CSE 344 MIDTERM

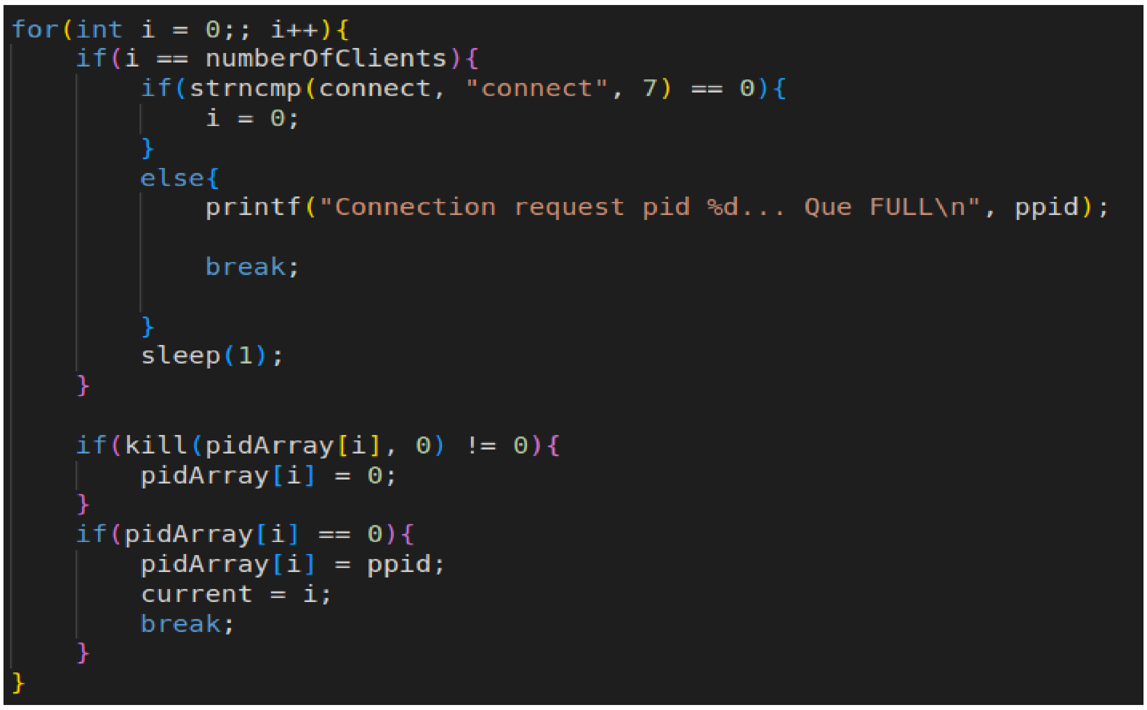
INTRODUCTION

The project is made for CSE 344 lecture. A lot of important concepts of the system programming such as piping, parent-child process, inter process communication, busy waiting, semaphores are used to satisfy midterm requirements. In the rest of the report, a detailed explanation will be given about the purposes and locations where these concepts are used.

CONCEPTS

BUSY WAITING

The client program with connect option request a spot from the server que and connects if a spot is available, otherwise waits until a spot is available. To satisfy this requirement busy waiting is used.

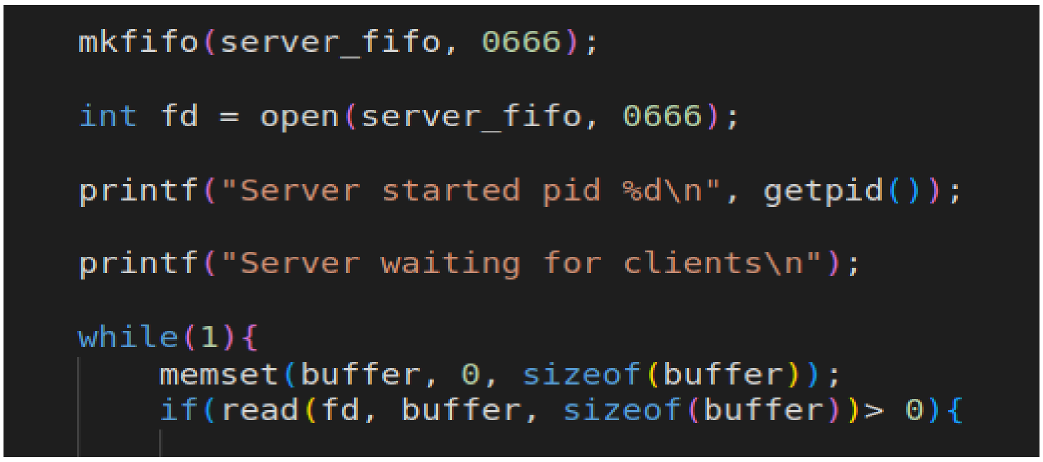


```c
for(int i = 0;; i++){
    if(i == numberOfClients){
        if(strncmp(connect, "connect", 7) == 0){
            i = 0;
        }
        else{
            printf("Connection request pid %d... Que FULL\n", ppid);

            break;

        }
        sleep(1);
    }

    if(kill(pidArray[i], 0) != 0){
        pidArray[i] = 0;
    }
    if(pidArray[i] == 0){
        pidArray[i] = ppid;
        current = i;
        break;
    }
}
```

