GEBZE TECHNICAL UNIVERSITY

CSE 344
SYSTEM PROGRAMMING
Midterm Report

OGUZ MUTLU
1801042624

1.Problem Definition

Our task is to design and implement a file server that enables multiple clients to connect and modify and archive the files in a specific directory located at the server side.

This project should be implemented as a server and cliend side.

Server side :

neHosServer <dirname> <max. #ofClients>

- create a log file for the clients and prompt its PID for the clients to connect.

- The for each client connected will fork a copy of itself in order to serve the specified client

-If a kill signal is generated (either by Ctrl-C or from a client side request)

Server is expected to display the request, send kill signals (requests) to its child processes, ensure the log file is created properly and exit.

Client side :

neHosClient <Connect/tryConnect> ServerPID

-the client program with Connect option request a spot from the Server Que with ServerPID and connects if a spot is available

- performing request like; help, list, readF, writeT upload, download killServer, quit, archServer.

My program has to be ensure data consistency and avoid data corruption.

My program has to be used multiple processes for file access and synchronization

Signals has tp be handled properly on both client and Server sides

My program also should enforce mutual exclusion to prevent race conditions

2.Implementation

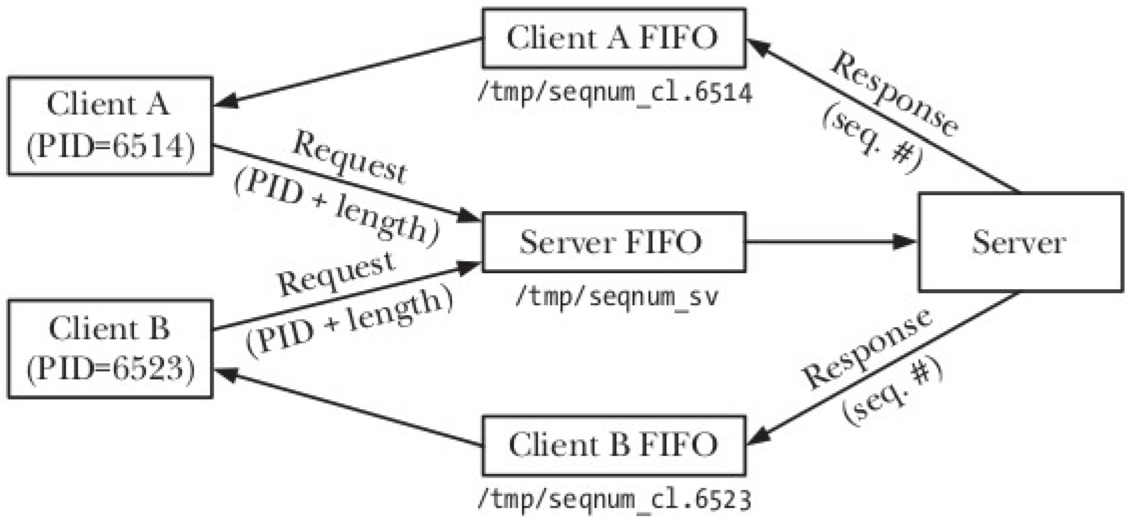


Figure 44-6: Using FIFOs in a single-server, multiple-client application

Figure 1

FIFO was preferred as the IPC mechanism for the homework application. The client-server model was created with FIFO, which was explained in the lesson.

First, a server fifo is created. Every client who wants to connect via this server fifo file sends a request with the **server pid** and its **own pid**. If the queue is not full and the process can be connected, another fifo is opened with the client pid so that separate communication can be made for each client.(/tmp/clientFifo.%ld→ ld takes clients pid)

To explain it in a little more detail; First of all, the client server writes its own pid and the pid of the server to which it will send the request to the server fifo. The server fifo read is waiting for this request. After the necessary checks are made, if the appropriate conditions are met, the client says that it will open the server with its own pid value, and it communicates with the client by creating a child process on the server.

Since the child process does this communication work, the server is actually always listening for requests from other clients.

On the client side, a structure similar to the one for the initial communication of the server has been established. Differently, this time both reading and writing operations will occur over the same fifo.

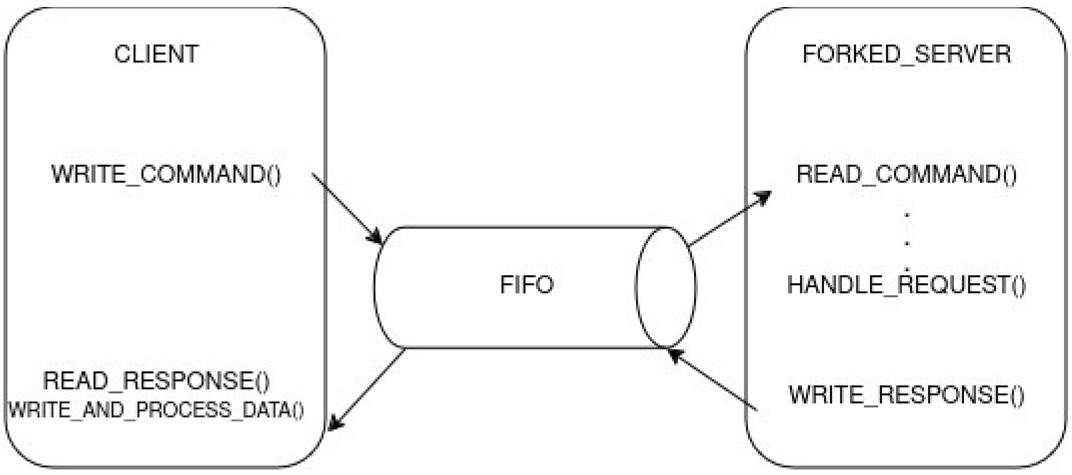


Figure 2

The command is expected with a struct (" >> Enter command: ") and after this value is entered, it is sent to the server. After the necessary parse operations and command correctness checks are performed, it is written to the appropriate response buffer and sent back to the client via fifo.

SYNCRONIZATION PROBLEMS AND IMPLEMENTATION

1.SYNCRONIZATION PROBLEM-1 (FIFO SERVER CLIENT): *Although it can wait thanks to the read system call while communicating over the same fifo, the ipc mechanism may be disrupted **if read reads EOF or opened with nonblocking mode.** An external solution has been introduced to solve this. An attempt was made to solve this problem by using **semaphore**.*

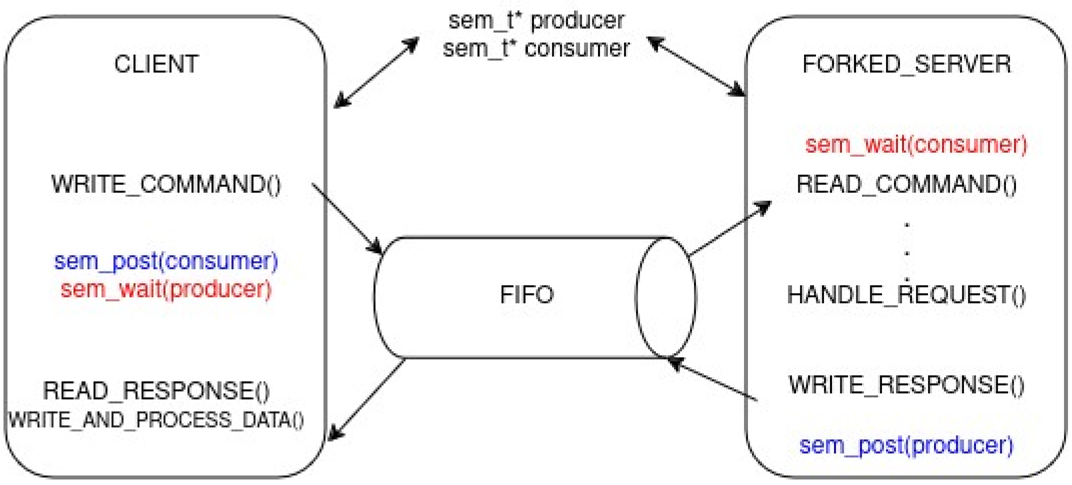


Figure 3

In case of a synchronization problem explained in the problem section above, simultaneous access to the buffer by semaphore wait and post processes is prevented and appropriate communication is ensured.

