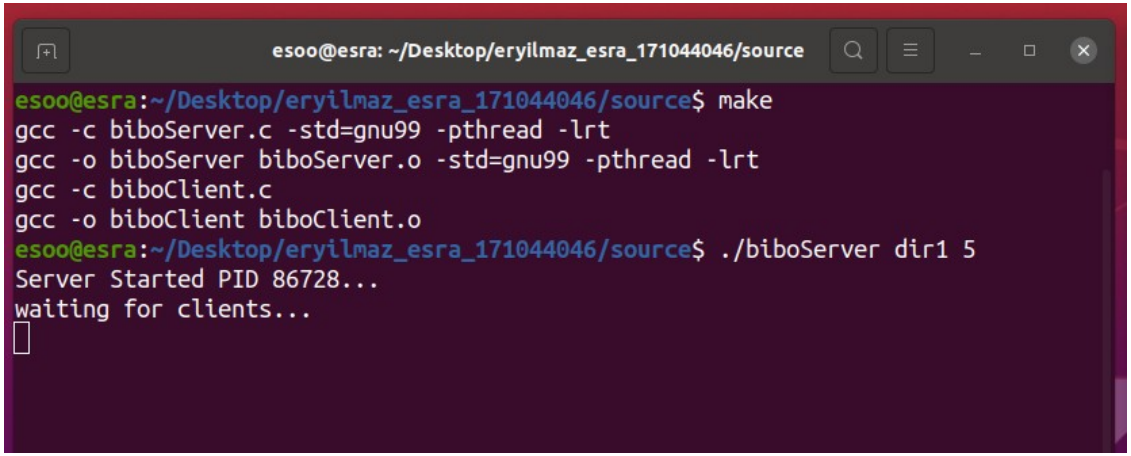
- **makefile**

  - It compiles the project as a whole.

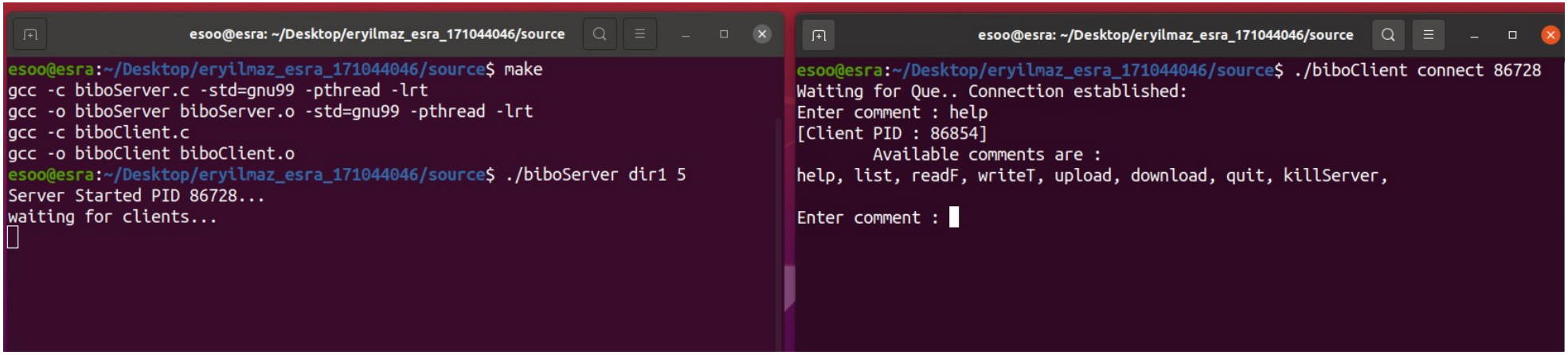
Tests

- Usage :

