

# **Gebze Technical University**

## **CSE-244 System Programming**

### **2024, Spring**

Midterm Project

Ahmet Yiğit - 200104004066

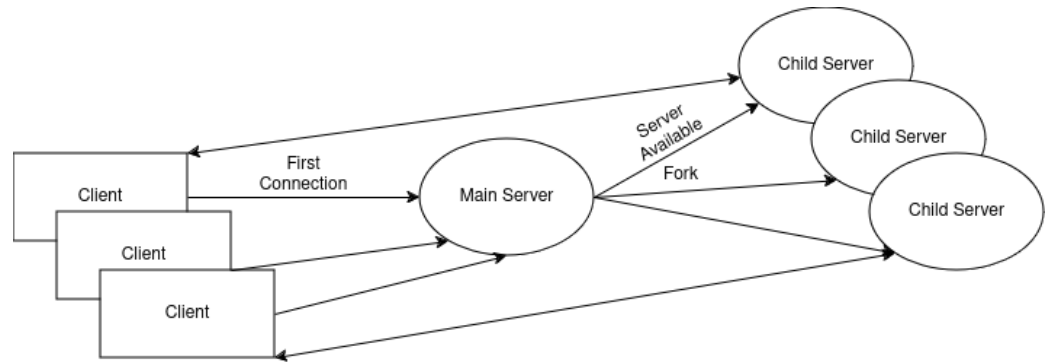
---

## **Introduction**

- a. The server-client architecture employed in this project follows a classic paradigm where the server acts as a centralized entity responsible for managing and servicing multiple client connections. Clients communicate with the server by sending requests, which are then processed by the server, and appropriate responses are sent back to the clients. This model enables distributed applications to leverage the computational power and resources of the server while providing a platform for seamless communication and collaboration among connected clients.

## **System Design & Architecture**

- b. System architecture based on inter processes communication (IPC), file input/output, processes and synchronization.
- c. IPC
  - i. System uses FIFO's which are First in First out files. When a client requests to connect to our server it writes its request to server fifo and server reads it if available it creates a fork and child server handles the client.



ii.

iii. Our server-client communication is described as an image. Server creator enters a number which is the maximum number of client that server can handle, when server is available and client wants to connect to server:

1. Client writes its requests to server fifo.
2. Main server reads it fifo.
3. if Main Server is available it exec fork and creates its child server, if not based on client request it waits or sends an answer to kill the client.
4. After this line occurs if the Client connects successfully.
5. Child server reads to the client fifo, when a client writes a request, the child server manages it. After writing the operation now the client waits to read the operation and the server will write to answer to fifo.
6. With this communication all clients connect to the server together and execute their requests without waiting if they do not affect the same file.

---

## Common Side

**d. Common.h holds error codes, structure and functions for client and server.**

**e. Variables/Structures**

i. Request struct

1. This structure holds user request information

a. pid / client pid

b. command / user input command

c. args[3] / user input arguments

d. argsCount / user input argument count

ii. Error Codes

**f. Functions**

i. Create Fifo | fifoLocation

1. This function creates fifo to location which taken as a argument

ii. isNumber | str

1. Checks str is number or not.

iii. transferFile | fileDestinationPath | fileSourcePath | response

1. This function copies file in fileSourcePath to

fileDestinationPath and also sets response message.

2. When same file name occurs in client side when client attempt to download or occurs in server side when the client attempt to download, new file name will be changed as newFile(1) if

---

there is same name it will increment one by one like  
newFile(X:int) until the name identical.

3. To prevent race condition, function uses fcntl lock mechanism so when any client wants to access file in source or destination , which will be created in function run time, they will wait until the lock mechanism is released.
4. The file transfer mechanism is read line by line and write the result line by line to.

iv. `rmtree` | `path`

1. This function needed to remove files for archServer temp files.
2. It is a well known linux function in many forums.
3. The remove mechanism is so simple: get file/folders in the path and if the folder goes into a folder like recursion at the end of the folder remove all things and come to the parent folder.

## Client Side

### g. Functions

i. `sigint_handler` | `signum`

1. This function helps us to get a SIGINT signal, which is usually triggered when the user presses CTRL+C in the terminal.
2. When user enter CTRL+C
  - a. If it connects to the server it sends a quit request and the server knows that that client will be dead, otherwise there is no connection with the server it kills itself without sending any messages.

ii. `mapUserInputToRequest` | `command` | `request`

- 
1. It processes command to new requests and if the command not valid it does not create any valid request and warn user.

