

GEBZE
TEKNİK ÜNİVERSİTESİ



SPRING 2023 – CSE 344 SYSTEM PROGRAMMING

HOMEWORK 4

MERT GÜRŞİMŞİR

1901042646

System Architecture, Design Decisions & Implementation Details

To understand the needs of this project we need to know the concepts of inter process communication (IPC), POSIX threads, file input output, signals, and synchronization. In the midterm, I was creating a process for each child. Now, I am creating thread pool and they are waiting for tasks. Each thread is doing these in an infinite loop:

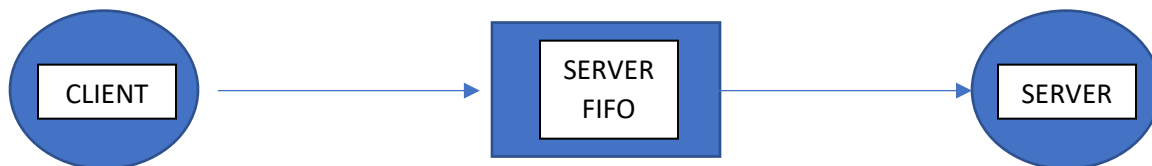
- Waiting and getting task from task queue
- Executing the task

For IPC, I have used FIFO for my system. First of all, high-level architecture has to be designed. The maximum number of client processes has been given to the system as input from the user. Semaphores are used to run at most given number of clients.

- Wait semaphore when new client is connected.
- Post semaphore when you done with client.

With this design, system is able to run at most maximum number of clients at the same time.

Other important issue that is need to be talked about is design of FIFOs. Different than the midterm, I didn't use any client FIFO for request. I have used server fifo for all the requests including connections. To connect to the server, server FIFO is used with the same name as server's process id:



Client writes one end of this FIFO while server reads other end all the time. Whenever new request (that consists of client's current working directory, its pid, and request message) comes from the client, server adds this request to task queue and one of the threads get the first task from the queue, executes it and sends result back to the client.

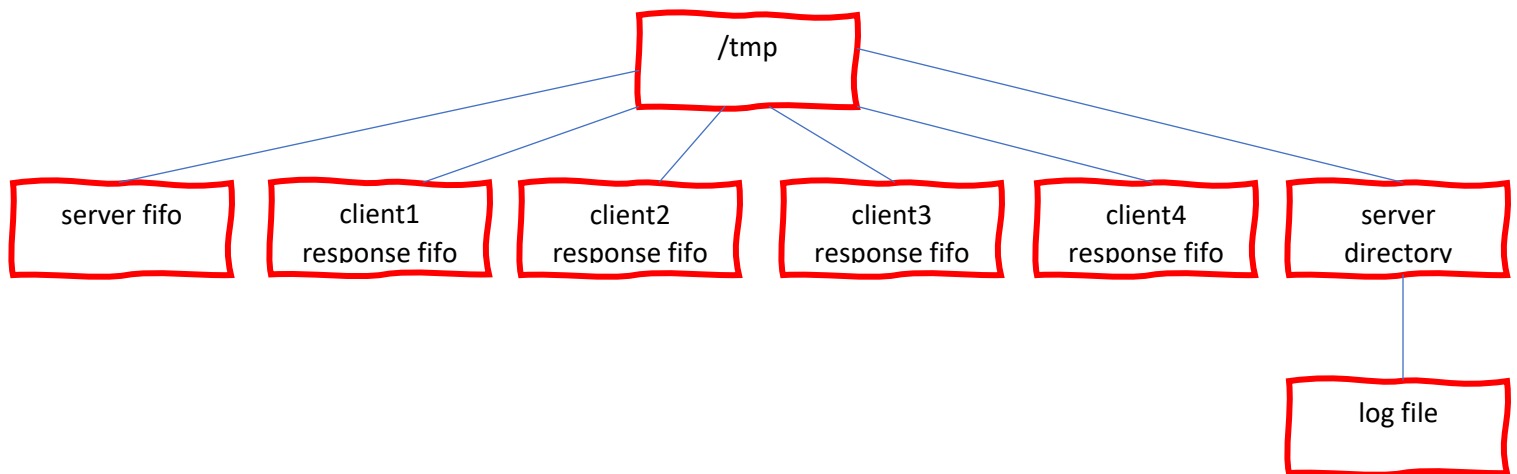
For sending result back to the client, I have created a FIFO with the name of the client's pid:



Client is sending all of its requests (including connection requests) by server FIFO and get all the response from the its own client FIFO.

For synchronization, I have used semaphores. Whenever a file job is going to be done, code waits for the binary semaphore and at the end, semaphore is posted.

Architecture is mostly based on /tmp directory. Server's directory is being created here. Server's FIFO and client's FIFO are also here. Client access to /tmp/{ServerPID} file which is FIFO to connect to the server.



Client FIFOs can be as many as possible in theory.

Client program can be executed wherever user wants. All the system are on top of /tmp directory so proper name alignments are being done for the program.

For the threads, firstly I am creating given poolSize number of threads that are detachable so I will not join them. Then, main thread starts looping infinitely. It is doing these in the loop:

- Get request from the server FIFO
- Create a task according to request
- Submit the task

Submitting the task is easily adding task to the task queue. I am using mutex locks while I am adding to task to the queue and at the end, I am signaling condition variable to say that there is a task at the queue.

Each thread is doing these:

- Waiting and getting task from task queue → critical region so it is protected by mutex
- Executing the task

Executing the task consists these steps:

- Open the client FIFO for sending response
- Provide proper connection mechanism for clients
- Implement the command
- Send back the result through client FIFO

I am using semaphore for connection for maximum client size. Waiting for the semaphore is request is connection, and posting the semaphore if request is quitting.