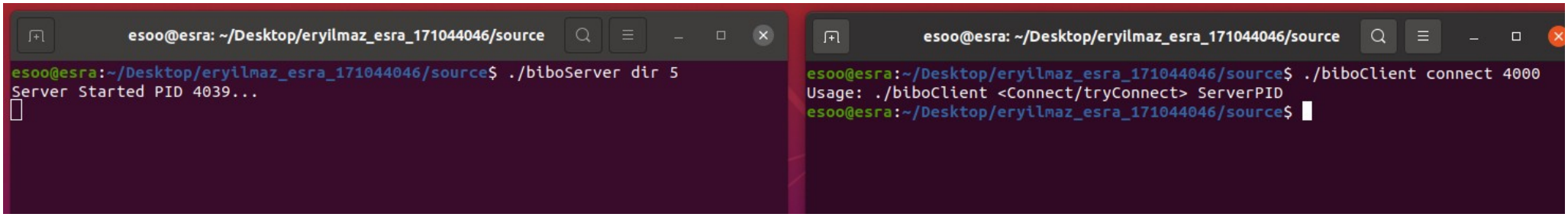
Firstly we start our server.



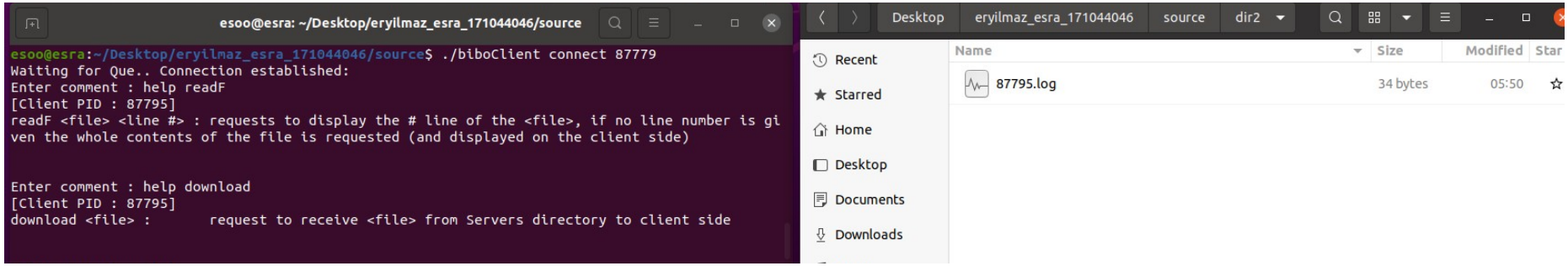
We can connect client to the server.



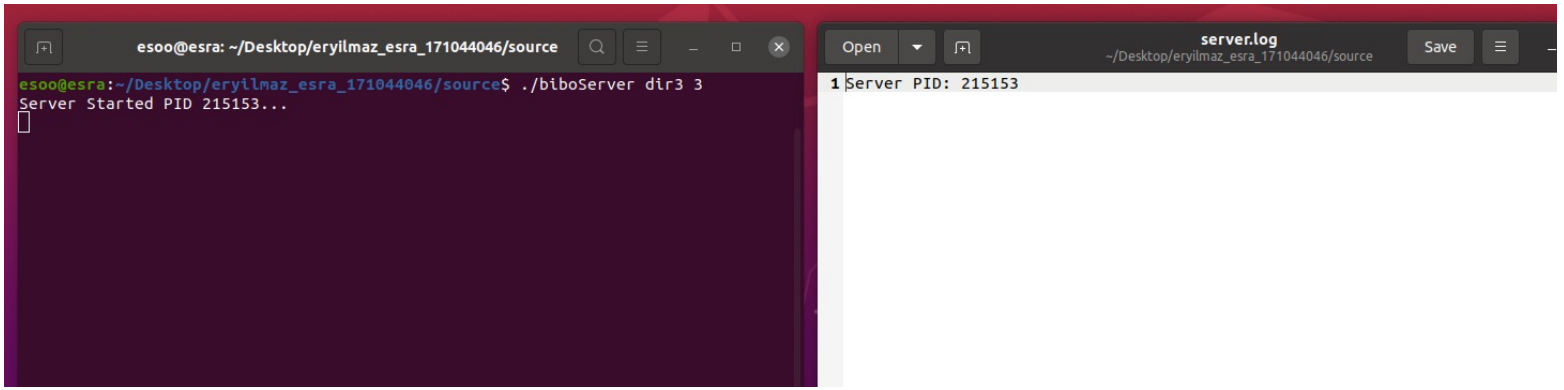
The client should enter the correct server PID



A log file is created when the client connects and the name of the created log file is the client's PID.