2.SYNCRONIZATION PROBLEM 2(UPLOAD DOWNLOAD):

The second synchronization problem we encounter in the project is seen in the upload and download processes. After accessing the file for the upload process, the entire content of the file must be accessed in order to be copied in accordance with the given file path. As will be explained below, it was predicted that the fifon was not sufficient (the reason will be explained below). The architecture was established to communicate via shared memory. However, writing to the same shared memory and reading it later creates a synchronization problem.(Solution producer-consumer approach)

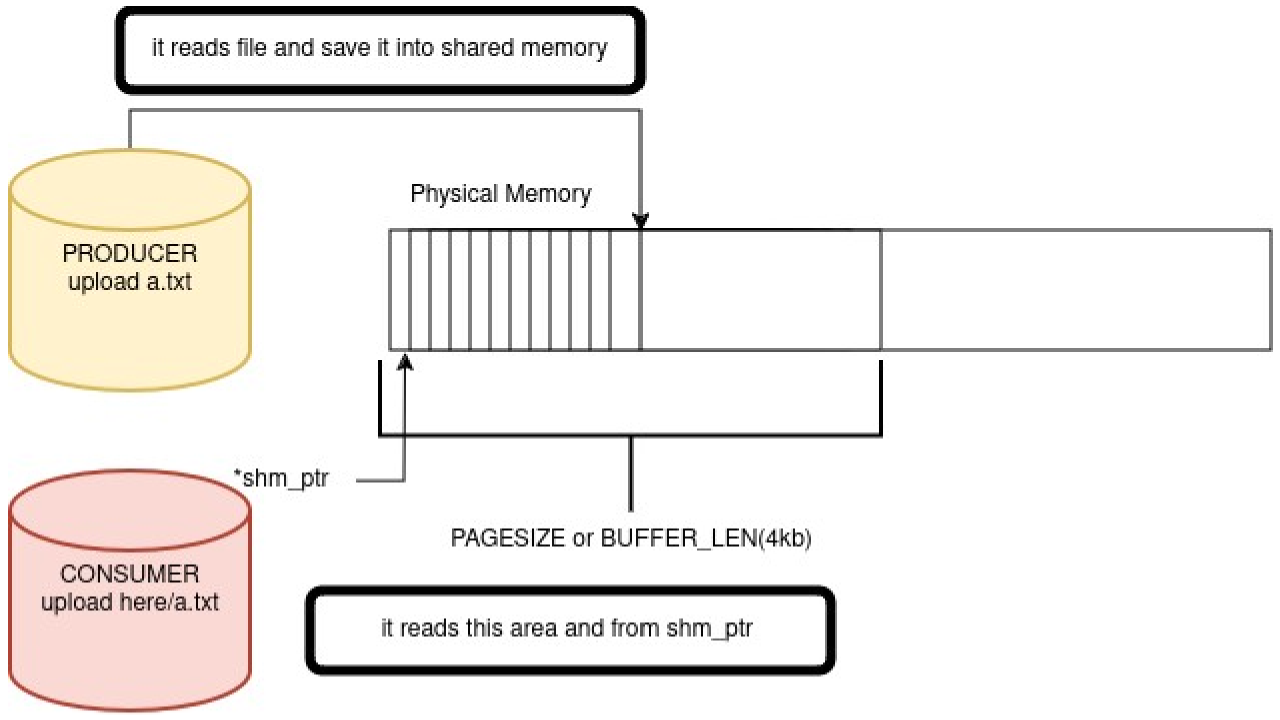

it reads file and save it into shared memory
PRODUCER
upload a.txt
Physical Memory
*shm_ptr
CONSUMER
upload here/a.txt
PAGESIZE or BUFFER_LEN(4kb)
it reads this area and from shm_ptr

Figure 4

As explained in the figure above, a new synchronization mechanism has been established with semaphore to prevent both writing and reading operations in the same shared memory mapping area from being performed at the same time. Due to its similarity to the producer consumer logic, the semaphore structure has been used as given below in the form of pseudocode.

```
void producer(void){
int item;
while (TRUE) {
  item = produce item( );
  down(&empty);
  down(&mutex);
  inser t item(item);
  up(&mutex);
  up(&full);
}}
```

```
void consumer(void){
 int item;
 while (TRUE) {
   down(&full);
   down(&mutex);
   item = remove item( );
   up(&mutex);
   up(&empty);
   consume item(item);
}}
```

LARGE FILE HANDLE SYSTEM

Problem: Due to the FIFO being established on a one-time response, problems arise in the upload, download and read operations of large files. Because the buffer is determined as a maximum of 256 bytes and a one-time response is frozen or read.

Solution: As a solution to this problem, when the readF, upload, download commands come, the shared memory structure is switched, regardless of the size of the file.

Since these commands are considered a one-time response, the same command and file name are sent to the client again (for example, upload grades.txt). The client side understands whether the first command is upload download or not and runs the **producer consumer function**. **The server side performs another fork operation before performing the return of response. This fork operation is performed to fill and empty the buffer in the while loop.Explained in figure below.**

