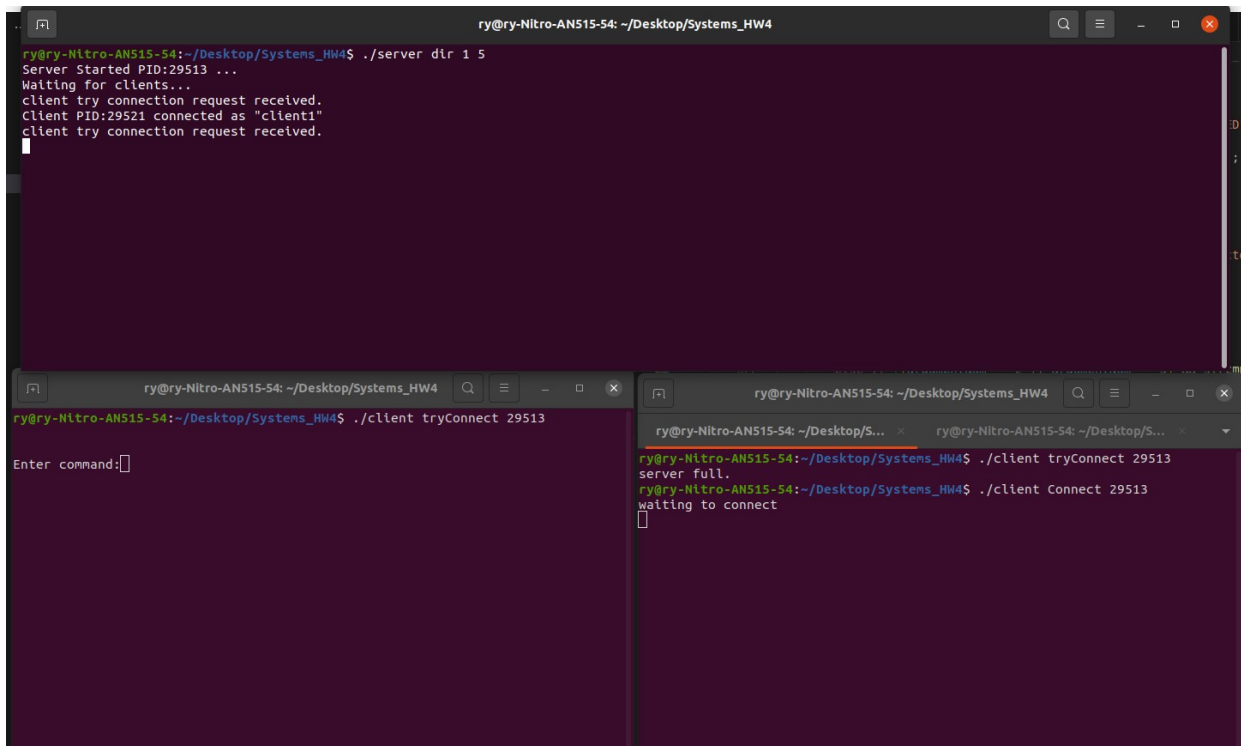


# HW4 REPORT

In this homework, I used some part of the midterm's code but the code structure is a little bit different from midterm because I used FIFO for inter process communication. Also, to synchronize write and read operations on files, when a file is read or written then the writer or reader count of the file is increased and blocked with semaphores as we saw in the lecture.

Besides all these, the difference of this homework is threads. Threads are used instead of creating child processes. To my understanding, a thread pool should be created with given size, when there is a request from a connected client, a thread should be assigned to the request and give response to the request, then the thread should be able to handle another request. In my program, thread pool is created instantly and clients send their requests to server, server stores the requests in a queue and notifies the threads that a task is added to queue,(as a structure Task which consist of message and pid of the client) meantime one of the free threads gets the task and sends response to the client. Because of this implementation, even if server does not have as much threads as connected client, server can still serve maybe slower but it is better than nothing.

## Screen Shots Of Tests



The image displays three terminal windows from a Linux environment, showing the execution of a server and client program. The top window shows the server starting and accepting connections. The bottom-left window shows a client attempting to connect using the 'tryConnect' option. The bottom-right window shows a client attempting to connect using the 'Connect' option, which results in a 'server full' error and a 'waiting to connect' message.

```
ry@ry-Nitro-AN515-54: ~/Desktop/Systems_HW4
ry@ry-Nitro-AN515-54:~/Desktop/Systems_HW4$ ./server dir 1 5
Server Started PID:29513 ...
Waiting for clients...
client try connection request received.
Client PID:29521 connected as "client1"
client try connection request received.
```

```
ry@ry-Nitro-AN515-54:~/Desktop/Systems_HW4$ ./client tryConnect 29513
Enter command:
```

```
ry@ry-Nitro-AN515-54:~/Desktop/Systems_HW4$ ./client tryConnect 29513
server full.
ry@ry-Nitro-AN515-54:~/Desktop/Systems_HW4$ ./client Connect 29513
waiting to connect
```