Signal handler for server is doing these:

- Destroy mutexes.
- Writes all of the child process (clients) information to the log file and then deletes the clients FIFOs.
- Closes all the file descriptors.
- Frees dynamically allocated spaces.
- Destroy semaphores.
- Deletes the server FIFO.

Threads can access global space. They share that place. So I don't use shared memory as I did at midterm project.

For client requests, my implementations are goes like these:

- readF
 - If line number is not given, read the whole file and assign it to an array that is dynamically allocated.
 - If line number is given, read the file byte by byte and count the '\n' characters. If wanted line is reached, read it to buffer.
- writeT
 - If line is not given, write directly to the end by opening the file with O_APPEND flag.
 - Get to the line if it is given by reading the file, save what comes next until end of the file, then write given string and then write again the saved part of the file.
- upload/download
 - These 2 requests simply opens the file in the corresponding directory, read it to dynamically allocated buffer, then writes it to the file that is newly created with same name to the corresponding directory.

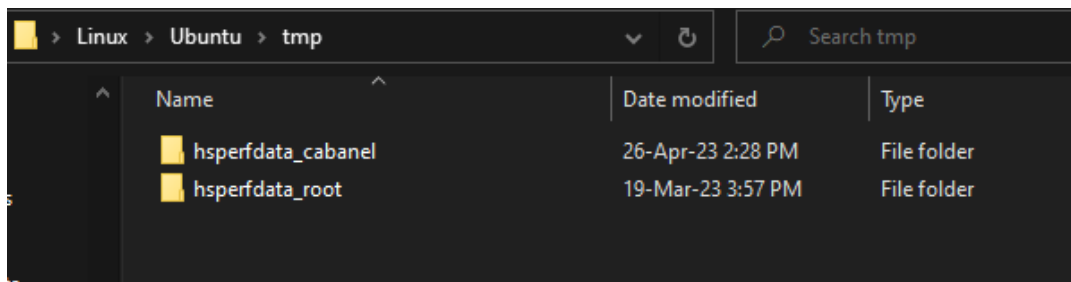
Tests

To be sure that this system works properly, these parts should be tested:

- Can client and server communicate properly?
- Can client that exceeds the maximum client number wait for one of other processes to finish?
- Can waiting client connect to the server when there is enough place for it?
- Can server serve to many clients which are connected to itself at the same time?
- Can client display the requests that it can perform and learn the details of them?
- Can client see the list of files in server's directory?
- Can client read file that is at the server side?
- Can client write to file that is at the server side?
- Can client upload a file to the server's directory?
- Can client download a file from server's directory?
- Can log file be created and filled properly?
- Can client quit from the server that it is connected to?
- Can client sends a kill request to server?
- Can server handle the signal that tries to kill it? It needs to perform some operations?
- Can client connect with tryConnect when queue is empty?
- Does client wait when it tries to connect with tryConnect when queue is full?

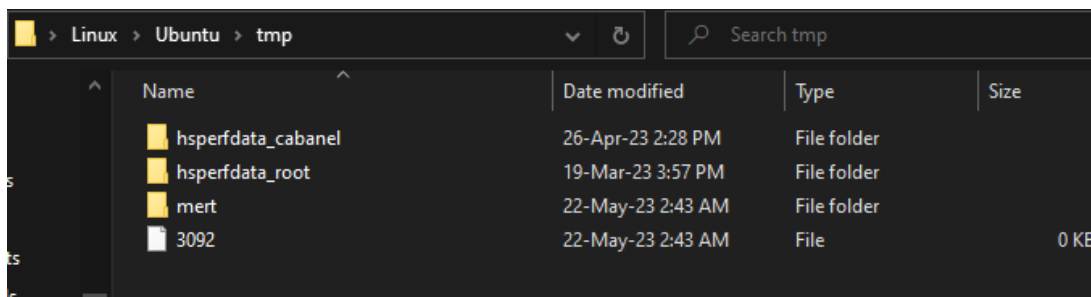
1) Can client and server communicate properly?

/tmp directory before running the code:



| Name | Date modified | Type |
|--------------------|-------------------|-------------|
| hsperfdata_cabanel | 26-Apr-23 2:28 PM | File folder |
| hsperfdata_root | 19-Mar-23 3:57 PM | File folder |

/tmp directory after running the biboServer.c:



| Name | Date modified | Type | Size |
|--------------------|-------------------|-------------|------|
| hsperfdata_cabanel | 26-Apr-23 2:28 PM | File folder | |
| hsperfdata_root | 19-Mar-23 3:57 PM | File folder | |
| mert | 22-May-23 2:43 AM | File folder | |
| 3092 | 22-May-23 2:43 AM | File | 0 KB |

3092 is the pid of the server.

```
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboServer mert 1 5
Server Started PID 3092...
waiting for clients...
```

Now server is waiting for clients to connect to it. Let's connect:

```
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboServer mert 1 5
Server Started PID 3092...
waiting for clients...
Client PID 3103 connected as "client1"

cabanel@MSI: /mnt/c/Users/Mert/Desktop
>> Waiting for Que...Killed
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboClient Connect 3092
>> Waiting for Que...Connection established
>>
```

As you can see client and server communicated and client is connected to server.

2) Can client that exceeds the maximum client number wait for one of other processes to finish?

As you can see from the previous example, we have chosen maximum client number as 1. So next client that connects to the server will have to wait for the process with pid 3103.

```
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboServer mert 1 5
Server Started PID 3092...
waiting for clients...
Client PID 3103 connected as "client1"
Connection request PID 3105... Que FULL

cabanel@MSI: /mnt/c/Users/Mert/Desktop
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboClient Connect 3092
>> Waiting for Que...Connection established
>> _

cabanel@MSI: /mnt/c/Users/Mert/Desktop
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboClient Connect 3092
>> Waiting for Que...
```

New client is waiting for the queue to be empty. When process with pid 3103 finishes its execution, waiting process can communicate with server.