#### **h. Main Code | `./client <Connect/tryConnect> serverPid`**

- i. Client main code takes three arguments.
- ii. First creates its fifo for speciality it uses its pid. Sets arguments.
- iii. Tries to connect to the server by using server pid which is taken by argument. If the connection is successful, it goes an infinite loop for server communication until quit or CTRL+C or any client enters killServer command.
- iv. Except for archServer requests, the working logic of all of them is the same. They write their request to client fifo and server reads that, after this transaction client waits to read and server writes a response message to client fifo and it goes on like this.
- v. When the user enters archServer command, there will be a child created and it downloads all files in the server side to the temp archive directory. After all downloads are completed, the child executes tar commands and makes tar files and the child dies. When tar file created client makes custom upload request and upload tar file to server also copies tar file to main client folder. After copy operations, the client deletes the temp archive file.
- vi. Rule: Do not create an archive named file in the client because when archServer command works it will delete.

---

## Server Side

### i. Semaphores

#### i. semControlClientConnection

1. It checks the rule of there is no client enter after server capacity . Also protects global and shared variable clientCounter.

### j. Functions

#### i. sigint\_handler | signum

1. This function helps us to get a SIGINT signal, which is usually triggered when the user presses CTRL+C in the terminal.
2. When user enter CTRL+C
  - a. It kills all client and child processes.

#### ii. sendResponse | request | message

1. It creates custom responses by using message request information and sends to client fifo.
2. Race condition prevents by using fcntl lock mechanism.

#### iii. logger | request | fdServerPath | message

1. It holds and writes all server messages/operations to log.txt file in the server side every operation owner (client) writes in log.txt file like:
  - a. PID-8296 readFull command tries to work..

---

iv. loggerInServer | fdServerPath | message

1. It holds and writes all server messages/operations to log.txt file in the server side every operation owner (server) writes in log.txt file like:

- a. SERVER Server Started PID 8978...

2. NO LOG FILE DELETED... If you open the same server again all logs will append log.txt.

v. helpWithoutArgument | request | fdServerPath | response

1. Set a response message to the valid help method.

vi. helpWithArgument | request | fdServerPath

1. Set response message to valid help method by comparing argument count.

vii. listServeDir | request | fdServerPath | response

1. Function executes ls function.
2. Set response to output of ls execution.

viii. readFull | request | fdServerPath | clientFifoPath | response

1. Function executes cat function to get all lines from a given file.
2. Set response message to cat result.
3. Race conditions are prevented by using fcntl.

ix. readPartial | request | fdServerPath | clientFifoPath | response

1. Function read line by line when it comes given line from user selected file sets response message.

- 
2. Race conditions are prevented by using fcntl.
- x. writeBasedLocation | request | fdServerPath | clientFifoPath | response
1. Function writes line of data end of file if user does not enter number of line, if enters desired line.
  2. It reads line by line and writes temp files. Temp file name includes pid so no race condition is expected. After reading/writing all files it rewinds file and temp file and does the vice-versa of the first operation. After all operations are done, release, unlock and delete temp files.
  3. If a file is not found, it creates a new one.
  4. Race conditions are prevented by using fcntl.
- xi. xLoadFile | request | fdServerPath | type | response
1. It arranges file names, their paths and makes ready to call transferFile function.
  2. If the type is one or two (archServer upload) it is an upload operation, otherwise it's a download operation.
  3. It calls transferFile function, and prepares a response message if the function is successful, otherwise does not set a response message it sends an error as a response.
- xii. archServer | request | fdServerPath | type | response
1. It reads the directory and takes every entry and transfer to the client as a download operation by using the transferFile function.
  2. Archived file saves in server and client also.