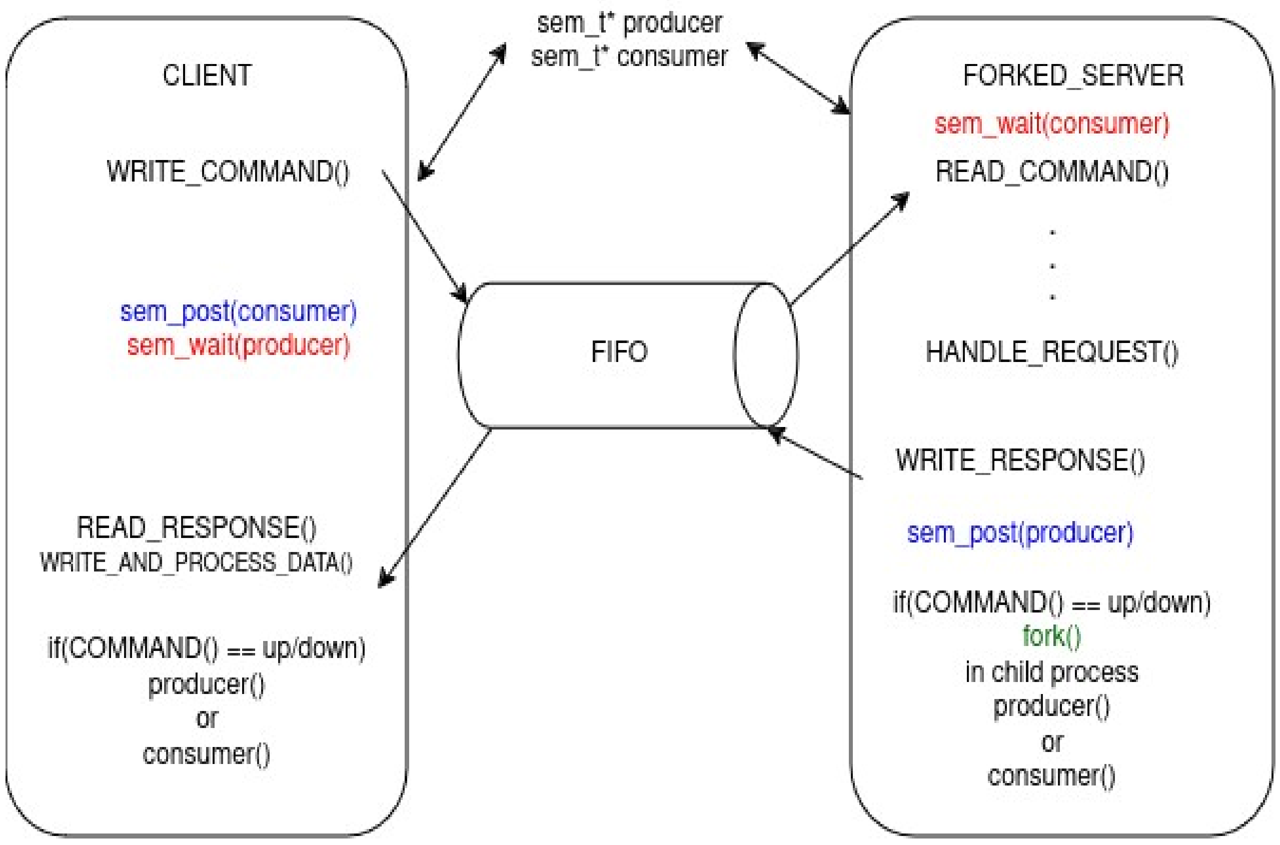


Figure 5

SIGNAL HANDLING

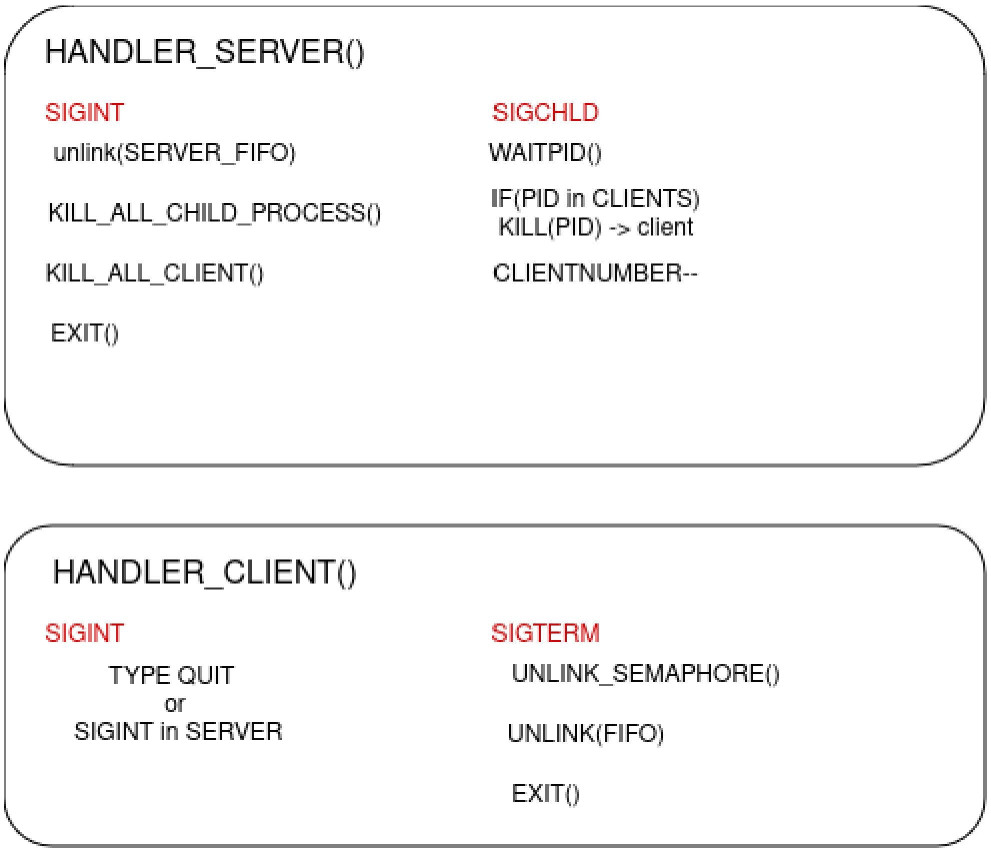
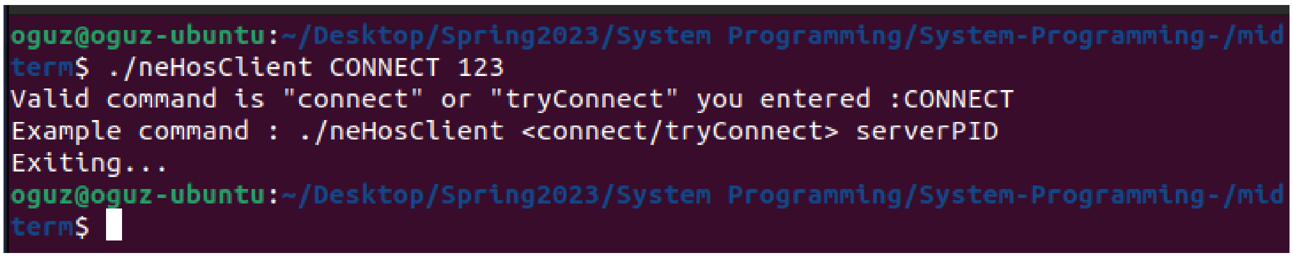


Figure 6

3.Results and Test Cases

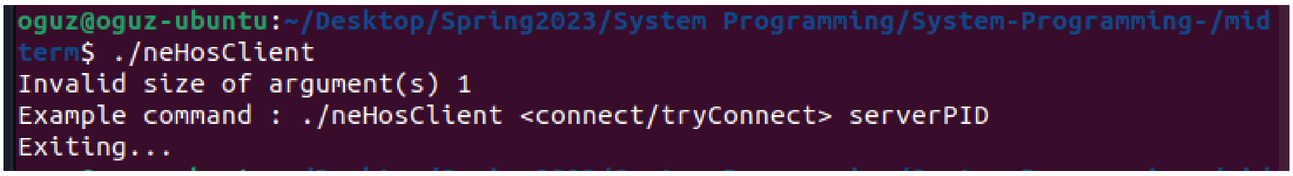
3.1 Invalid Entry Tests

3.1.1. Connect Error Client



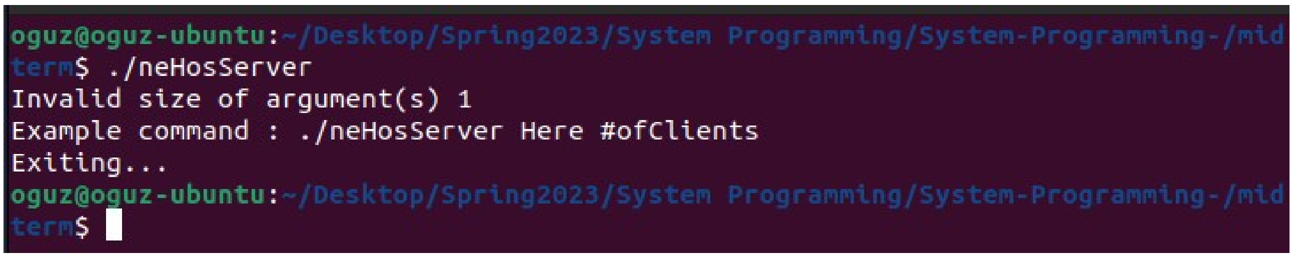
```
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ ./neHosClient CONNECT 123
Valid command is "connect" or "tryConnect" you entered :CONNECT
Example command : ./neHosClient <connect/tryConnect> serverPID
Exiting...
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ 
```