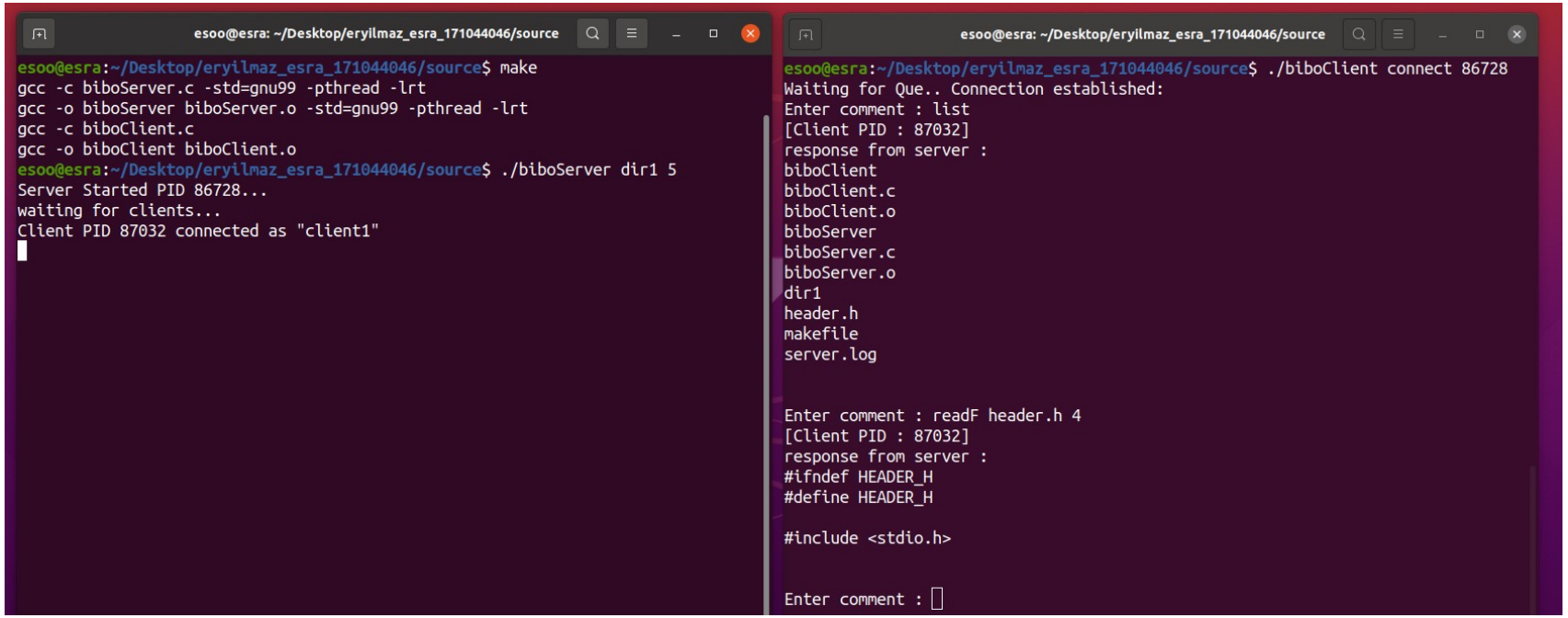


Server log

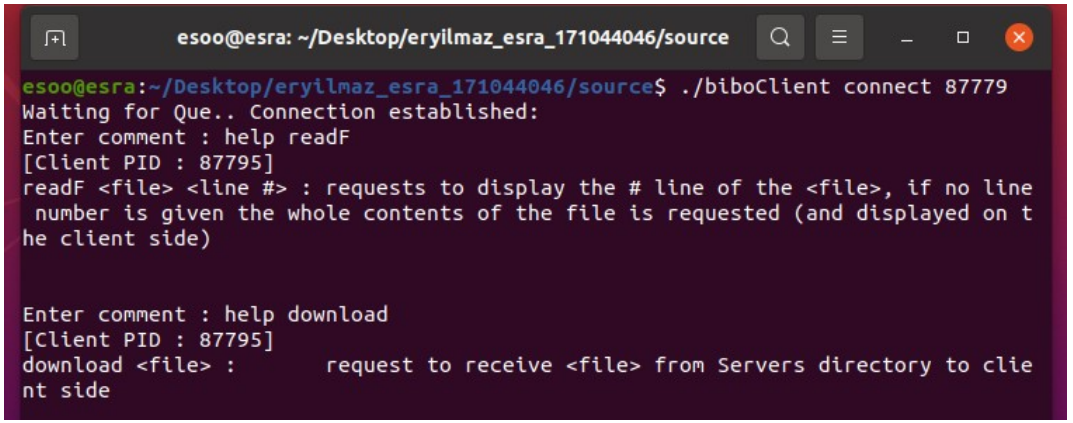


List

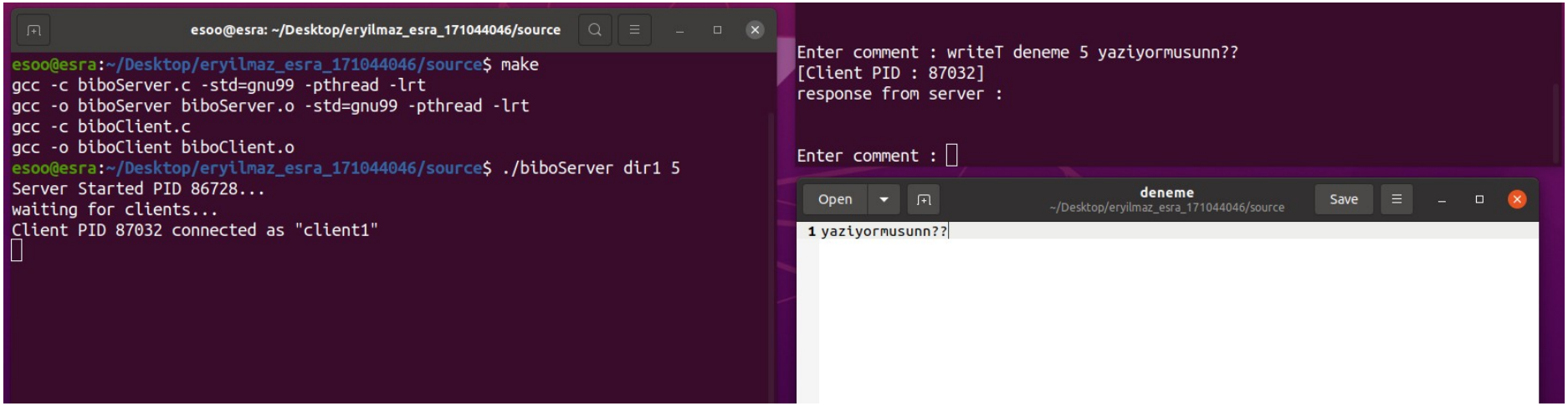
ReadF



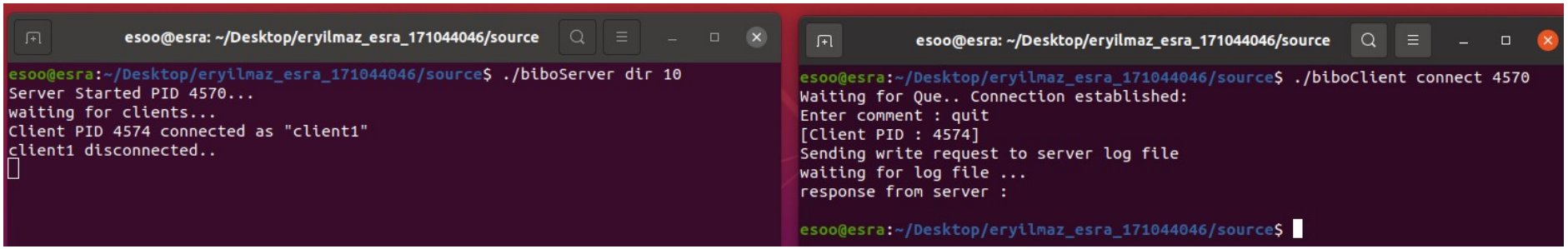
Help


esoo@esra: ~/Desktop/eryilmaz_esra_171044046/source
esoo@esra:~/Desktop/eryilmaz_esra_171044046/source$ ./biboClient connect 87779
Waiting for Que.. Connection established:
Enter comment : help readF
[Client PID : 87795]
readF <file> <line #> : requests to display the # line of the <file>, if no line