---

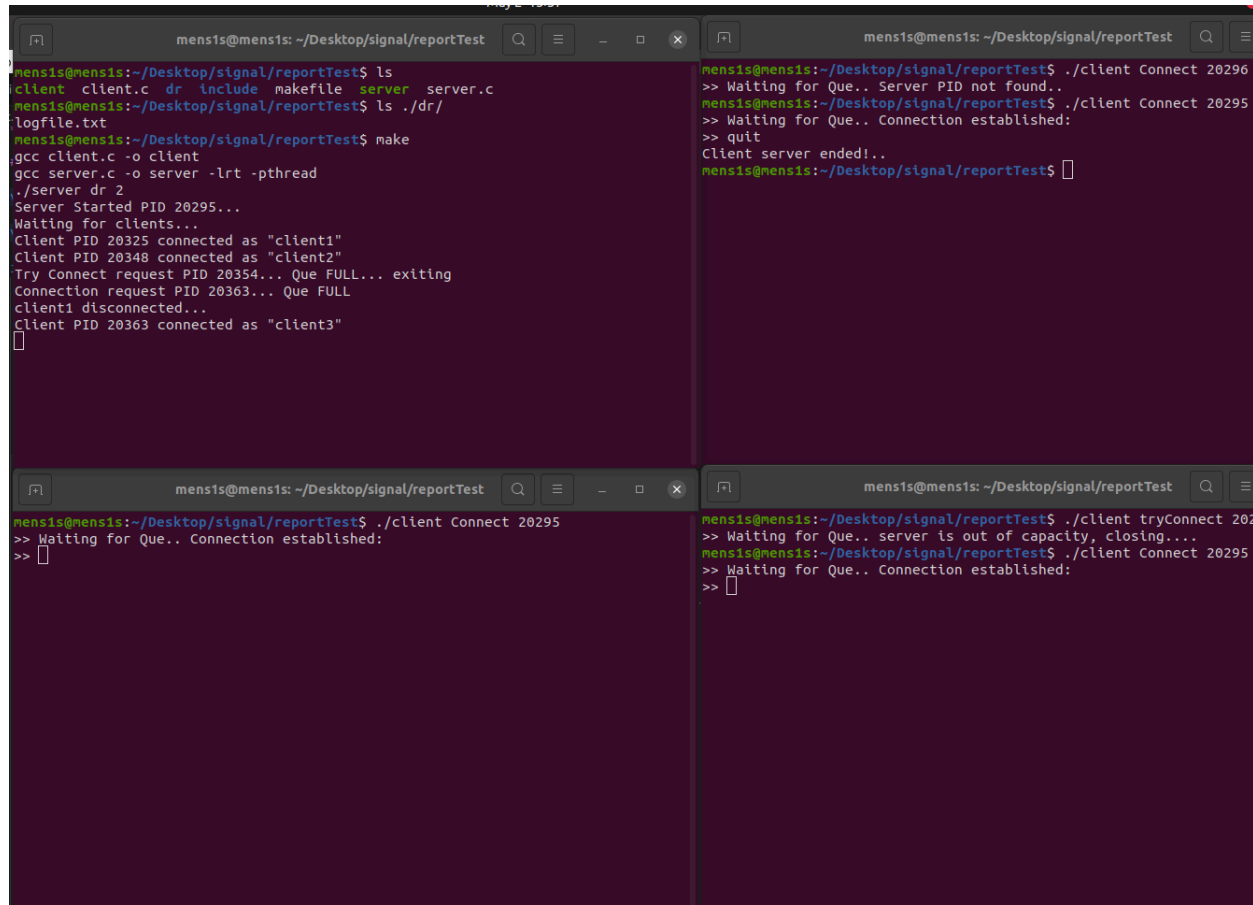
3. It prepares response messages.

k. Main Code | `./server <dirname> <max #ofClients>`

- i. Prepares local variables, initializes semaphore and shared variables.
- ii. In the infinitive loop reads server fifo and if a client wants to connect it decides to reach max number of clients or not and prepare a request to client to enter-wait-quit.
- iii. If a client enters a server, the server creates a fork and a child server serves the client.
- iv. Every request compared with default commands and selects one of them and calls the desired function if it cannot be selectable sends `INVALID_COMMAND`.
- v. When the `killServer` command is created all childs and clients will die.