3) Can waiting client connect to the server when there is enough place for it?

```
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboServer mert 1 2
Server Started PID 3116...
waiting for clients...
Client PID 3119 connected as "client1"
Connection request PID 3121... Que FULL
client 3119 disconnected...
Client PID 3121 connected as "client2"

cabanel@MSI:/mnt/c/Users/Mert/Desktop$
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboClient Connect 3116
>> Waiting for Que...Connection established

>> quit
bye
cabanel@MSI:/mnt/c/Users/Mert/Desktop$

cabanel@MSI:/mnt/c/Users/Mert/Desktop$
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboClient Connect 3116
>> Waiting for Que...Connection established

>>
```

As it is seen, when client 1 disconnects, client 2 connects to the server.

Also this execution answers the question: "Can client quit from the server that it is connected to?". It can quit and disconnected message is shown at server side immediately. Then other clients may be served.

4) Can server serve to many clients which are connected to itself at the same time?

```
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboServer mert 2 2
Server Started PID 3122...
waiting for clients...
Client PID 3125 connected as "client1"
Client PID 3126 connected as "client2"
```

```
cabanel@MSI:/mnt/c/Users/Mert/Desktop
>> help

Available commands are:
help, list, readF, writeT, upload, download, quit, killServer
```

```
>> _
```

```
cabanel@MSI:/mnt/c/Users/Mert/Desktop
>> list
.
..
LOG.txt
>>
```

Maximum number of clients is 2 now. So 2 clients can be connected to the server and server can serve to both of them.

With this execution we know the answer of questions “Can client display the requests that it can perform?” and “Can client see the list of files in server’s directory?”.

To be able to display details of requests, following format is used → “help {request}”

```
>> help list

list
    sends a request to display the list of files in Servers directory
    (also displays the list received from the Server)

>>
```

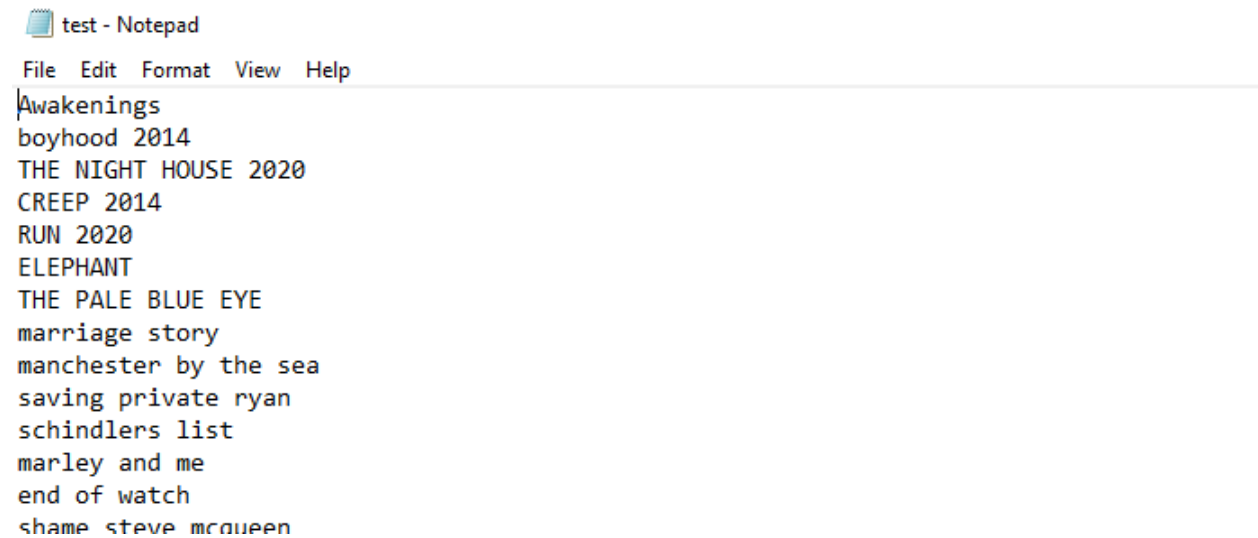
```
>> help upload

upload <file>
    uploads the file from the current working directory of client to the Servers directory
    (beware of the cases no file in clients current working directory and file with the same
    name on Servers side)
```

You can learn about all the details of all the requests like these.

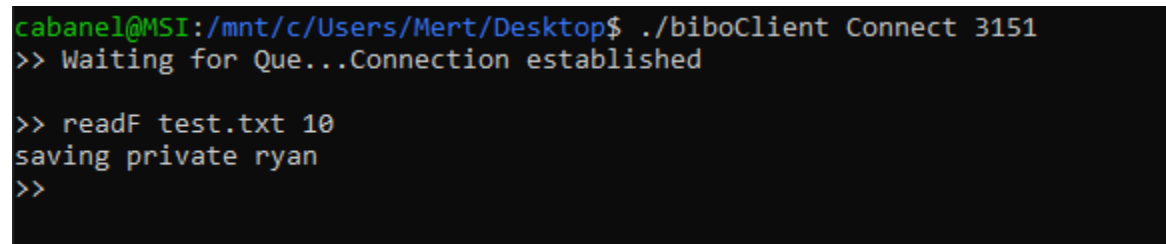
5) Can client read file that is at the server side?