number is given the whole contents of the file is requested (and displayed on t
he client side)
Enter comment : help download
[Client PID : 87795]
download <file> : request to receive <file> from Servers directory to clie
nt side

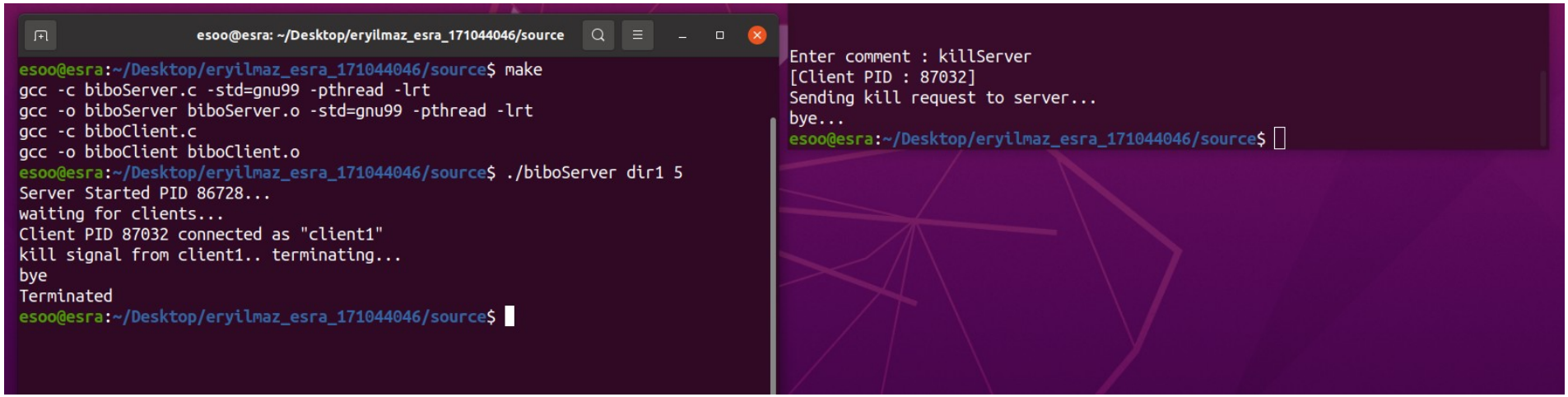
Write


esoo@esra: ~/Desktop/eryilmaz_esra_171044046/source
esoo@esra:~/Desktop/eryilmaz_esra_171044046/source$ make
gcc -c biboServer.c -std=gnu99 -pthread -lrt
gcc -o biboServer biboServer.o -std=gnu99 -pthread -lrt
gcc -c biboClient.c
gcc -o biboClient biboClient.o
esoo@esra:~/Desktop/eryilmaz_esra_171044046/source$ ./biboServer dir1 5
Server Started PID 86728...
waiting for clients...
Client PID 87032 connected as "client1"
Enter comment : writeT deneme 5 yaziyormusunn??
[Client PID : 87032]
response from server :
Enter comment :
Open
deneme
~/Desktop/eryilmaz_esra_171044046/source
Save
1 yaziyormusunn??