# Test Cases

## I. Client Connection



```
mens1s@mens1s: ~/Desktop/signal/reportTest
mens1s@mens1s:~/Desktop/signal/reportTest$ ls
client client.c dr include makefile server server.c
mens1s@mens1s:~/Desktop/signal/reportTest$ ls ./dr/
logfile.txt
mens1s@mens1s:~/Desktop/signal/reportTest$ make
gcc client.c -o client
gcc server.c -o server -lrt -pthread
./server dr 2
Server Started PID 20295...
Waiting for clients...
Client PID 20325 connected as "client1"
Client PID 20348 connected as "client2"
Try Connect request PID 20354... Que FULL... exiting
Connection request PID 20363... Que FULL
client1 disconnected...
Client PID 20363 connected as "client3"

```

```
mens1s@mens1s:~/Desktop/signal/reportTest$ ./client Connect 20296
>> Waiting for Que.. Server PID not found..
mens1s@mens1s:~/Desktop/signal/reportTest$ ./client Connect 20295
>> Waiting for Que.. Connection established:
>> quit
Client server ended!..
mens1s@mens1s:~/Desktop/signal/reportTest$

```

```
mens1s@mens1s:~/Desktop/signal/reportTest$ ./client Connect 20295
>> Waiting for Que.. Connection established:
>>

```

```
mens1s@mens1s:~/Desktop/signal/reportTest$ ./client tryConnect 20295
>> Waiting for Que.. server is out of capacity, closing....
mens1s@mens1s:~/Desktop/signal/reportTest$ ./client Connect 20295
>> Waiting for Que.. Connection established:
>>

```

- WriteT - ReadF BASIC

```

mensis@mensis: ~/Desktop/signal/reportTest
mensis@mensis:~/Desktop/signal/reportTest$ ls
client client.c dr include makefile server server.c
mensis@mensis:~/Desktop/signal/reportTest$ ls ./dr/
logfile.txt
mensis@mensis:~/Desktop/signal/reportTest$ make
gcc client.c -o client
gcc server.c -o server -lrt -pthread
./server dr 2
Server Started PID 20295...
Waiting for clients...
Client PID 20325 connected as "client1"
Client PID 20348 connected as "client2"
Try Connect request PID 20354... Que FULL... exiting
Connection request PID 20363... Que FULL
client1 disconnected...
Client PID 20363 connected as "client3"

mensis@mensis:~/Desktop/signal/reportTest$ ./client Connect 20295
>> Waiting for Que.. Connection established:
>> help
Available commands are :
help, list, readF, writeT, upload, download, archServer, killServer, quit
>> readF deneme.txt
firstSentence
>> █

mensis@mensis:~/Desktop/signal/reportTest$ ./client tryConnect 20295
>> Waiting for Que.. server is out of capacity, closing...
mensis@mensis:~/Desktop/signal/reportTest$ ./client Connect 20295
>> Waiting for Que.. Connection established:
>> list
..
logfile.txt
log.txt
>> writeT deneme.txt firstSentence
Write operation done.
>> list
..
deneme.txt
logfile.txt
log.txt
>> █

// create special folder for upload-download

```

- WriteT - ReadF Hard
- WriteT to 6330226th line, ReadF from it. They start work at the same time.

```

mensis@mensis:~/Desktop/signal/reportTest
mensis@mensis:~/Desktop/signal/reportTest$ ls
client client.c dr include makefile server server.c
mensis@mensis:~/Desktop/signal/reportTest$ ls ./dr/
logfile.txt
mensis@mensis:~/Desktop/signal/reportTest$ make
gcc client.c -o client
gcc server.c -o server -lrt -pthread
./server dr 2
Server Started PID 20295...
Waiting for clients...
Client PID 20325 connected as "client1"
Client PID 20348 connected as "client2"
Try Connect request PID 20354... Que FULL... exiting
Connection request PID 20363... Que FULL
client1 disconnected...
Client PID 20363 connected as "client3"

mensis@mensis:~/Desktop/signal/reportTest$ ./client Connect 20295
>> Waiting for Que.. Connection established:
>> help
Available commands are :
help, list, readF, writeT, upload, download, archServer, killServer, quit
>> readF deneme.txt
firstSentence
>> writeT logfile.txt 6330226 May2:15:42
Write operation done.
>> █

mensis@mensis:~/Desktop/signal/reportTest$ ./client tryConnect 20295
>> Waiting for Que.. server is out of capacity, closing...
mensis@mensis:~/Desktop/signal/reportTest$ ./client Connect 20295
>> Waiting for Que.. Connection established:
>> list
..
logfile.txt
log.txt
>> writeT deneme.txt firstSentence
Write operation done.
>> list
..
deneme.txt
logfile.txt
log.txt
>> readF logfile.txt 6330226
May2:15:42
>> █