3.1.2. Invalid Arg Client



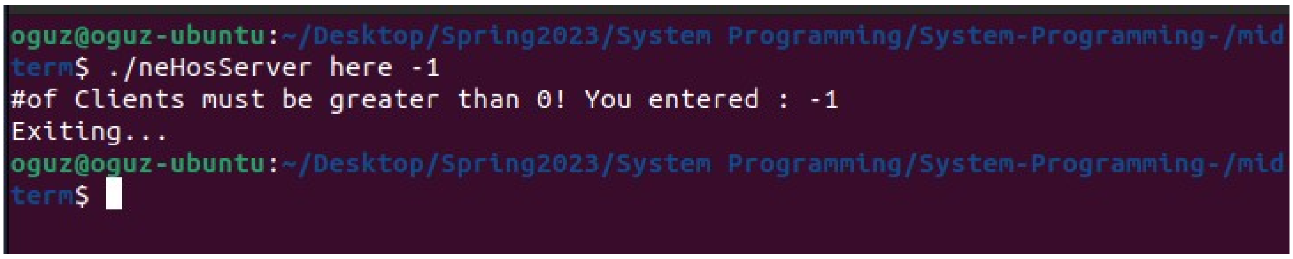
```
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ ./neHosClient
Invalid size of argument(s) 1
Example command : ./neHosClient <connect/tryConnect> serverPID
Exiting...
```

3.1.3 Invalid size of argc Server



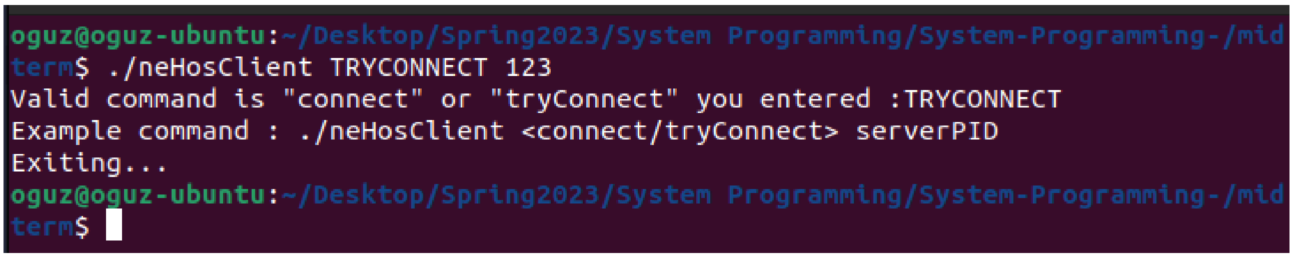
```
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ ./neHosServer
Invalid size of argument(s) 1
Example command : ./neHosServer Here #ofClients
Exiting...
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ 
```

3.1.4 Invalid size of client



```
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ ./neHosServer here -1
#of Clients must be greater than 0! You entered : -1
Exiting...
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ 
```

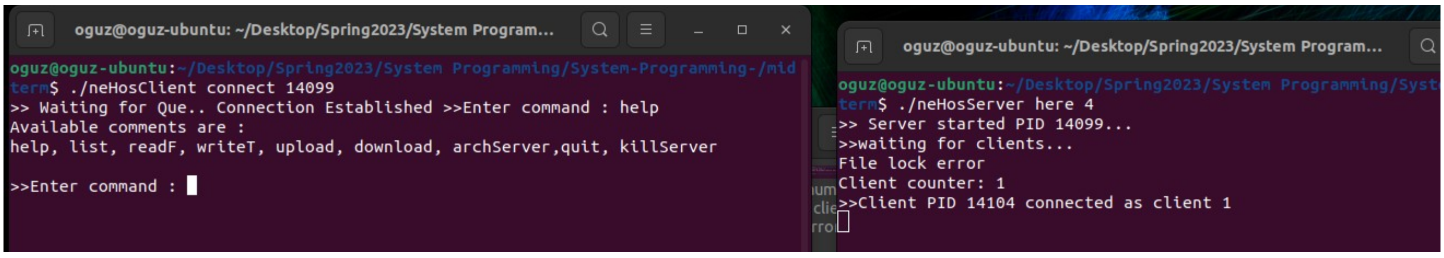
3.1.5 Invalid entry of tryconnect



```
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ ./neHosClient TRYCONNECT 123
Valid command is "connect" or "tryConnect" you entered :TRYCONNECT
Example command : ./neHosClient <connect/tryConnect> serverPID
Exiting...
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ 
```

3.2 System Requirement And Command Test

3.2.1 Help



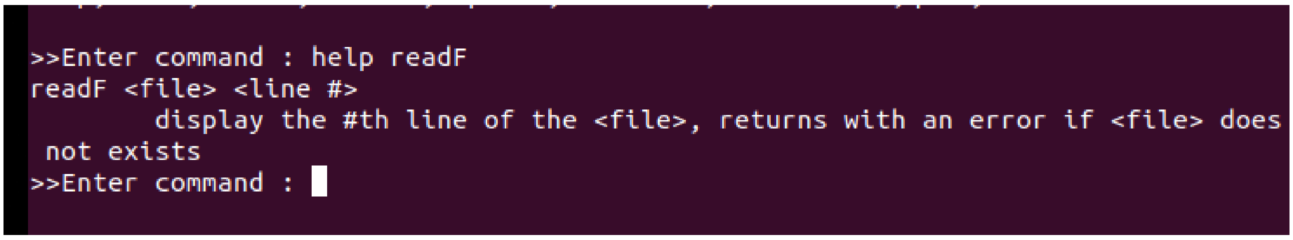
```
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ ./neHosClient connect 14099
>> Waiting for Que.. Connection Established >>Enter command : help
Available comments are :
help, list, readF, writeT, upload, download, archServer,quit, killServer

>>Enter command :
```

```
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/Syst
term$ ./neHosServer here 4
>> Server started PID 14099...
>>waiting for clients...
File lock error
Client counter: 1
>>Client PID 14104 connected as client 1
```

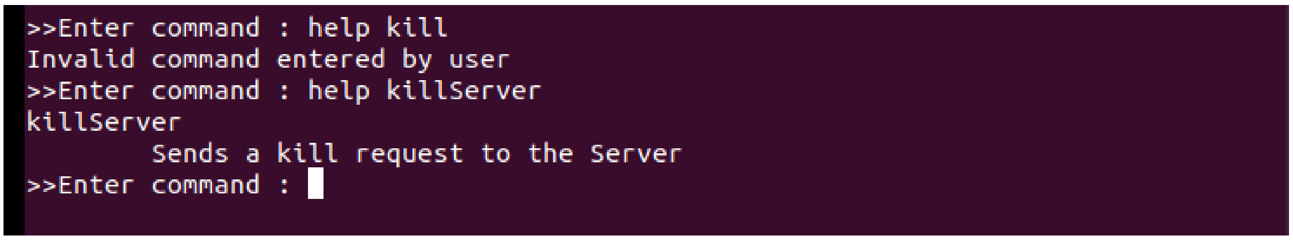
oguz@oguz-ubuntu: ~/Desktop/Spring2023/System Program...
oguz@oguz-ubuntu: ~/Desktop/Spring2023/System Program...

