

# Eren Torlak 21010404090 CSE344 Midterm Report

## Server Side:

- The server creates a server FIFO with pid
- The server waits for connection requests from clients by reading from the FIFO.
- When a connection request is received, the server checks if there is an available place for the client. If not, the client is added to a queue.
- If a slot is available, the server forks a new child process to handle the client's requests.
- The child process communicates with the client using a separate FIFO created for that client.
- The child process receives requests from the client, processes them, and sends responses back to the client.

## Client Side:

- The client creates a response FIFO before sending request as lecture notes.
- The client sends a connection request to the server by writing to the server's FIFO.
- If the server accepts the connection, the client receives a response indicating that it is connected.
- The client sends requests to the server by server fifo
- The client receives responses from the server by reading from its own FIFO.

## Communication Protocol:

- The server and client communicate using a structured data format, which includes fields such as the request type, client ID, and data buffers.
- The server and client use synchronization mechanisms, such as semaphores, to ensure that only one process can access a resource at a time.
- The server and client use signal handling to handle termination signals, such as SIGCHLD and SIGINT.

## Request Types:

- The client can send various types of requests to the server, including:
  - CONNECT: Request to connect to the server.
  - QUIT: Request to disconnect from the server.
  - LIST: Request to list the files in the server's directory.
  - READF: Request to read a file from the server's directory.
  - WRITET: Request to write to a file in the server's directory.
  - UPLOAD: Request to upload a file to the server's directory.
  - DOWNLOAD: Request to download a file from the server's directory.
  - ARCHSERVER: Request to archive the server's directory.

# Eren Torlak 21010404090 CSE344 Midterm Report

## Semaphore Initialization

Three semaphores are initialized:

**free\_place\_num**: initialized with the value of maxClients, which represents the number of available place for clients.

**occupied\_place\_num**: initialized with the value of 0, which represents the number of connected clients.

**connection\_requested**: initialized with the value of 1. It does handle connection requests safely.

## Handling CONNECT and TRY\_CONNECT

- **CONNECT:** The server checks for free slots. If there's a slot:
  - It connects the client immediately.
  - If not, the client is added to a queue until a slot becomes available.
  - This uses **sem\_wait()**
- **TRY\_CONNECT:**
  - The client sends a request and expects an immediate response.
  - If no slots are available at the moment of request, the server rejects the connection attempt.

## readF (Read File)

This operation reads content from a specified file on the server.

- **Steps:**
  1. The server constructs the full path to the file using the directory name and file path provided by the client.
  2. The server opens the file for reading.
  3. It reads chunks of the file and sends these chunks to the client until the entire file has been read.

### 2. writeT (Write to File)

The process begins with the client preparing a command and sending a request to the server. On receiving this request, the server initiates by creating a temporary file. It then proceeds to copy the contents of the original file—where new text is to be inserted—into this temporary file. Throughout this operation, a counter is utilized to monitor the progression through the lines of the file. Upon reaching the specified line number, the text provided by the client is inserted. Once the entire content has been successfully transferred to the temporary file, the original file is deleted. Subsequently, the temporary file is renamed to take the place of the original file. To ensure data integrity during this procedure, the file being modified is locked to prevent other processes from accessing it simultaneously. ensures client requests are handled effectively and safely.

### Upload

Clients can upload files to the server, which stores them in a specified directory.

- **Steps:**

1. The server receives the path and name of the file from the client.
2. It creates or opens the file in write mode.
3. File lock is initiated
4. The server then reads chunks of data from the client requests and writes these directly to the file until the upload is complete.
5. File unlocked
6. After finishing the file write, the server confirms the successful upload to the client.

```
lock_file(uploadFile);
if (fwrite(reqClient.buffer, sizeof(char),
          strlen(reqClient.buffer),
          uploadFile) != strlen(reqClient.buffer)) {
    fprintf(log_fp, "Error writing to the file.\n");
    break;
}
unlock_file(uploadFile);
```

## Download

- **Steps:**

1. The server constructs the path to the target file.
2. It opens the file for reading.
3. The server sends responses about if reading is done or not .
4. The process continues until the entire file has been sent.

## archServer

it does create directory.

Calls `execvp` with arguments

Renames it .

May not be working well.

## Eren Torlak 21010404090 CSE344 Midterm Report

Queue full case:

```
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosServer deneme 2
Server PID: 163
Logfile: deneme/server163.log created...
Waiting for clients...
>> Client PID 164 connected as client01
>> Client PID 166 connected as client02
Connection request PID 168... Queue FULL

(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosClient Connect 163
I am client number : 1 .
>> Enter command:

(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosClient Connect 163
I am client number : 2 .
>> Enter command:

(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosClient Connect 163
I am client number : 3 .
>> Enter command:
```

One client quited , the one in the queue joined.

```
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosServer deneme 2
Server PID: 163
Logfile: deneme/server163.log created...
Waiting for clients...
>> Client PID 164 connected as client01
>> Client PID 166 connected as client02
Connection request PID 168... Queue FULL
>> client02 disconnected..
>> Client PID 168 connected as client03
Child process with pid 0 terminated.

(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosClient Connect 163
I am client number : 1 .
>> Enter command:

(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosClient Connect 163
I am client number : 2 .
>> Enter command: quit
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ |

(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosClient Connect 163
I am client number : 3 .
>> Enter command: |
```

trtConnect

```
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosClient tryConnect 163
I am client number : 5 .
Could not connect to the server... The queue is full
asdadsad .
```