Each client spot on the server are checked whether the spot is available or not. If all the spots are unavailable, argument of the new client is checked. If the new client is tried to connect to server with “connect” argument, the counter is reset, and check all clients until it finds an available spot. So, busy waiting is accomplished. On the other hand, if the client tried to connect to server with “tryConnect” argument, que full message is printed.

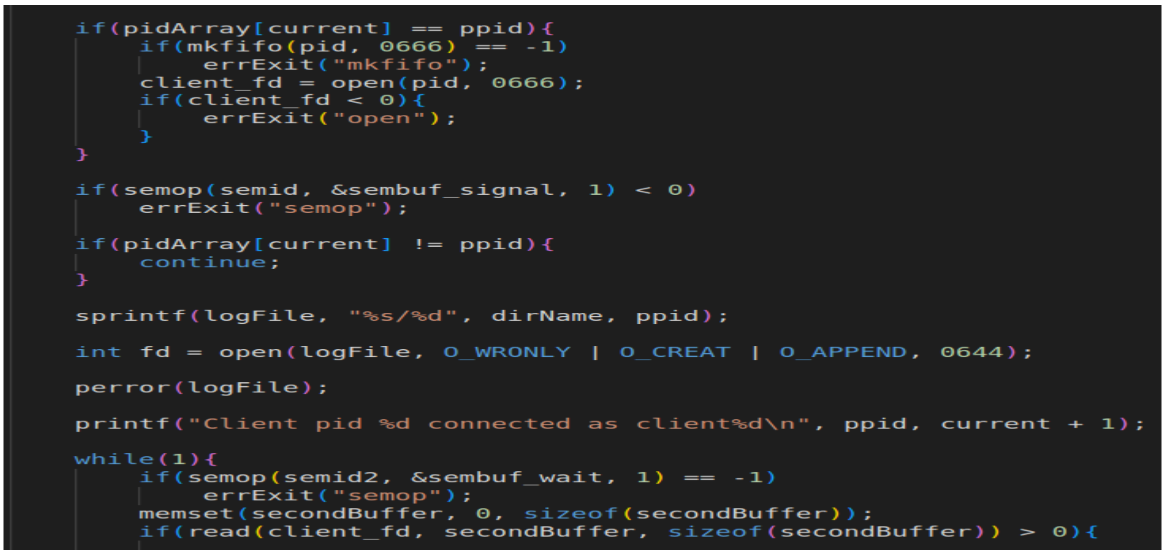
INTERPROCESS COMMUNICATION

One of the main parts of the homework is ensuring the communication between server process and client processes. To provide inter-process communication, there is two options: FIFO and shared memory. Both options are tested but FIFO is preferred because it is safer than shared memory as my experience.



```c
mkfifo(server_fifo, 0666);

int fd = open(server_fifo, 0666);

printf("Server started pid %d\n", getpid());

printf("Server waiting for clients\n");

while(1){
    memset(buffer, 0, sizeof(buffer));
    if(read(fd, buffer, sizeof(buffer))> 0){
```