3.2.2 Help <command>



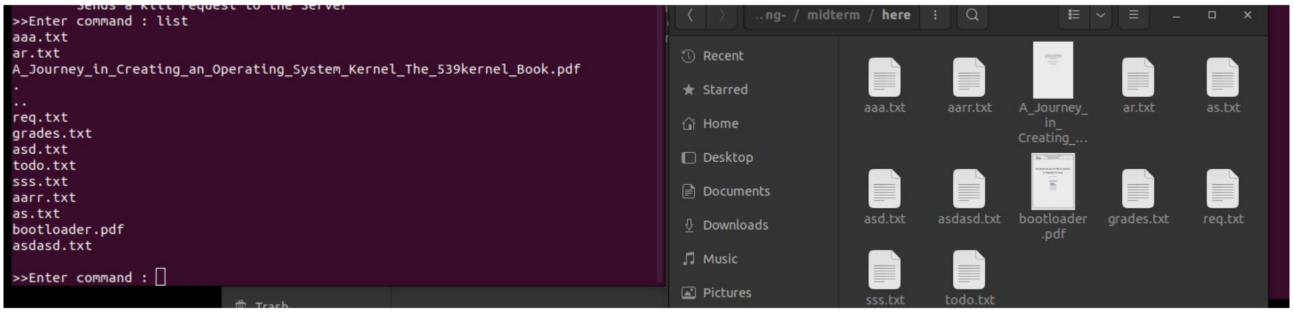
```
>>Enter command : help readF
readF <file> <line #>
        display the #th line of the <file>, returns with an error if <file> does
 not exists
>>Enter command :
```

3.2.3 Help <command>



```
>>Enter command : help kill
Invalid command entered by user
>>Enter command : help killServer
killServer
        Sends a kill request to the Server
>>Enter command :
```

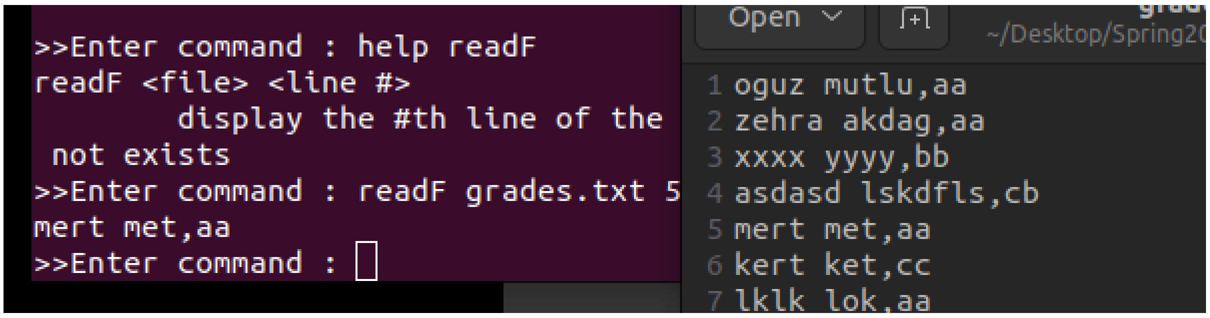
3.2.4 List



```
>>Enter command : list
aaa.txt
ar.txt
A_Journey_in_Creating_an_Operating_System_Kernel_The_539kernel_Book.pdf
.
..
req.txt
grades.txt
asd.txt
todo.txt
sss.txt
aarr.txt
as.txt
bootloader.pdf
asdasd.txt

>>Enter command :
```

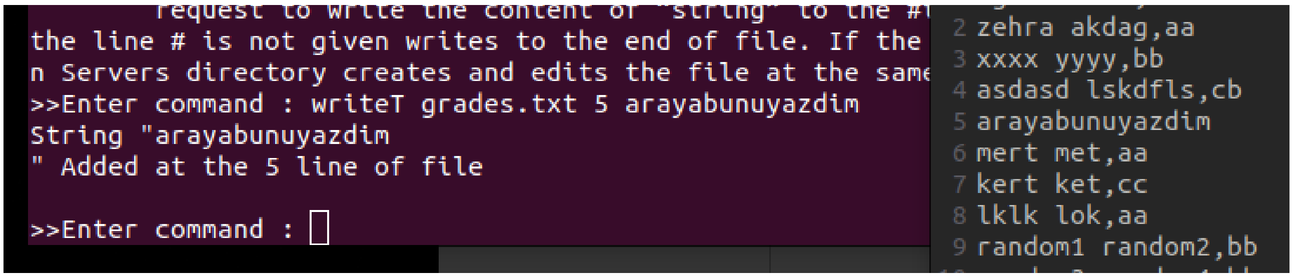
3.2.5 readF <file> <line>



```
>>Enter command : help readF
readF <file> <line #>
        display the #th line of the
 not exists
>>Enter command : readF grades.txt 5
mert met,aa
>>Enter command :
```

```
1 oguz mutlu,aa
2 zehra akdag,aa
3 xxxx yyyy,bb
4 asdasd lskdfls,cb
5 mert met,aa
6 kert ket,cc
7 lklk lok,aa
```

3.2.6 writeT<file> <line> <string>

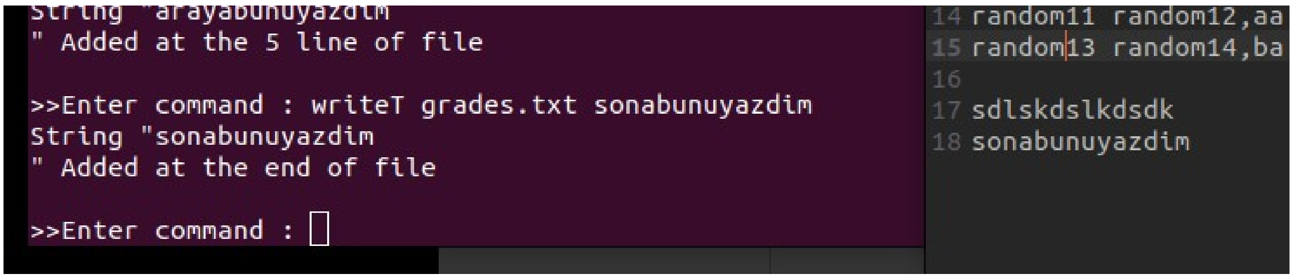


```
    request to write the content of "string" to the #l
the line # is not given writes to the end of file. If the
n Servers directory creates and edits the file at the same
>>Enter command : writeT grades.txt 5 arayabunuyazdim
String "arayabunuyazdim
" Added at the 5 line of file

>>Enter command : 
```

```
 2 zehra akdag,aa
 3 xxxx yyyy,bb
 4 asdasd lskdfls,cb
 5 arayabunuyazdim
 6 mert met,aa
 7 kert ket,cc
 8 lklk lok,aa
 9 random1 random2,bb
```