```

- WriteT - ReadF unexpected values

```

mensis@mensis: ~/Desktop/signal/reportTest
mensis@mensis:~/Desktop/signal/reportTest$ ls
client client.c dr include makefile server server.c
mensis@mensis:~/Desktop/signal/reportTest$ ls ./dr/
logfile.txt
mensis@mensis:~/Desktop/signal/reportTest$ make
gcc client.c -o client
gcc server.c -o server -lrt -pthread
./server dr 2
Server Started PID 20295...
Waiting for clients...
Client PID 20325 connected as "client1"
Client PID 20348 connected as "client2"
Try Connect request PID 20354... Que FULL... exiting
Connection request PID 20363... Que FULL
client1 disconnected...
Client PID 20363 connected as "client3"

mensis@mensis:~/Desktop/signal/reportTest$ ./client Connect 20295
>> Waiting for Que.. Connection established:
>> help
Available commands are :
help, list, readF, writeT, upload, download, archServer, killServer, quit

>> readF deneme.txt
firstSentence

>> writeT logfile.txt 60.satiryokeklemazanayinedegidersonunaekler
Write operation done.

>> writeT deneme.txt 60.satiryokeklemazanayinedegidersonunaekler
Write operation done.

>> readF deneme.txt
firstSentence
60.satiryokeklemazanayinedegidersonunaekler

>>

```

- Upload / Download small files / same file occurs in server

```

mensis@mensis:~/Desktop/signal/reportTest
mensis@mensis:~/Desktop/signal/reportTest$ ls
client client.c dr include makefile server server.c
mensis@mensis:~/Desktop/signal/reportTest$ ls ./dr/
logfile.txt
mensis@mensis:~/Desktop/signal/reportTest$ make
gcc client.c -o client
gcc server.c -o server -lrt -pthread
./server dr 2
Server Started PID 20295...
Waiting for clients...
Client PID 20325 connected as "client1"
Client PID 20348 connected as "client2"
Try Connect request PID 20354... Que FULL... exiting
Connection request PID 20363... Que FULL
client1 disconnected...
Client PID 20363 connected as "client3"

mensis@mensis:~/Desktop/signal/reportTest$ ./client Connect 20295
>> Waiting for Que.. Connection established:
>> help
Available commands are :
help, list, readF, writeT, upload, download, archServer, killServer, quit

>> readF deneme.txt
firstSentence

>> writeT logfile.txt 60.satiryokeklemazanayinedegidersonunaekler
Write operation done.

>> writeT deneme.txt 60.satiryokeklemazanayinedegidersonunaekler
Write operation done.

>> readF deneme.txt
firstSentence
60.satiryokeklemazanayinedegidersonunaekler

>> list
..
deneme.txt
logfile.txt
log.txt

>> readF logfile.txt 6330226
May2:15:42

>> readF deneme.txt 60
FILE_LINE_COUNT_MATCH_ERROR

>> list
..
deneme.txt
logfile.txt
log.txt

>> list
..
deneme(1).txt
deneme.txt
logfile.txt
log.txt

>>

```

- Upload download big files | Same name in client side (i did not download log(1).txt)

```

mens1s@mens1s: ~/Desktop/signal/reportTest
mens1s@mens1s:~/Desktop/signal/reportTest$ ls
client client.c dr include makefile server server.c
mens1s@mens1s:~/Desktop/signal/reportTest$ ls ./dr/
logfile.txt
mens1s@mens1s:~/Desktop/signal/reportTest$ make
gcc client.c -o client
gcc server.c -o server -lrt -pthread
./server dr 2
Server Started PID 20295...
Waiting for clients...
Client PID 20325 connected as "client1"
Client PID 20348 connected as "client2"
Try Connect request PID 20354... Que FULL... exiting
Connection request PID 20363... Que FULL
client1 disconnected...
Client PID 20363 connected as "client3"

mens1s@mens1s:~/Desktop/signal/reportTest
mens1s@mens1s:~/Desktop/signal/reportTest$ ./client
birtncisatir
firstSentence
60.satiryokeklemezamayinedegidersonunaekler

>> list
..
deneme.txt
logfile.txt
log.txt

>> download deneme.txt
File downloaded succesfully.
71 bytes transferred
>> upload deneme.txt
File uploaded succesfully.
71 bytes transferred
>> download logfile.txt
File downloaded succesfully.
24048199 bytes transferred
>> upload logfile.txt
File uploaded succesfully.
24048199 bytes transferred
>>

mens1s@mens1s:~/Desktop/signal/reportTest
mens1s@mens1s:~/Desktop/signal/reportTest$ ./server
File uploaded succesfully.
24048199 bytes transferred
>> list
..
deneme(1).txt
deneme.txt
logfile(1).txt
logfile(2).txt
logfile.txt
log.txt

>> download log.txt
INVALID_COMMAND
>> download log.txt
File downloaded succesfully.
1232 bytes transferred
>> download log.txt
File downloaded succesfully.
1268 bytes transferred
>> upload log(1).txt
File uploaded succesfully.
1268 bytes transferred
>>