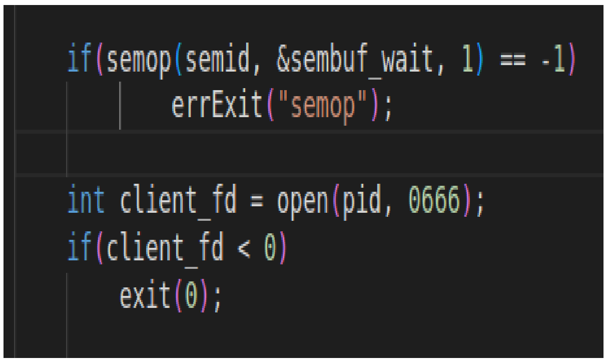
In the architecture of this project, there are two FIFOs, one specific to the server and one specific to the client. The first of these is the FIFO created with the PID of the server when the server process starts. Then the process server is blocked with the read function until something is written to the FIFO. At this time, the client process is started with the PID of the server. Using this PID, the client connects to the server FIFO and prints its own PID. So, the server process continues to run. As a result, server FIFO takes part in sending the PID of the clients to the server, while the client FIFO takes part in sending the requests to the server and sending the responses to the client.



```c
if(pidArray[current] == ppid){
    if(mkfifo(pid, 0666) == -1)
        errExit("mkfifo");
    client_fd = open(pid, 0666);
    if(client_fd < 0){
        errExit("open");
    }
}

if(semop(semid, &sembuf_signal, 1) < 0)
    errExit("semop");

if(pidArray[current] != ppid){
    continue;
}

sprintf(logFile, "%s/%d", dirName, ppid);

int fd = open(logFile, O_WRONLY | O_CREAT | O_APPEND, 0644);

perror(logFile);

printf("Client pid %d connected as client%d\n", ppid, current + 1);

while(1){
    if(semop(semid2, &sembuf_wait, 1) == -1)
        errExit("semop");
    memset(secondBuffer, 0, sizeof(secondBuffer));
    if(read(client_fd, secondBuffer, sizeof(secondBuffer)) > 0){
```

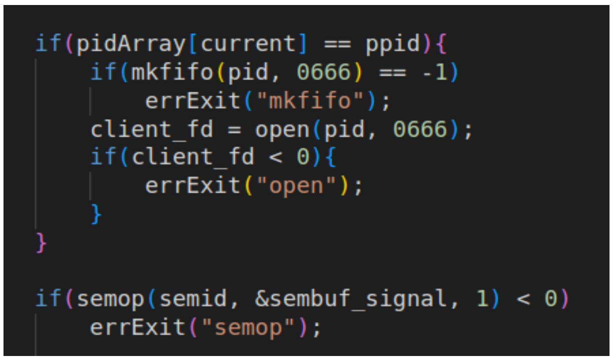
SEMAPHORES

In this project, to achieve having unique FIFO for each client, The FIFOs must be created in server side, and client must be connected to FIFO. However, it is probable that client trying to connect to client FIFO before it is created because of the synchronization issues. To avoid this, the client process must be blocked until the FIFOs are created.



```
if(semop(semid, &sembuf_wait, 1) == -1)
    errExit("semop");

int client_fd = open(pid, 0666);
if(client_fd < 0)
    exit(0);
```

(Block)