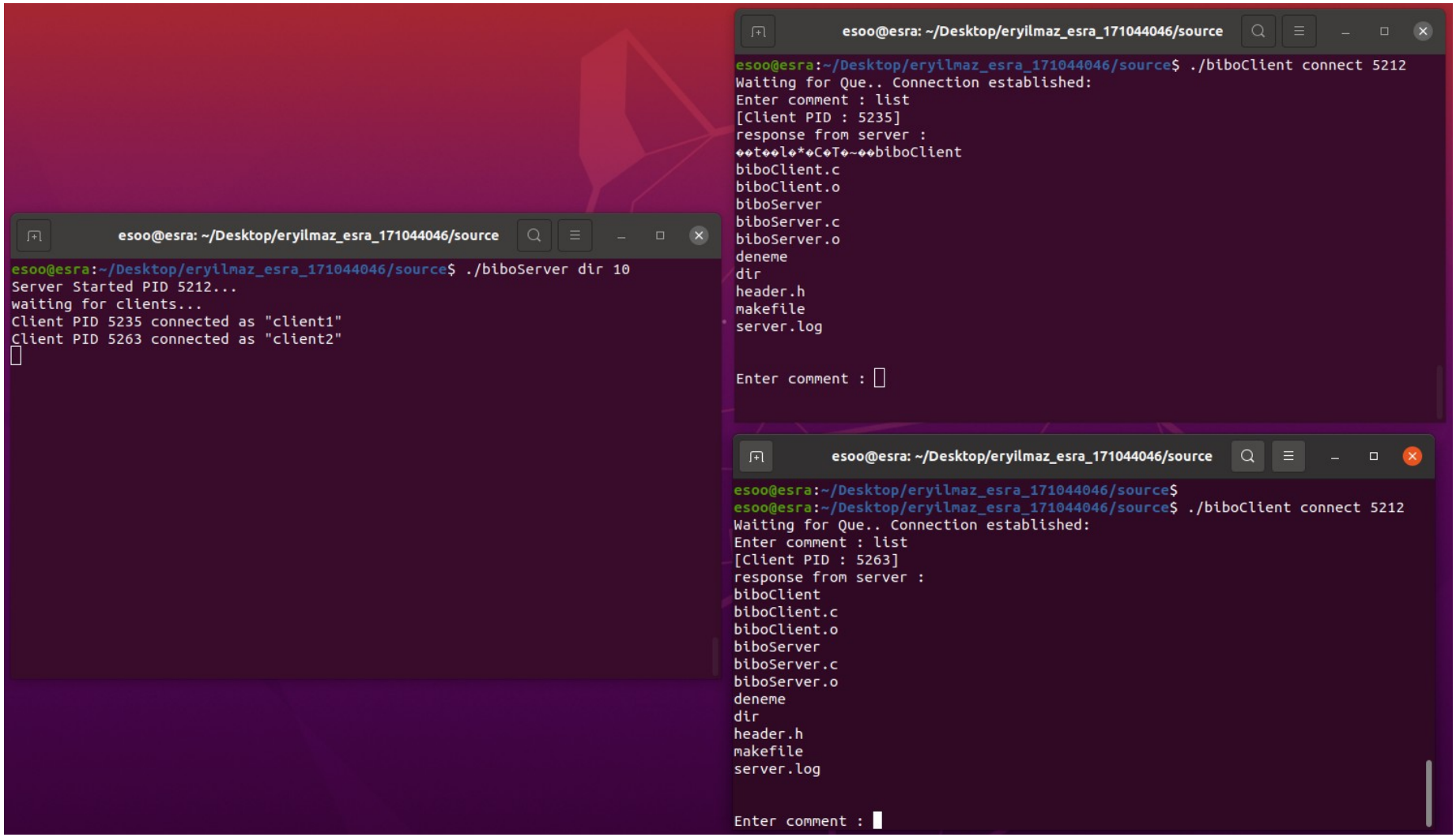
Quit



KillServer



Connecting multiple clients



It may not always works properly. For example :

It didn’t give any response.