For this test, a txt file is copied to the server directory.



```
test - Notepad
File Edit Format View Help
Awakenings
boyhood 2014
THE NIGHT HOUSE 2020
CREEP 2014
RUN 2020
ELEPHANT
THE PALE BLUE EYE
marriage story
manchester by the sea
saving private ryan
schindlers list
marley and me
end of watch
shame steve mcqueen
```

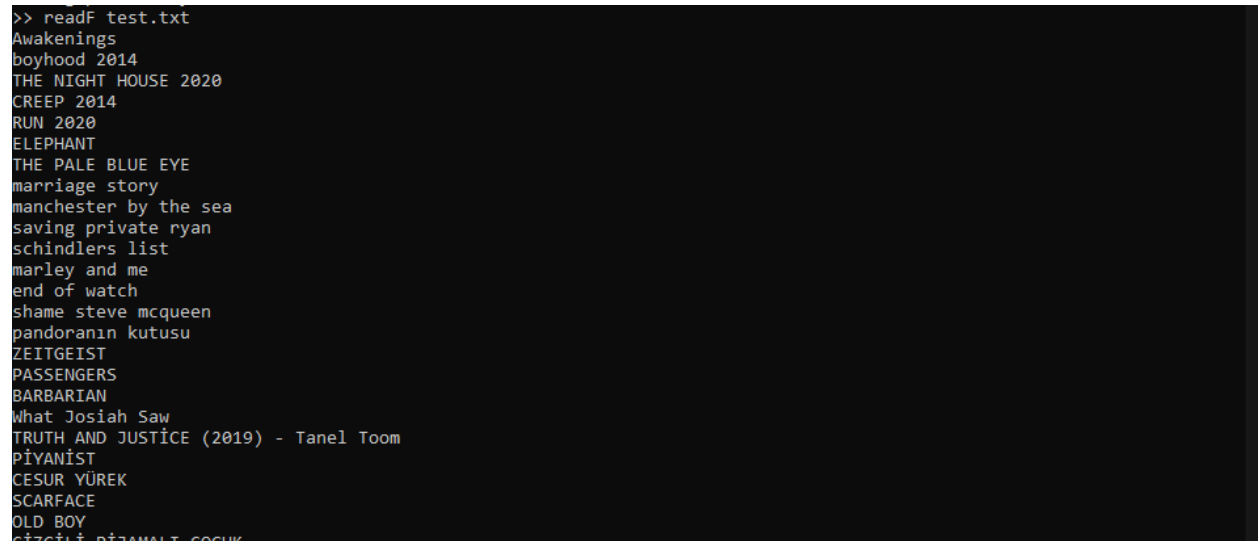
Let's read the 3rd line of this file:



```
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboClient Connect 3151
>> Waiting for Que...Connection established

>> readF test.txt 10
saving private ryan
>>
```

If you want, you can read the all file too:



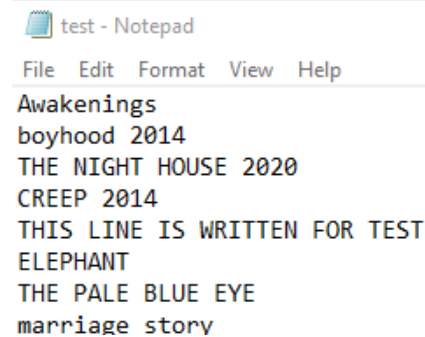
```
>> readF test.txt
Awakenings
boyhood 2014
THE NIGHT HOUSE 2020
CREEP 2014
RUN 2020
ELEPHANT
THE PALE BLUE EYE
marriage story
manchester by the sea
saving private ryan
schindlers list
marley and me
end of watch
shame steve mcqueen
pandoranin kutusu
ZEITGEIST
PASSENGERS
BARBARIAN
What Josiah Saw
TRUTH AND JUSTICE (2019) - Tanel Toom
PIYANIST
CESUR YÜREK
SCARFACE
OLD BOY
çizgi piyano çocuk
```

6) Can client write to file that is at the server side?

```
>> writeT test.txt 5 THIS LINE IS WRITTEN FOR TEST  
SUCCESS  
>>
```

“SUCCESS” message is returned to us from server.

After this request file becomes this:



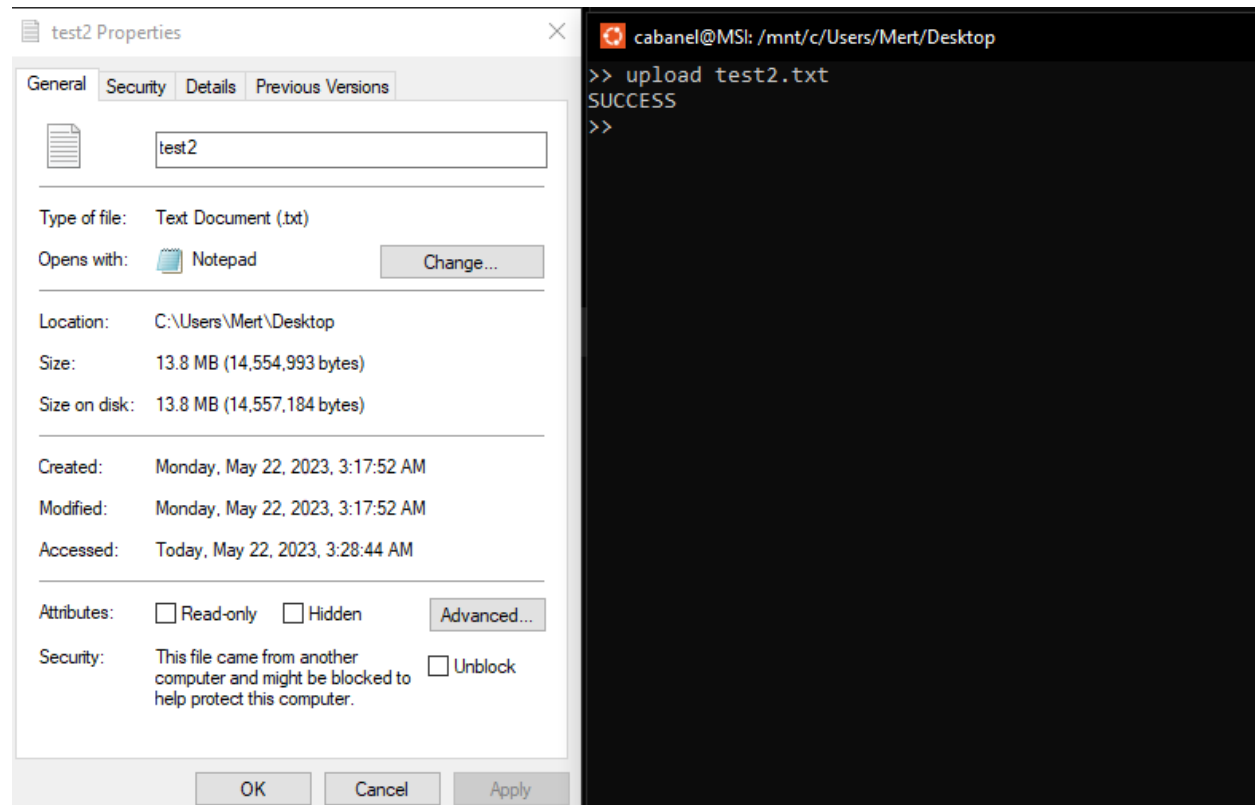
test - Notepad

File Edit Format View Help

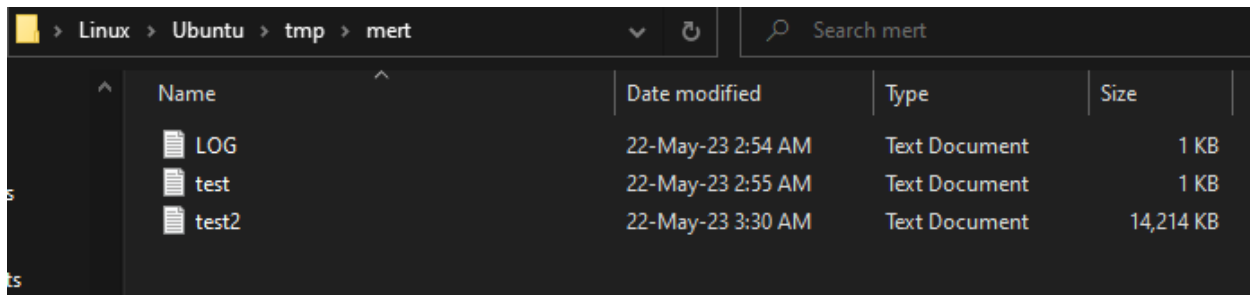
Awakenings
boyhood 2014
THE NIGHT HOUSE 2020
CREEP 2014
THIS LINE IS WRITTEN FOR TEST
ELEPHANT
THE PALE BLUE EYE
marriage story

7) Can client upload a file to the server's directory?

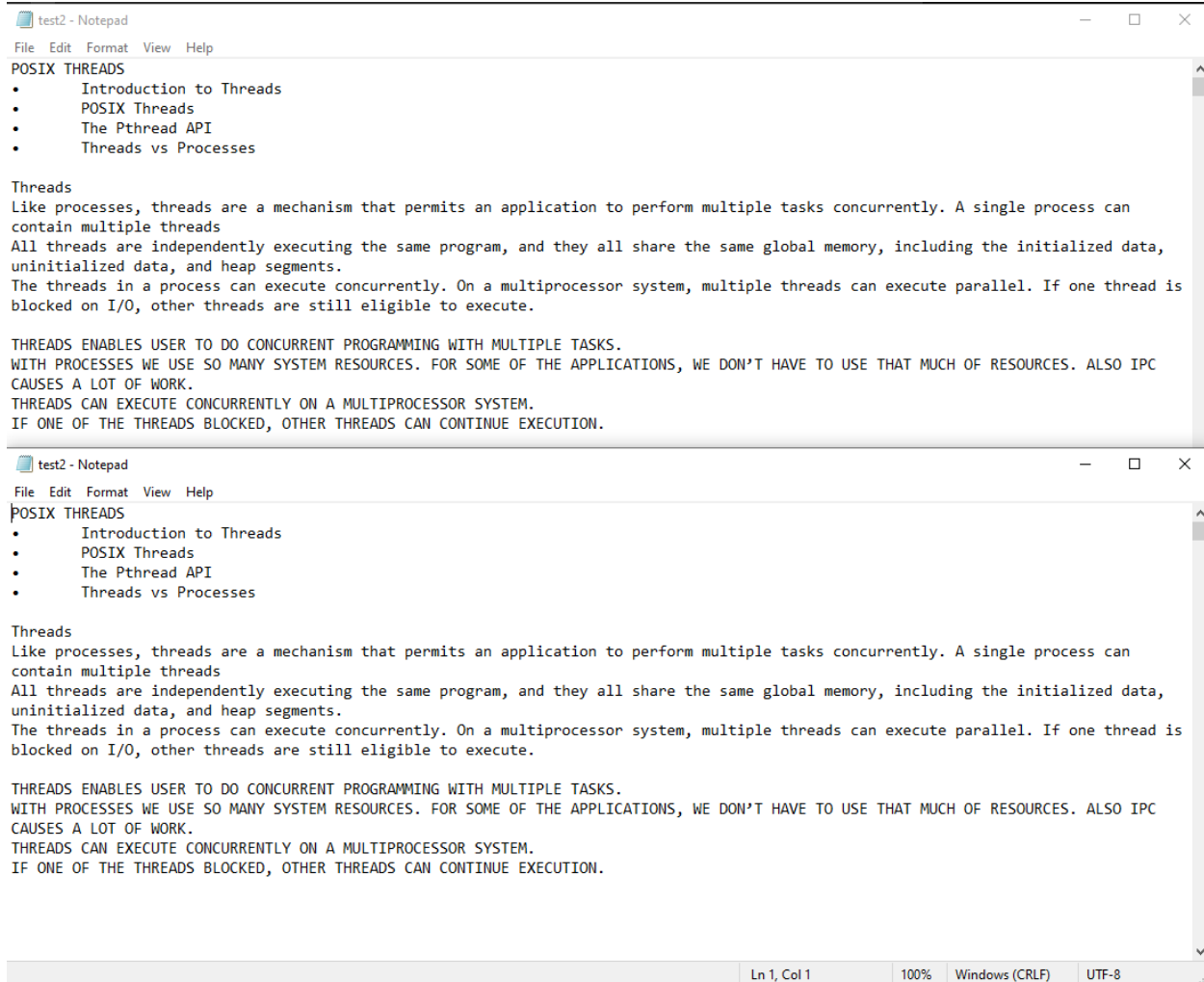
test2.txt is the file that is at the client's side.