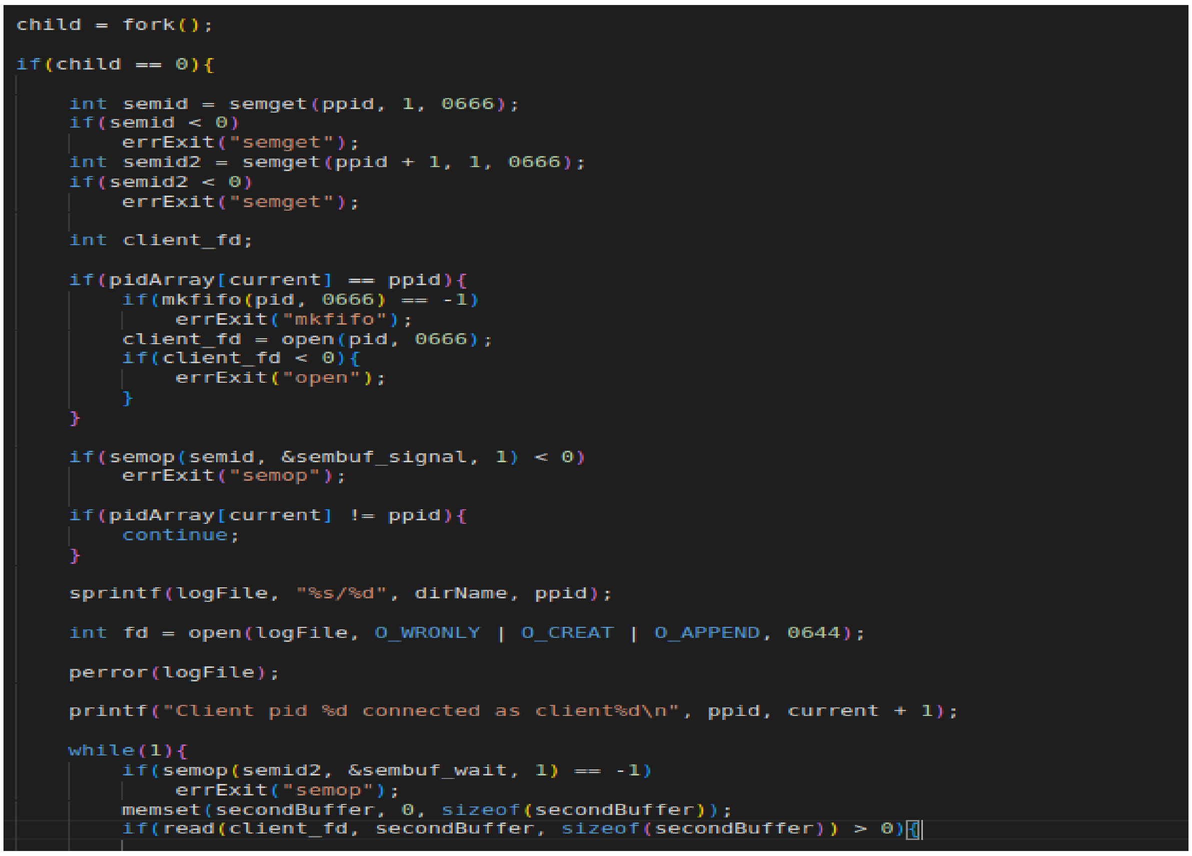


```
if(pidArray[current] == ppid){
    if(mkfifo(pid, 0666) == -1)
        errExit("mkfifo");
    client_fd = open(pid, 0666);
    if(client_fd < 0){
        errExit("open");
    }
}

if(semop(semid, &sembuf_signal, 1) < 0)
    errExit("semop");
```

(Release)

PARENT-CHILD PROCESSES

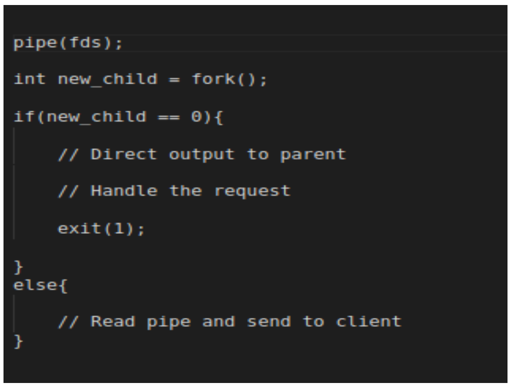
To meet the requirement that for each client connected will fork a copy of itself to serve the specified client, process is forked after the PID of the client is saved to an array. After it reads the requests which are sent from the client in a while loop.



```
child = fork();

if(child == 0){

    int semid = semget(ppid, 1, 0666);
    if(semid < 0)
        errExit("semget");
    int semid2 = semget(ppid + 1, 1, 0666);
    if(semid2 < 0)
        errExit("semget");

    int client_fd;

    if(pidArray[current] == ppid){
        if(mkfifo(pid, 0666) == -1)
            errExit("mkfifo");
        client_fd = open(pid, 0666);
        if(client_fd < 0){
            errExit("open");
        }
    }

    if(semop(semid, &sembuf_signal, 1) < 0)
        errExit("semop");

    if(pidArray[current] != ppid){
        continue;
    }

    sprintf(logFile, "%s/%d", dirName, ppid);

    int fd = open(logFile, O_WRONLY | O_CREAT | O_APPEND, 0644);

    perror(logFile);

    printf("Client pid %d connected as client%d\n", ppid, current + 1);

    while(1){
        if(semop(semid2, &sembuf_wait, 1) == -1)
            errExit("semop");
        memset(secondBuffer, 0, sizeof(secondBuffer));
        if(read(client_fd, secondBuffer, sizeof(secondBuffer)) > 0){
```

PIPING

To handle each request which is sent from the client, server creates a child. Before the forking, piping operation is done. In this way, child takes necessary action according to request and direct its output to input of the parent. After that parent sends the response to the client.