```

- Testing race condition... WriteT logfile.txt 5M first writeT logfile.txt 5M second readF logfile.txt 5M logfile.txt sets to 10mb> so read and write operations are so long...
- first image

```

mens1s@mens1s: ~/Desktop/signal/reportTest
mens1s@mens1s:~/Desktop/signal/reportTest$ make
gcc client.c -o client
gcc server.c -o server -lrt -pthread
./server dr 2
Server Started PID 20295...
Waiting for clients...
Client PID 20325 connected as "client1"
Client PID 20348 connected as "client2"
Try Connect request PID 20354... Que FULL... exiting
Connection request PID 20363... Que FULL
client1 disconnected...
Client PID 20363 connected as "client3"
^CExiting from server (signum): (2)
Exiting from server (signum): (2)
Exiting from server (signum): (2)
make: *** [makefile:4: all] Error 2

mens1s@mens1s:~/Desktop/signal/reportTest$ ./server dr 3
Server Started PID 21962...
Waiting for clients...
Client PID 21994 connected as "client1"
Client PID 22017 connected as "client2"
Client PID 22023 connected as "client3"

mens1s@mens1s:~/Desktop/signal/reportTest$
>> list
..
deneme.txt
logfile.txt
log.txt

>> download deneme.txt
File downloaded successfully.
71 bytes transferred
>> upload deneme.txt
File uploaded successfully.
71 bytes transferred
>> download logfile.txt
File downloaded successfully.
24048199 bytes transferred
>> upload logfile.txt
File uploaded successfully.
24048199 bytes transferred
>> Killed
mens1s@mens1s:~/Desktop/signal/reportTest$ ./client Connect 21962
>> Waiting for Que.. Connection established:
>> writeI logfile.txt 5000000 first

mens1s@mens1s:~/Desktop/signal/reportTest$
>>
deneme(1).txt
deneme.txt
logfile(1).txt
logfile(2).txt
logfile.txt
log.txt

>> download log.txt
INVALID_COMMAND
>> download log.txt
File downloaded successfully.
1232 bytes transferred
>> download log.txt
File downloaded successfully.
1268 bytes transferred
>> upload log(1).txt
File uploaded successfully.
1268 bytes transferred
>> Killed
mens1s@mens1s:~/Desktop/signal/reportTest$ ./client Connect 21962
>> Waiting for Que.. Connection established:
>> writeI logfile.txt 5000000 second