When we upload it, we can see it at the server's side:

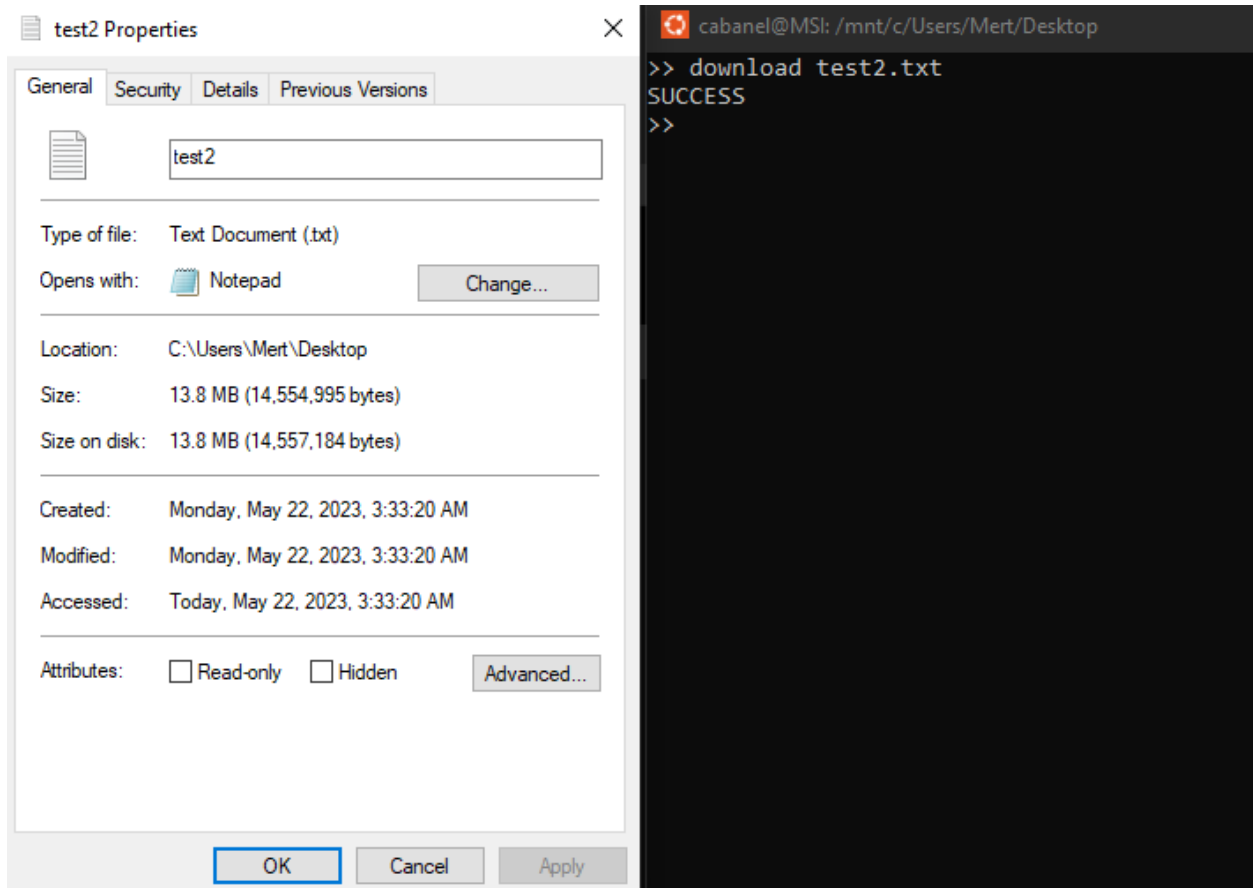


| Name | Date modified | Type | Size |
|-------|-------------------|---------------|-----------|
| LOG | 22-May-23 2:54 AM | Text Document | 1 KB |
| test | 22-May-23 2:55 AM | Text Document | 1 KB |
| test2 | 22-May-23 3:30 AM | Text Document | 14,214 KB |

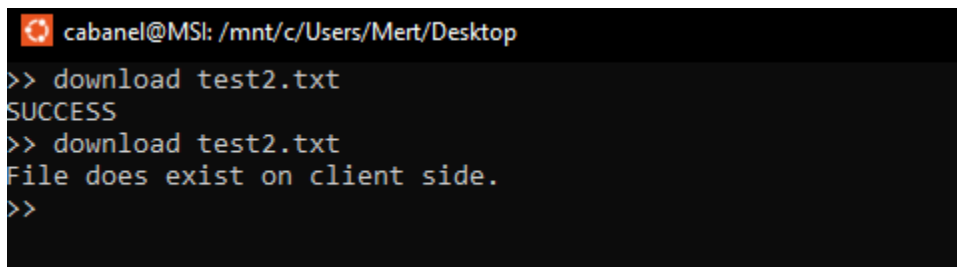


Contents are the same.

8) Can client download a file from server's directory?