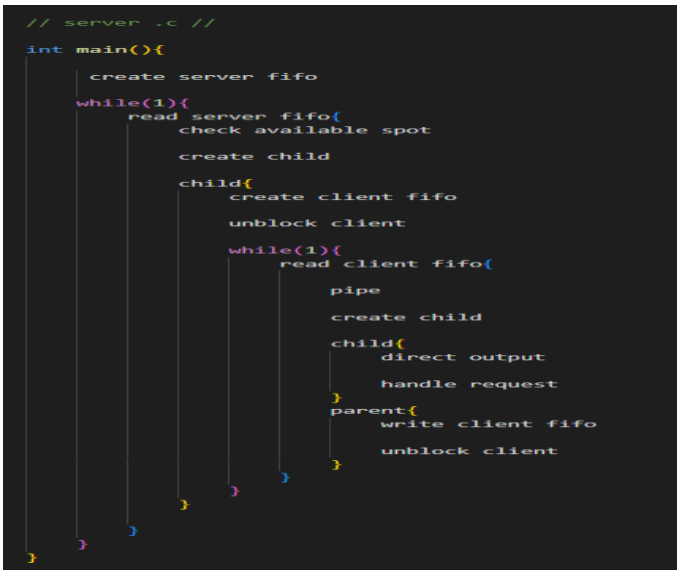


```
pipe(fds);
int new_child = fork();
if(new_child == 0){
    // Direct output to parent
    // Handle the request
    exit(1);
}
else{
    // Read pipe and send to client
}
```

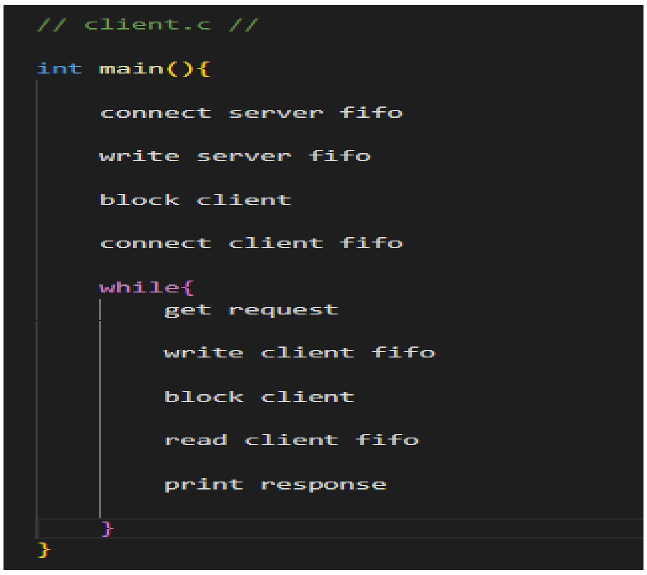
(Template piping)

WORKING PRINCIPLE

The working principle of this project is as follows: The server starts running and creates a FIFO (First-In-First-Out) queue for reading. The server process is blocked until something is written to this FIFO. When a new client works with the server PID, it sends its PID to the FIFO and the client process is blocked using a semaphore. The client waits in suspension until the server has an available space for the client. Then the server creates a child process for the client and a specific FIFO for the client. The client starts receiving requests. The received requests are transferred from the client to the server using the client's specific FIFO. For each incoming request, the server creates a child process, handles the request, and sends the resulting output to the parent process. The parent process then sends this output to the client. This way, communication is established between the client and the server.



```
// server .c //
int main(){
        create server fifo
    while(1){
        read server fifo{
            check available spot
            create child
            child{
                create client fifo
                unblock client
                while(1){
                    read client fifo{
                        pipe
                        create child
                        child{
                            direct output
                            handle request
                        }
                        parent{
                            write client fifo
                            unblock client
                        }
                    }
                }
            }
        }
    }
}
```