This shows that tryConnect and Connect options works, when there is a place tryConnect connects, otherwise immediately leaves the program, but Connect waits till there is a place available.

```
ry@ry-Nitro-AN515-54: ~/Desktop/Systems_HW4
ry@ry-Nitro-AN515-54:~/Desktop/Systems_HW4$ ./server dir 1 5
Server Started PID:29513 ...
Waiting for clients...
client try connection request received.
Client PID:29521 connected as "client1"
client try connection request received.
client1 disconnected
Client PID:29549 connected as "client2"
[]

ry@ry-Nitro-AN515-54:~/Desktop/Systems_HW4$ ./client tryConnect 29513
Enter command:quit
leaving...
ry@ry-Nitro-AN515-54:~/Desktop/Systems_HW4$

ry@ry-Nitro-AN515-54:~/Desktop/Systems_HW4$ ./client Connect 29513
server full.
ry@ry-Nitro-AN515-54:~/Desktop/Systems_HW4$ ./client Connect 29513
waiting to connect

Enter command:[]
```

This shows when a client waits and someone leaves the server, waiting client connects to server.

```
ry@ry-Nitro-AN515-54: ~/Desktop/Systems_HW4
waiting to connect

Enter command:^C
Kill signal received, terminating...
ry@ry-Nitro-AN515-54:~/Desktop/Systems_HW4$ ./client Connect 30557
waiting to connect

Enter command:readF testfile.txt
line1

line5

line9

Enter command:[]

ry@ry-Nitro-AN515-54:~/Desktop/Systems_HW4$ ./client Connect 30557
waiting to connect

Enter command:readF testfile.txt 5
line5

Enter command:[]
```

ReadF reads whole file or requested line.

```
Enter command:readF testfile.txt
line1

line5

line9

new line

Enter command:[]

Enter command:readF testfile.txt 5
line5

Enter command:writeT testfile.txt new line
Successfully written.

Enter command:[]
```

*then read*

*first write*

WriteT writes to end of the file if no line number is given.

```
Enter command:readF testfile.txt 7
line 7

Enter command:
111      if (FrontOfPidQueue() != -1)
```

```
Enter command:writeT testfile.txt 7 line 7
Successfully written.

Enter command:
```

If line number is specified, writeT writes to that line.

SYSTEMS\_HW4

.vscode

settings.json

dir

bg.jpeg

log\_31408

testfile.txt

bg.jpeg

bg2.jpeg

client

client.c

client.o

file.txt

functions.h

makefile

myconst.h

mytypes.h

pidqueue.h

server

server.c

server.o

taskqueue.h


Enter command:download bg.jpeg  
Download finished successfully.

Enter command:download bg.jpeg  
File exists in your directory.Try changing the filename or use overw  
ite option.  
new file name usage: download <filename> <newfilename>  
overwrite usage: download <fileName> -ow

Enter command:download bg.jpeg -ow  
Download finished successfully.

Enter command:download bg.jpeg bg2.jpeg  
Download finished successfully.

Enter command:



First try

try again

overwrite

change name

SYSTEMS\_HW4

.vscode

settings.json

dir

background.jpeg

backgroundv2.jpeg

log\_31408

testfile.txt

background.jpeg

client

client.c

client.o

file.txt

functions.h

makefile

myconst.h

mytypes.h

pidqueue.h

server

server.c

server.o

taskqueue.h


Enter command:upload background.jpeg  
File uploaded

Enter command:upload background.jpeg  
File with this name exists in server directory.

Enter command:upload background.jpeg -ow  
File uploaded

Enter command:upload background.jpeg backgroundv2.jpeg  
File uploaded

Enter command:



First try

try again

overwrite

change name



Download and upload works, when name conflict occurs, then error is shown. But if -ow is given as third argument then the file is overwritten. Also new file name can be given as third parameter to change the downloaded or uploaded file name.

The image displays three terminal windows on a Kali Linux desktop, illustrating a network communication and control sequence:

- Left Terminal (Client Execution):** Shows the compilation of `client.c` and its execution. The client connects to the server at `127.0.0.1` on port `42305`. It receives a `Kill` signal from the server and responds with `kill signal received, terminating...`
- Middle Terminal (Server Execution):** Shows the execution of `client.c` as a server. It listens on port `42305` and accepts a connection from the client. It sends a `Kill` signal to the client and receives a response.
- Right Terminal (Kill Process Execution):** Shows the execution of `kill.c`. It sends a `Kill` signal to the client at `127.0.0.1` on port `42305` and receives a response.

When ctrl-c (SIGINT) is received on server side, then all the clients and server is killed.

The screenshot displays three terminal windows on a Kali Linux desktop. The leftmost window shows the compilation of a C program 'client.c' into 'client.o' and 'server.c' into 'server'. The middle window shows the execution of 'server' with arguments '-lthread -lrt' and the connection of 'client1' and 'client2'. The rightmost window shows the execution of 'client' with arguments '-lthread -lrt' and the connection to 'server'. A red arrow points to the 'killServer' command in the middle window, with the text 'kill server' written in red above it.

When killServer command is received, all the clients and server is killed.

```
Client PID:42919 connected as "client1"
Client PID:42940 connected as "client2"
Client PID:42943 connected as "client3"
Client PID:42947 connected as "client4"
Client PID:42951 connected as "client5"
client1 requested:help
client2 requested:list
client4 requested:readF testfile.txt
client5 requested:readF bg.jpeg 2
client5 requested:quit
client5 disconnected

client4 requested:quit
client4 disconnected

Client PID:42998 connected as "client6"
Client PID:43002 connected as "client7"
client6 requested:writeT file.txt newly created file
client6 requested:quit
client6 disconnected

client1 requested:writeT fff.txt this is created now
client1 requested:download bg.jpeg bg3.jpeg
client1 requested:quit
client1 disconnected

client2 requested:killServer
```

Log file consists of client connection info and commands.