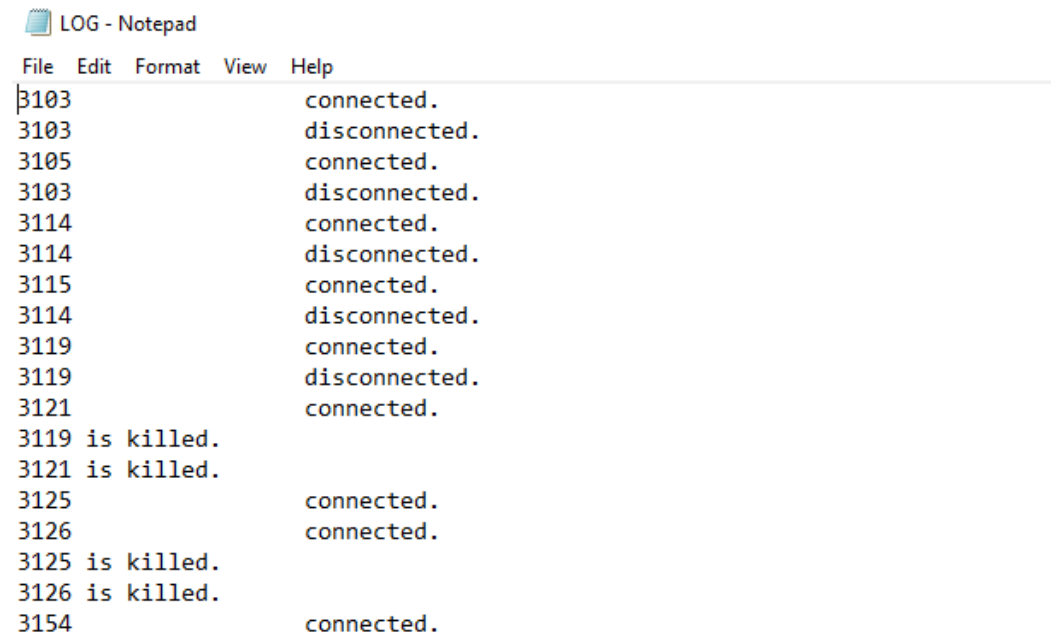


test2.txt is at the server's side but not in the client's side (I deleted it). If it exists at client's side too, we would get this message:



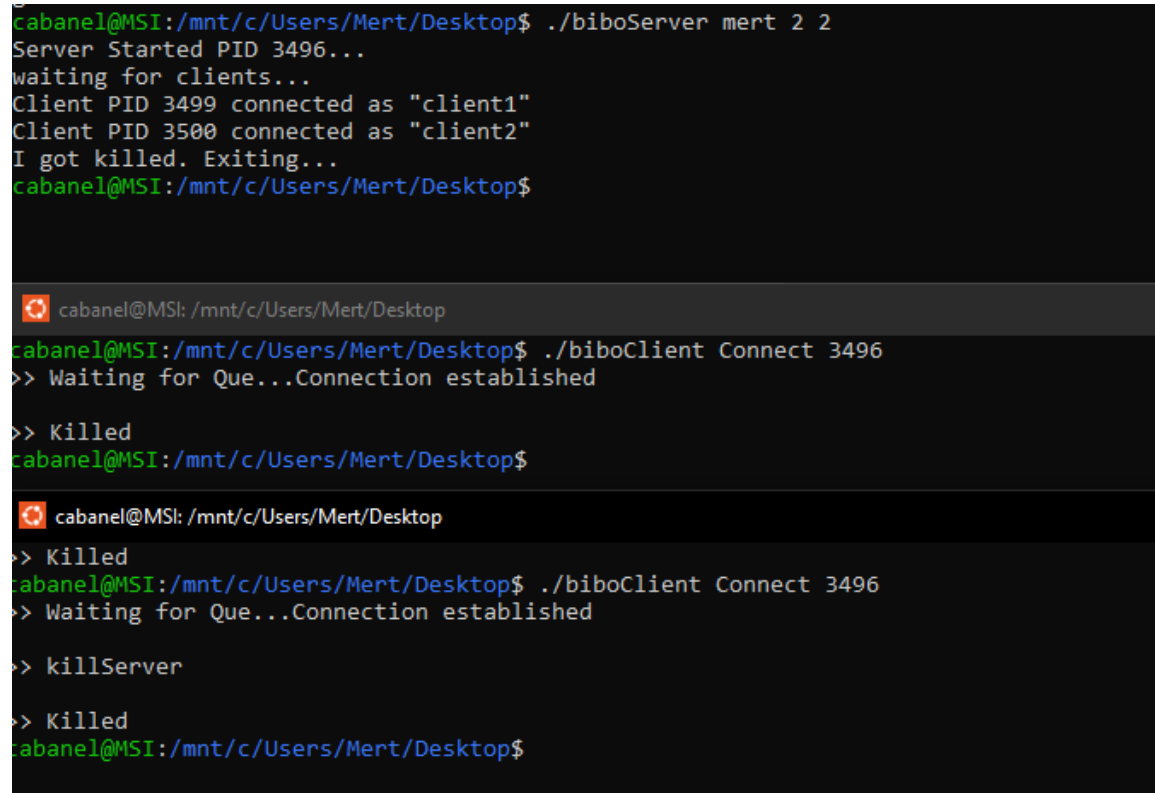
9) Can log file be created and filled properly?