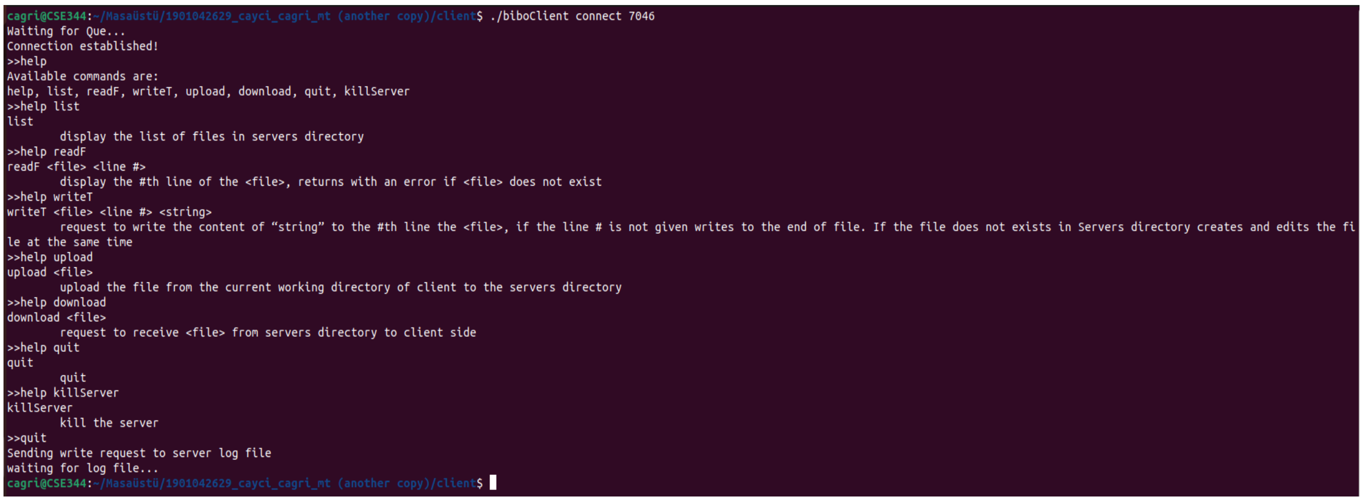


```
// client.c //
int main(){
    connect server fifo
    write server fifo
    block client
    connect client fifo
    while{
        get request
        write client fifo
        block client
        read client fifo
        print response
    }
}
```

(Template for server and client)