3.2.7 writeT<file> <string>



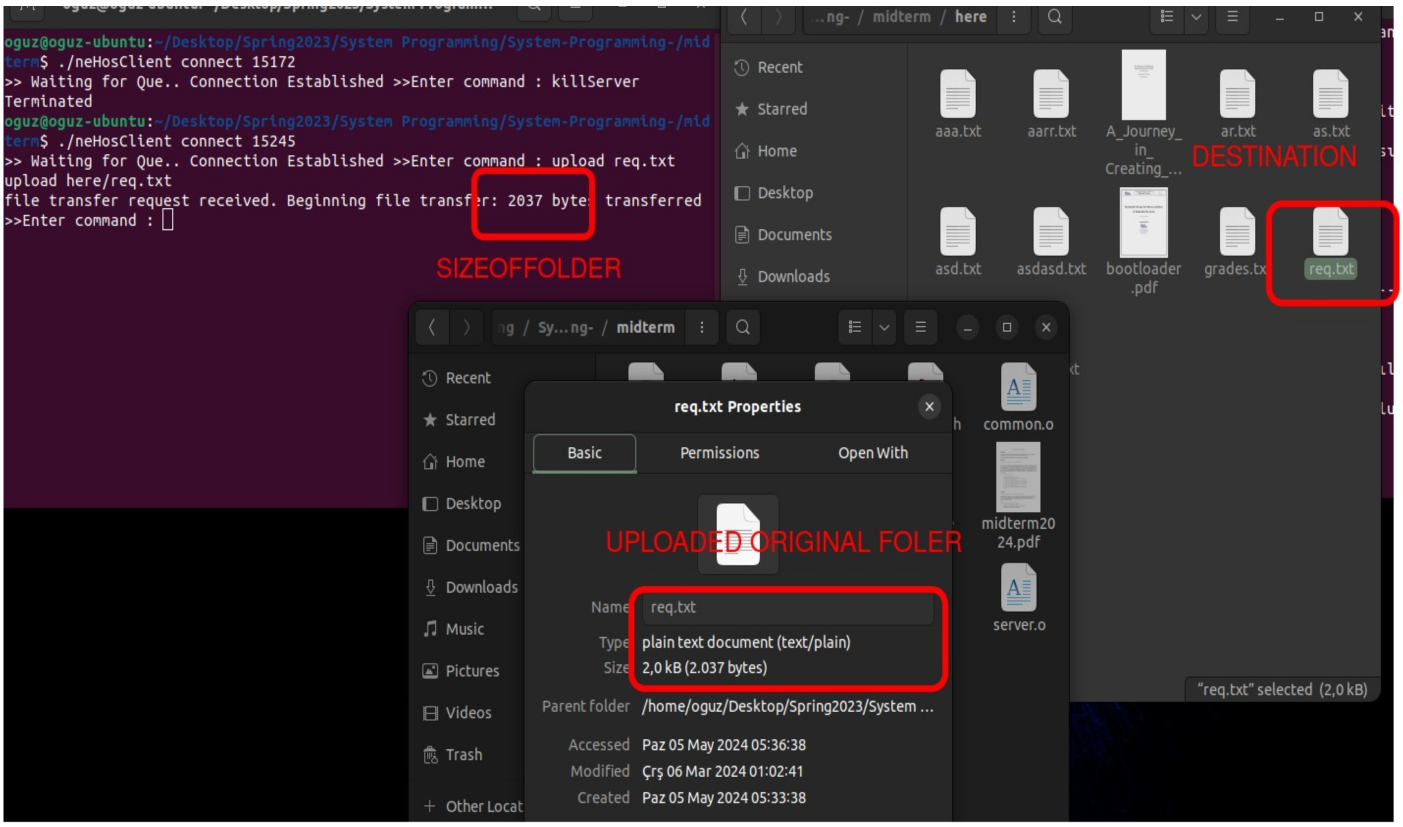
```
String "arayabunuyazdim
" Added at the 5 line of file

>>Enter command : writeT grades.txt sonabunuyazdim
String "sonabunuyazdim
" Added at the end of file

>>Enter command : 
```

```
14 random11 random12,aa
15 random13 random14,ba
16
17 sdlskdslkdsdk
18 sonabunuyazdim
```

3.2.8 upload <file>



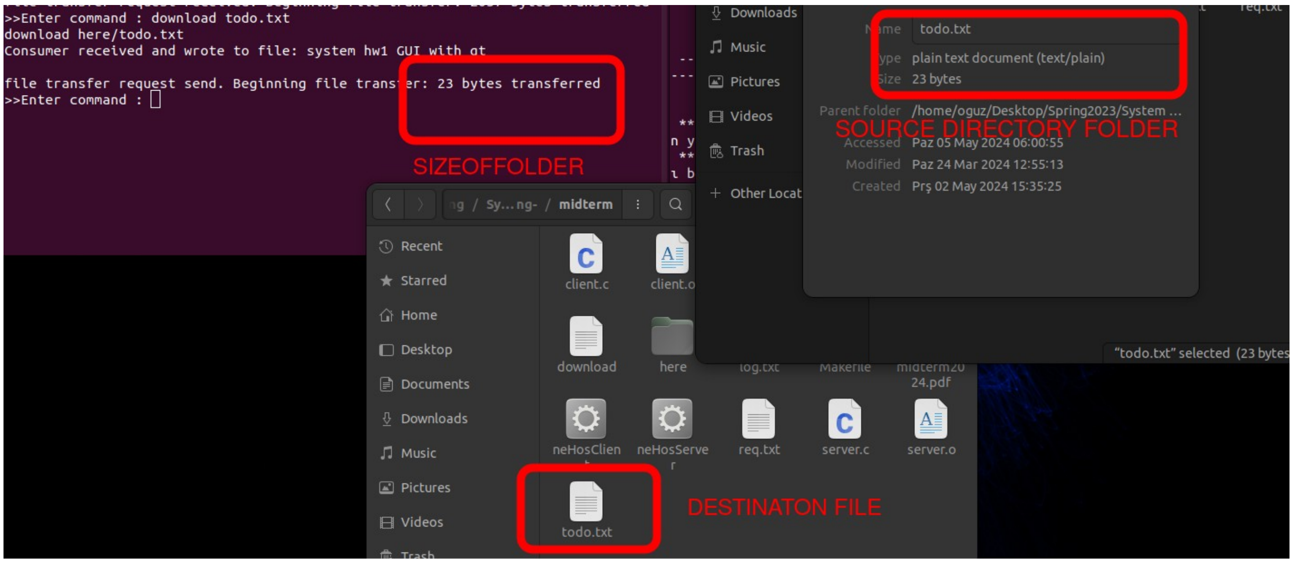
```
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ ./neHosClient connect 15172
>> Waiting for Que.. Connection Established >>Enter command : killServer
Terminated
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ ./neHosClient connect 15245
>> Waiting for Que.. Connection Established >>Enter command : upload req.txt
upload here/req.txt
file transfer request received. Beginning file transfer: 2037 bytes transferred
>>Enter command : 
```

req.txt Properties

Name: req.txt
Type: plain text document (text/plain)
Size: 2,0 kB (2.037 bytes)
Parent folder: /home/oguz/Desktop/Spring2023/System ...
Accessed: Paz 05 May 2024 05:36:38
Modified: Çrş 06 Mar 2024 01:02:41
Created: Paz 05 May 2024 05:33:38

"req.txt" selected (2,0 kB)

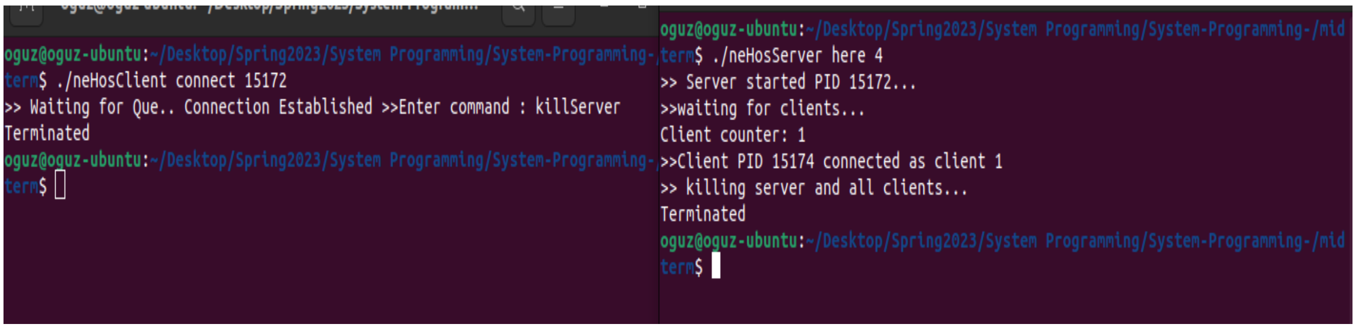
3.2.9 download <file>



```
>>Enter command : download todo.txt
download here/todo.txt
Consumer received and wrote to file: system hw1 GUI with qt

file transfer request send. Beginning file transfer: 23 bytes transferred
>>Enter command :
```