At the log file, you can see all the processes that are connected, disconnected, or killed:



```
LOG - Notepad
File Edit Format View Help
3103 connected.
3103 disconnected.
3105 connected.
3103 disconnected.
3114 connected.
3114 disconnected.
3115 connected.
3114 disconnected.
3119 connected.
3119 disconnected.
3121 connected.
3119 is killed.
3121 is killed.
3125 connected.
3126 connected.
3125 is killed.
3126 is killed.
3154 connected.
```

10) Can client sends a kill request to server?



```
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboServer mert 2 2
Server Started PID 3496...
waiting for clients...
Client PID 3499 connected as "client1"
Client PID 3500 connected as "client2"
I got killed. Exiting...
cabanel@MSI:/mnt/c/Users/Mert/Desktop$

cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboClient Connect 3496
>> Waiting for Que...Connection established

>> Killed
cabanel@MSI:/mnt/c/Users/Mert/Desktop$

cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboClient Connect 3496
>> Waiting for Que...Connection established

>> killServer

>> Killed
cabanel@MSI:/mnt/c/Users/Mert/Desktop$
```

As it can be seen, when killServer request is sent, server got killed and all other processes are killed too.

Server needs to perform some operations when it got the kill signal.

There are several things that are needed to be done. All these things are discussed at page 4. Code snippet for handler looks like this:

```
void sigint_handler(int signal) {
    printf("I got killed. Exiting...\n");
    int i;
    int fd;
    char pid[50];

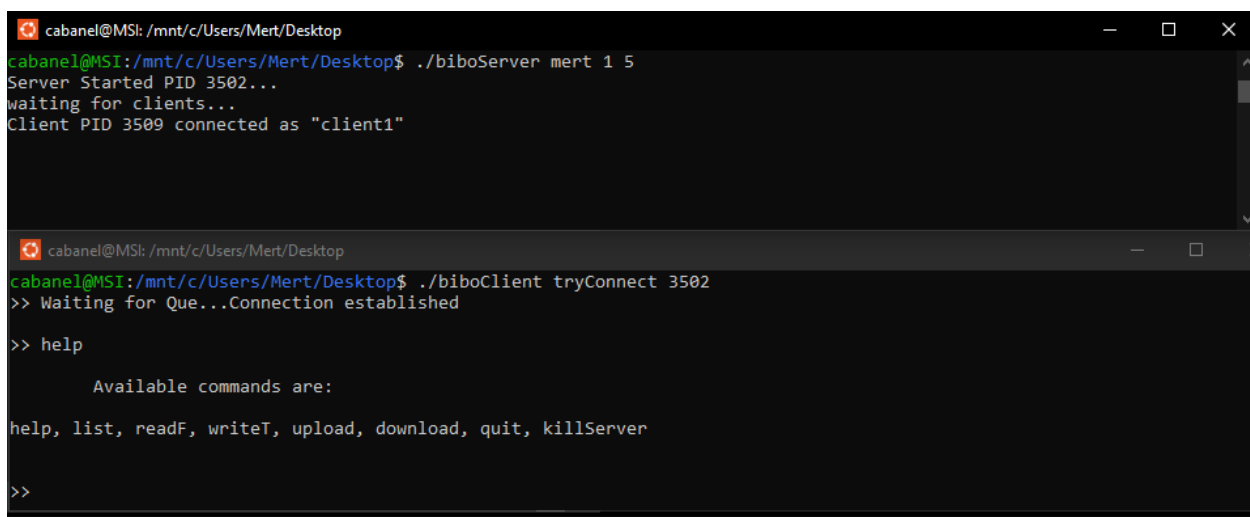
    pthread_mutex_destroy(&mutexQueue);
    free(th);

    for (i = 0; i < pid_pos; ++i){
        kill(childPIDs[i], SIGKILL);
        sprintf(pid, "%d", childPIDs[i]);
        write(logFile, pid, sizeof(pid_t));
        write(logFile, " is killed.\n", 12);
        sprintf(pid, "/tmp/%d_res", childPIDs[i]);
        unlink(pid);
    }
    for (fd = 0; fd < getdtablesize(); fd++) {
        close(fd);
    }
    if (buffer!=NULL) free(buffer);
    if (new_buffer!=NULL) free(new_buffer);
    if (bufferTemp.string!=NULL) free(bufferTemp.string);

    sem_destroy(&sem);
    sem_destroy(&semF);
    unlink(fifodir);

    exit(signal);
}
```

11) Can client connect with tryConnect when queue is empty?



The screenshot shows two terminal windows. The top window is the server's terminal, where the command `./biboServer mert 1 5` has been executed. The output shows the server starting with PID 3502, waiting for clients, and then a client with PID 3509 connecting as "client1". The bottom window is the client's terminal, where the command `./biboClient tryConnect 3502` has been executed. The output shows the client waiting for a queue, then connecting successfully. The client then enters the `help` command, which lists available commands: `help, list, readF, writeT, upload, download, quit, killServer`.

```
cabanel@MSI: /mnt/c/Users/Mert/Desktop
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboServer mert 1 5
Server Started PID 3502...
waiting for clients...
Client PID 3509 connected as "client1"

cabanel@MSI: /mnt/c/Users/Mert/Desktop
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboClient tryConnect 3502
>> Waiting for Que...Connection established

>> help

    Available commands are:
help, list, readF, writeT, upload, download, quit, killServer

>>
```

Client can connect with tryConnect if there is space.

```
cabanel@MSI: /mnt/c/Users/Mert/Desktop
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboServer mert 1 5
Server Started PID 3502...
waiting for clients...
Client PID 3509 connected as "client1"

cabanel@MSI: /mnt/c/Users/Mert/Desktop
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboClient tryConnect 3502
>> Waiting for Que...Connection established

>> help

    Available commands are:
help, list, readF, writeT, upload, download, quit, killServer

>>

cabanel@MSI: /mnt/c/Users/Mert/Desktop
cabanel@MSI:/mnt/c/Users/Mert/Desktop$ ./biboClient tryConnect 3502
>> Waiting for Que...Que is full. Goodbye.
cabanel@MSI:/mnt/c/Users/Mert/Desktop$
```

Other client cannot connect because queue is full.