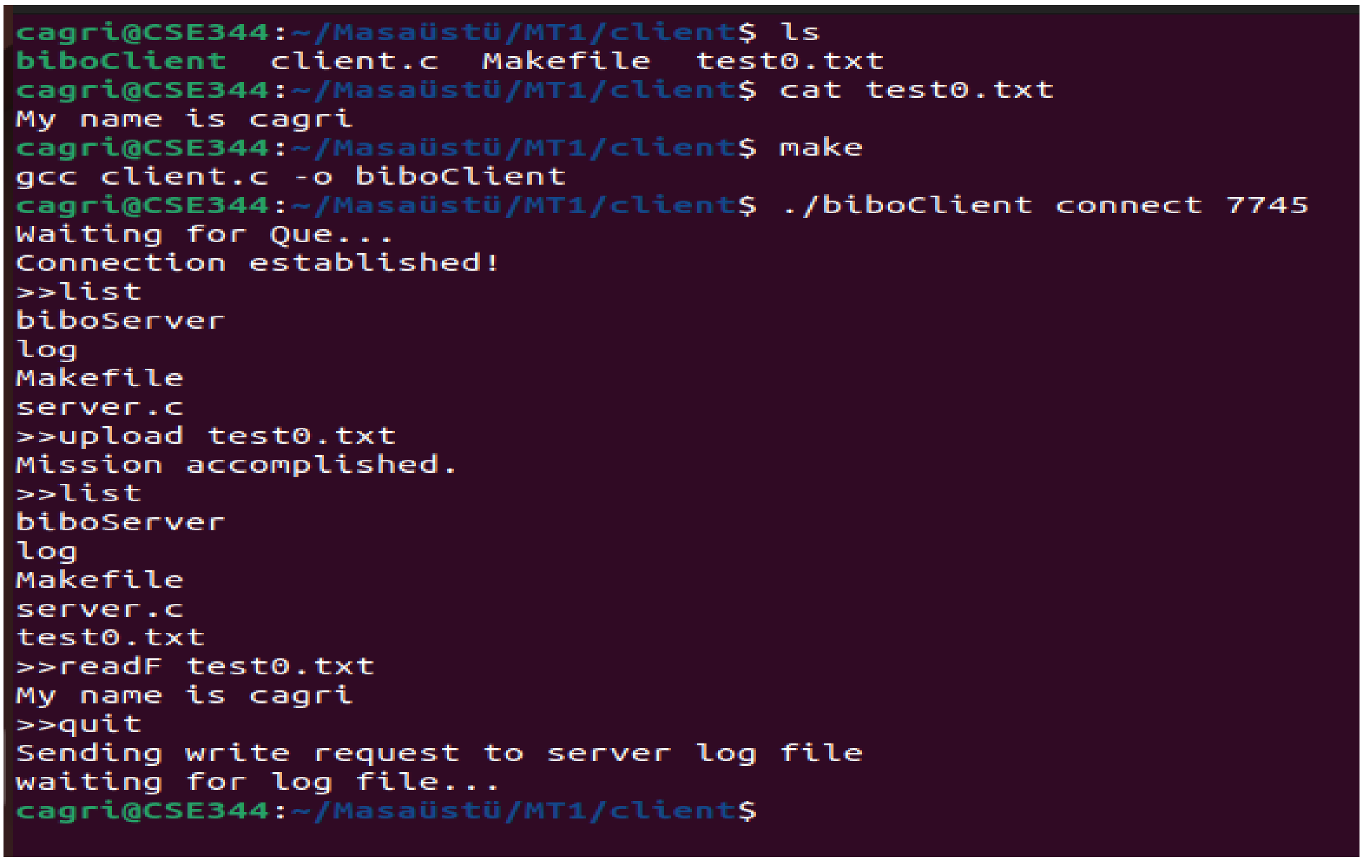
TESTS

HELP



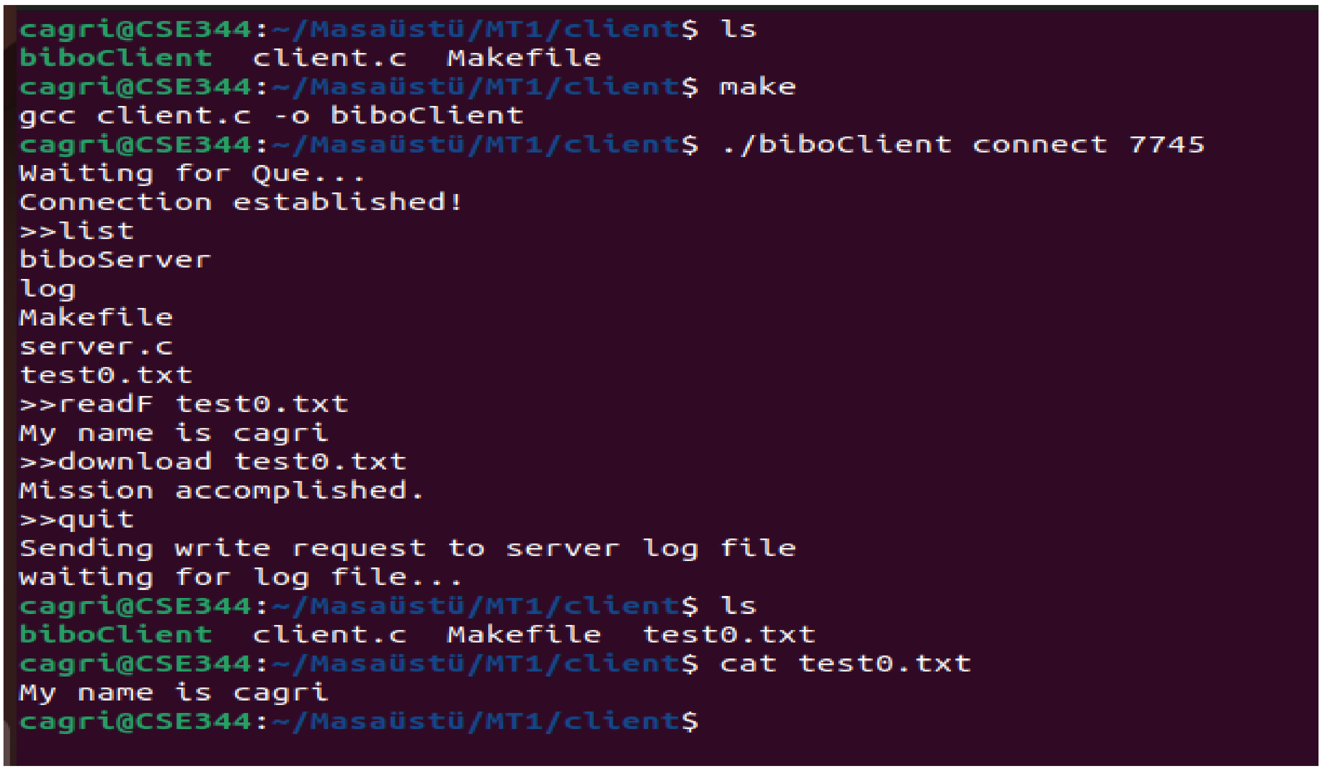
```
cagri@CSE344:~/Masaüstü/1901042629_cayci_cagri_mt (another copy)/client$ ./biboClient connect 7046
Waiting for Que...
Connection established!
>>help
Available commands are:
help, list, readF, writeT, upload, download, quit, killServer
>>help list
list
        display the list of files in servers directory
>>help readF
readF <file> <line #>
        display the #th line of the <file>, returns with an error if <file> does not exist
>>help writeT
writeT <file> <line #> <string>
        request to write the content of "string" to the #th line the <file>, if the line # is not given writes to the end of file. If the file does not exists in Servers directory creates and edits the fi
le at the same time
>>help upload
upload <file>
        upload the file from the current working directory of client to the servers directory
>>help download
download <file>
        request to receive <file> from servers directory to client side
>>help quit
quit
        quit
>>help killServer
killServer
        kill the server
>>quit
Sending write request to server log file
waiting for log file...
cagri@CSE344:~/Masaüstü/1901042629_cayci_cagri_mt (another copy)/client$ 
```