## Eren Torlak 21010404090 CSE344 Midterm Report

```
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosClient C
connect 163
I am client number : 1 .
>> Enter command: list
server163.log

>> Enter command: |
```

### Download

Midterm\_system

Ad	Değiştirme tarihi
deneme	5.05.2024 10:12
Makefile	5.05.2024 10:06
neHosClient	5.05.2024 10:10
neHosClient.c	5.05.2024 10:10
neHosServer	5.05.2024 10:10
neHosServer.c	5.05.2024 10:05
server163.log	5.05.2024 10:16

```
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosClient C
connect 163
I am client number : 1 .
>> Enter command: list
server163.log

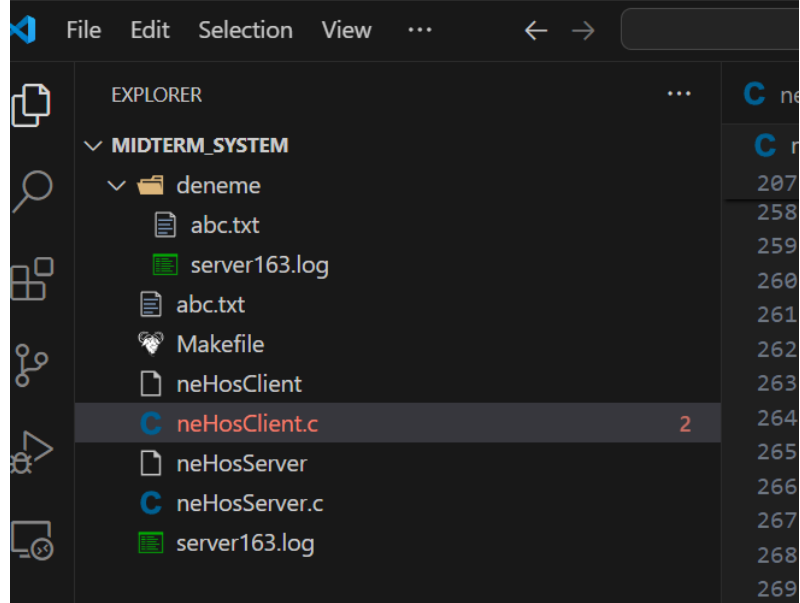
>> Enter command: download server163.log
Download started...
Download finished...
>> Enter command:
```

## Eren Torlak 21010404090 CSE344 Midterm Report

Upload

```
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/eren
connect 163
I am client number : 1 .
>> Enter command: list
server163.log

>> Enter command: download server163.log
Download started...
Download finished...
>> Enter command: upload abc.txt
Upload started
Upload finished
>> Enter command:
```



The screenshot shows a file explorer window with the following structure:

- EXPLORER
  - MIDTERM\_SYSTEM
    - deneme
      - abc.txt
      - server163.log
      - abc.txt
      - Makefile
      - neHosClient
      - neHosClient.c (highlighted in red)
      - neHosServer
      - neHosServer.c
      - server163.log

readF

```
>> Enter command: readF abc.tx
birinci satir
ikinci satir
ucuncu satir
>> Enter command: |
```

## Eren Torlak 21010404090 CSE344 Midterm Report

writeT

```
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosClient Connect 163
I am client number : 3 .
>> Enter command: help
Possible list of client requests:
list
readF <file> <line#>
writeT <file> <line#> <string>
upload <file>
download <file>
quit
killServer
>> Enter command: writeT abc.txt 2 eren
Writing finished.
>> Enter command:
```

help

```
esktop/Midterm_system$ ./neHosClient Connect 163
I am client number : 3 .
>> Enter command: help
Possible list of client requests:
list
readF <file> <line#>
writeT <file> <line#> <string>
upload <file>
download <file>
quit
killServer
```

arcServer

```
MIDTERM_SYSTEM
deneme
archive
archive.tar
abc.txt
server163.log
abc.txt
Makefile
neHosClient
neHosClient.c
OUTLINE
TIMELINE

neHosServer.c > request >
250 int main(int arg
698
699
700
701
702
703
704
705
706
707
708

readF <file> <line#>
writeT <file> <line#> <string>
upload <file>
download <file>
quit
killServer
>> Enter command: writeT abc.txt 2 eren
Writing finished.
>> Enter command: arcServer archive.tar
>> Enter command: list
abc.txt
server163.log
>> Enter command: arcServer archive.tar
Archive started...
>> Enter command:
```



# Eren Torlak 21010404090 CSE344 Midterm Report

## killServer

```
Connection request PID 180... Queue FULL
Failed to send kill signal to pid 164
Failed to send kill signal to pid 166
Failed to send kill signal to pid 180
bye...
Child process with pid -1 terminated.
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ |

ucuncu satir
>> Enter command: writeT
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ ./neHosClient t
ryConnect 163
I am client number : 4 .
>> Enter command: killServer
>> Enter command: Server is terminated...
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/Midterm_system$ |
```