```

- Second image. It takes 1 minute. reads and writes logfile.txt temp \* 2

```

mens1s@mens1s: ~/Desktop/signal/reportTest
mens1s@mens1s:~/Desktop/signal/reportTest$ make
gcc client.c -o client
gcc server.c -o server -lrt -pthread
./server dr 2
Server Started PID 20295...
Waiting for clients...
Client PID 20325 connected as "client1"
Client PID 20348 connected as "client2"
Try Connect request PID 20354... Que FULL... exiting
Connection request PID 20363... Que FULL
client1 disconnected...
Client PID 20363 connected as "client3"
^CExiting from server (signum): (2)
Exiting from server (signum): (2)
Exiting from server (signum): (2)
make: *** [makefile:4: all] Error 2

mens1s@mens1s:~/Desktop/signal/reportTest$ ./server dr 3
Server Started PID 21962...
Waiting for clients...
Client PID 21994 connected as "client1"
Client PID 22017 connected as "client2"
Client PID 22023 connected as "client3"

mens1s@mens1s:~/Desktop/signal/reportTest$
>> list
..
deneme.txt
logfile.txt
log.txt

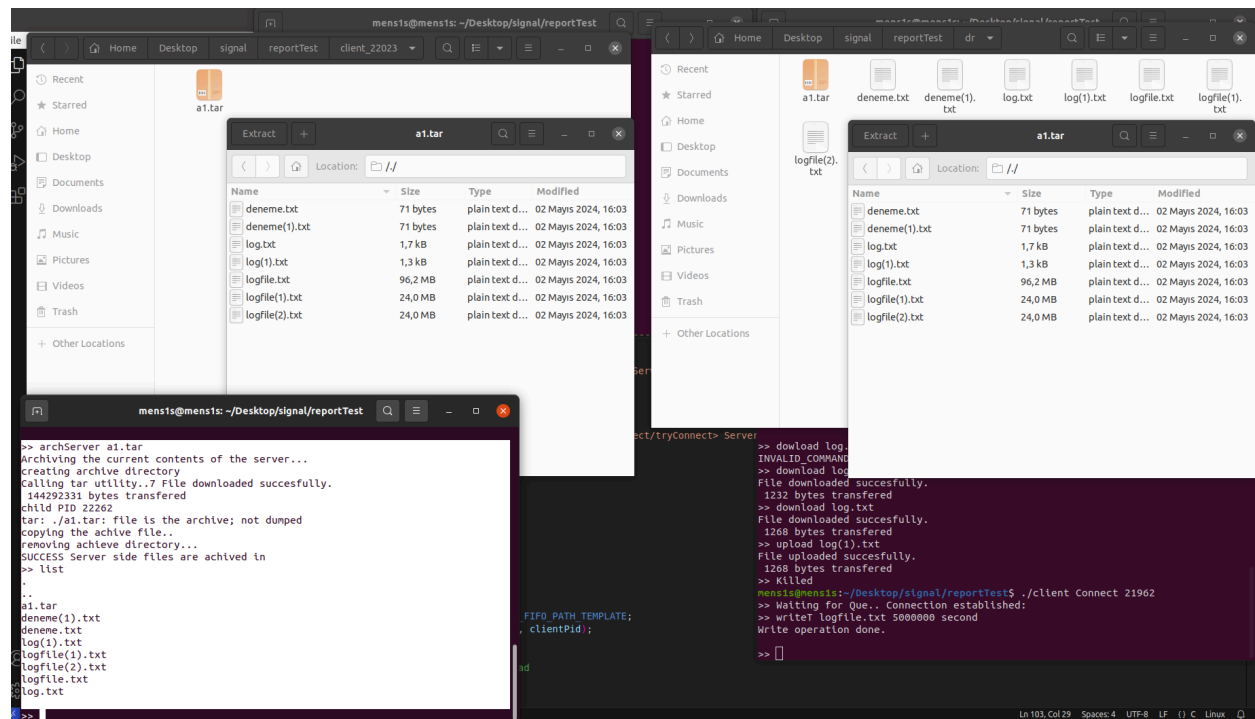
>> download deneme.txt
File downloaded successfully.
71 bytes transferred
>> upload deneme.txt
File uploaded successfully.
71 bytes transferred
>> download logfile.txt
File downloaded successfully.
24048199 bytes transferred
>> upload logfile.txt
File uploaded successfully.
24048199 bytes transferred
>> Killed
mens1s@mens1s:~/Desktop/signal/reportTest$ ./client Connect 21962
>> Waiting for Que.. Connection established:
>> writeI logfile.txt 5000000 first
Write operation done.

mens1s@mens1s:~/Desktop/signal/reportTest$
>>
deneme.txt
logfile(1).txt
logfile(2).txt
logfile.txt
log.txt

>> download log.txt
INVALID_COMMAND
>> download log.txt
File downloaded successfully.
1232 bytes transferred
>> download log.txt
File downloaded successfully.
1268 bytes transferred
>> upload log(1).txt
File uploaded successfully.
1268 bytes transferred
>> Killed
mens1s@mens1s:~/Desktop/signal/reportTest$ ./client Connect 21962
>> Waiting for Que.. Connection established:
>> writeI logfile.txt 5000000 second
Write operation done.

```

- ArchServer



- Quit - CTRL+C in Client - KillServer Test