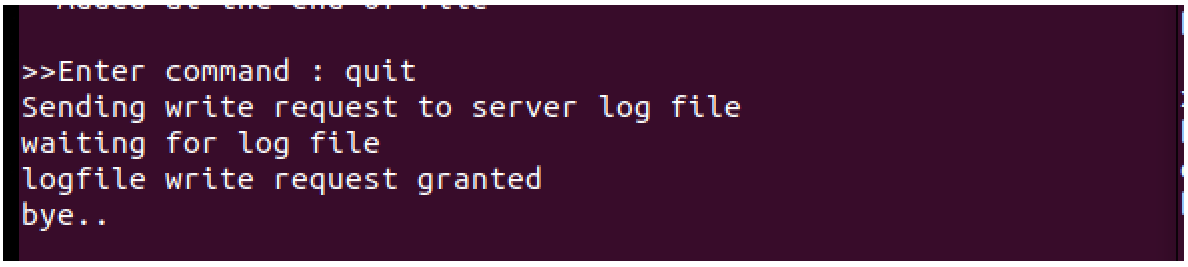
3.2.10 killServer



```
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-
term$ ./neHosClient connect 15172
>> Waiting for Que.. Connection Established >>Enter command : killServer
Terminated
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-
term$
```

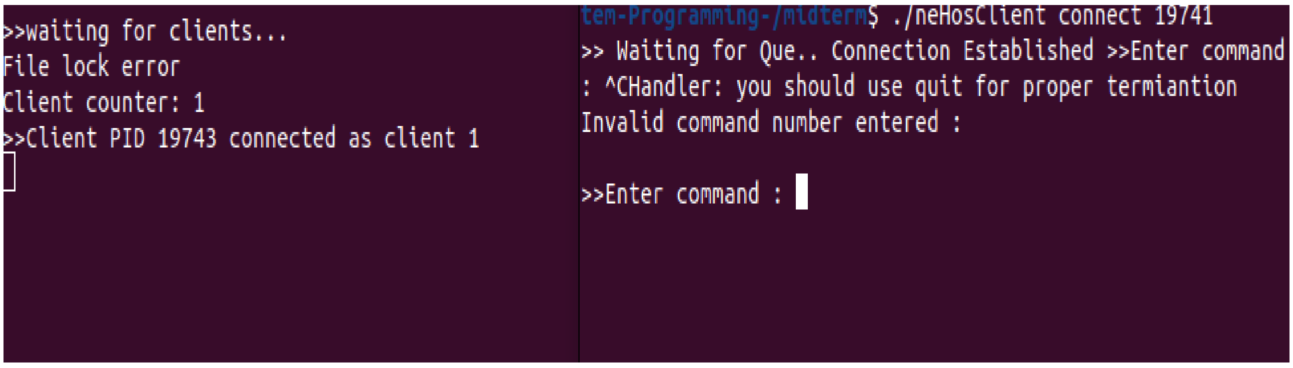
```
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$ ./neHosServer here 4
>> Server started PID 15172...
>>waiting for clients...
Client counter: 1
>>Client PID 15174 connected as client 1
>> killing server and all clients...
Terminated
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-Programming-/mid
term$
```

3.2.11 quit



```
>>Enter command : quit
Sending write request to server log file
waiting for log file
logfile write request granted
bye..
```

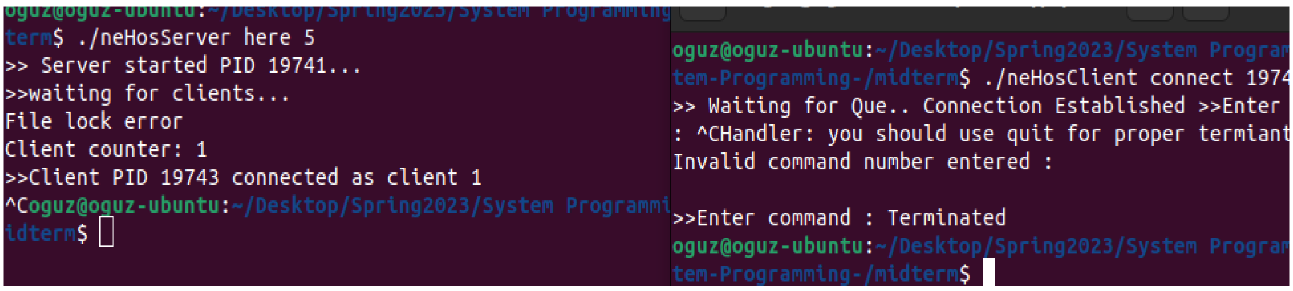
3.2.12 signalHandling



```
>>waiting for clients...
File lock error
Client counter: 1
>>Client PID 19743 connected as client 1
```

```
tem-Programming-/midterm$ ./neHosClient connect 19741
>> Waiting for Que.. Connection Established >>Enter command
: ^CHandler: you should use quit for proper termiantion
Invalid command number entered :

>>Enter command :
```



```
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming
term$ ./neHosServer here 5
>> Server started PID 19741...
>>waiting for clients...
File lock error
Client counter: 1
>>Client PID 19743 connected as client 1
^Coguz@oguz-ubuntu:~/Desktop/Spring2023/System Programmi
idterm$
```

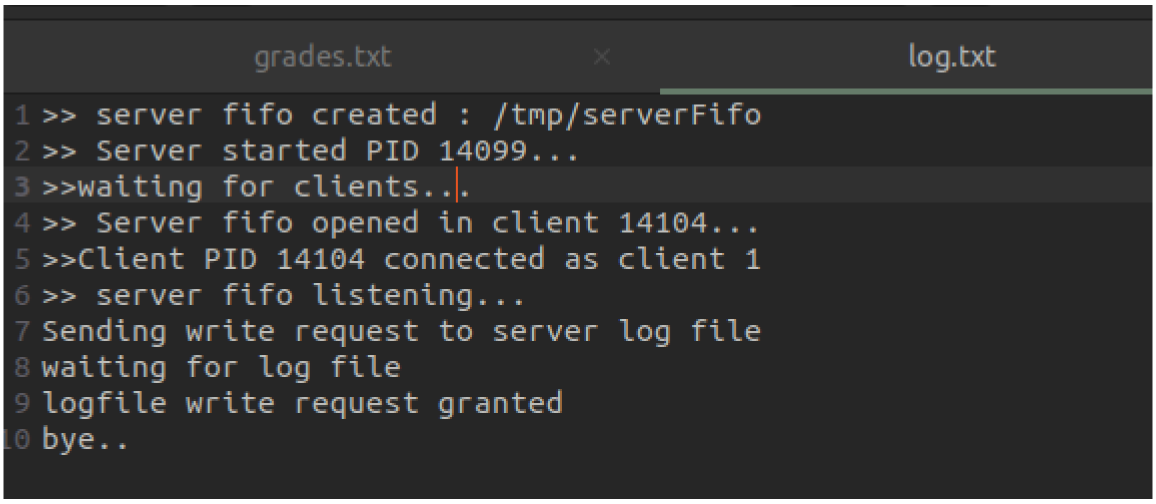
```
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Progran
tem-Programming-/midterm$ ./neHosClient connect 1974
>> Waiting for Que.. Connection Established >>Enter
: ^CHandler: you should use quit for proper termiant
Invalid command number entered :

>>Enter command : Terminated
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Progran
tem-Programming-/midterm$
```

3.2.13 waiting que (Could not achieved)

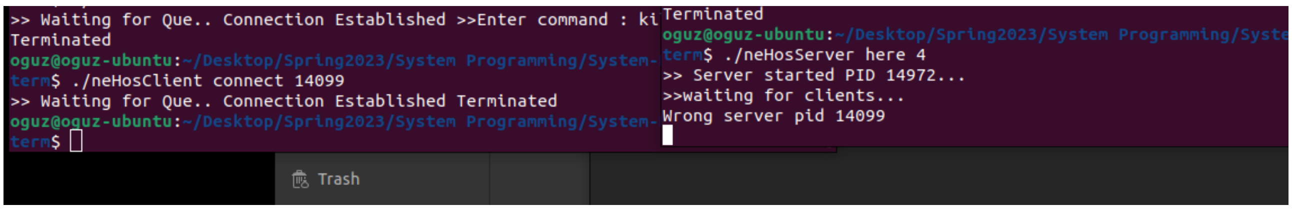
REASON : I thought of a design with semaphore for the "waiting que" structure. semaphore would start from the maximum number of clients, each new client would make sem_wait, it had to receive a separate response to prevent sem_wait from falling when try_connect was made, but this idea was not successful and I could not fix the problem where the program entered deadlock as soon as it started.

3.2.14 log file



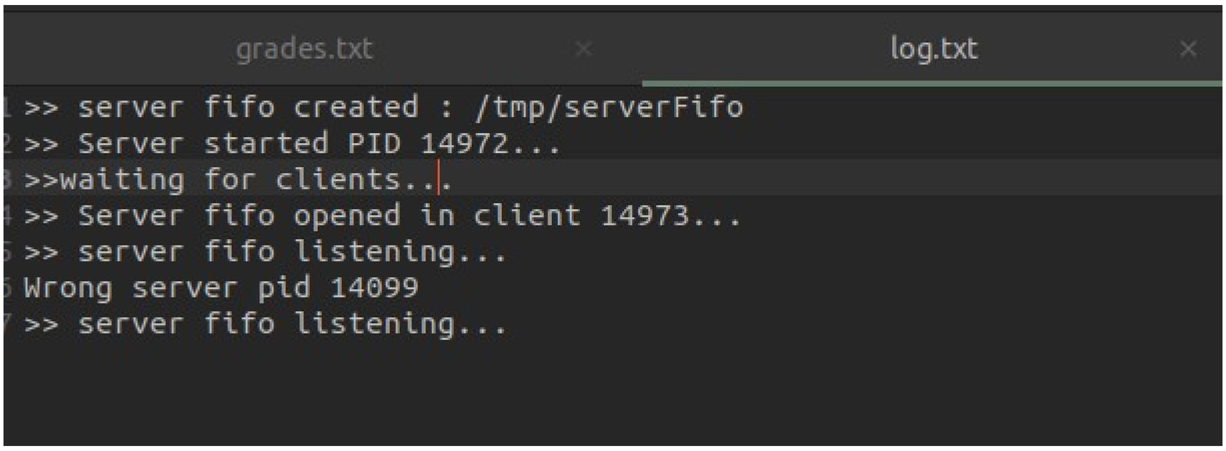
```
>> server fifo created : /tmp/serverFifo
>> Server started PID 14099...
>>waiting for clients...
>> Server fifo opened in client 14104...
>>Client PID 14104 connected as client 1
>> server fifo listening...
Sending write request to server log file
waiting for log file
logfile write request granted
bye..
```

3.2.15 multiple client and wrong id, flock

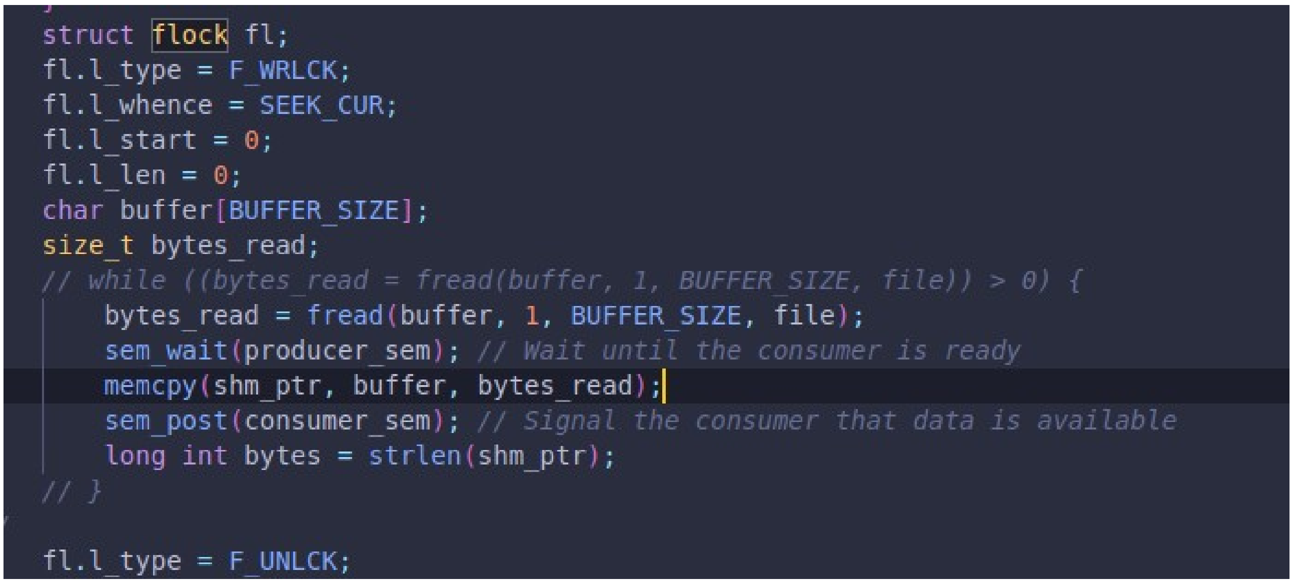


```
>> Waiting for Que.. Connection Established >>Enter command : ki
Terminated
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-
term$ ./neHosClient connect 14099
>> Waiting for Que.. Connection Established Terminated
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/System-
term$ 
```

```
Terminated
oguz@oguz-ubuntu:~/Desktop/Spring2023/System Programming/Syste
term$ ./neHosServer here 4
>> Server started PID 14972...
>>waiting for clients...
Wrong server pid 14099
```



```
>> server fifo created : /tmp/serverFifo
>> Server started PID 14972...
>>waiting for clients...
>> Server fifo opened in client 14973...
>> server fifo listening...
Wrong server pid 14099
>> server fifo listening...
```



```
struct flock fl;
fl.l_type = F_WRLCK;
fl.l_whence = SEEK_CUR;
fl.l_start = 0;
fl.l_len = 0;
char buffer[BUFFER_SIZE];
size_t bytes_read;
// while ((bytes_read = fread(buffer, 1, BUFFER_SIZE, file)) > 0) {
    bytes_read = fread(buffer, 1, BUFFER_SIZE, file);
    sem_wait(producer_sem); // Wait until the consumer is ready
    memcpy(shm_ptr, buffer, bytes_read);
    sem_post(consumer_sem); // Signal the consumer that data is available
    long int bytes = strlen(shm_ptr);
// }

fl.l_type = F_UNLCK;
```