UPLOAD

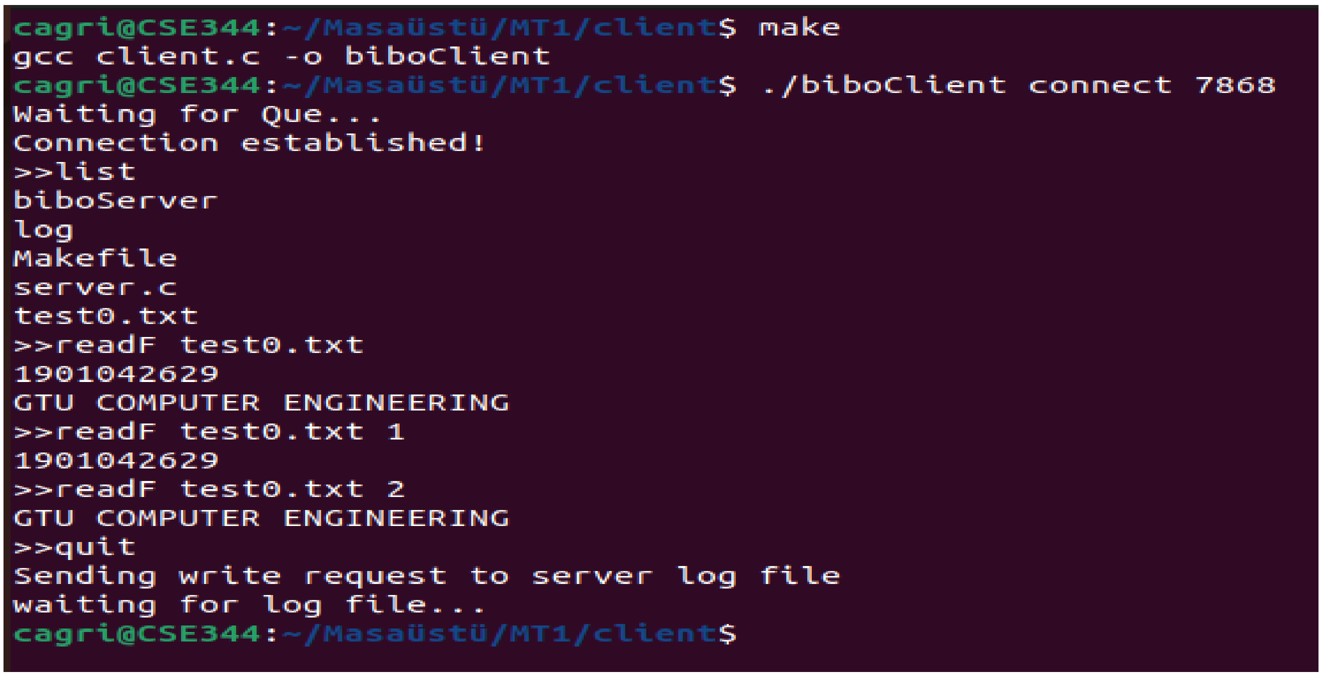


```
cagri@CSE344:~/Masaüstü/MT1/client$ ls
biboClient  client.c  Makefile  test0.txt
cagri@CSE344:~/Masaüstü/MT1/client$ cat test0.txt
My name is cagri
cagri@CSE344:~/Masaüstü/MT1/client$ make
gcc client.c -o biboClient
cagri@CSE344:~/Masaüstü/MT1/client$ ./biboClient connect 7745
Waiting for Que...
Connection established!
>>list
biboServer
log
Makefile
server.c
>>upload test0.txt
Mission accomplished.
>>list
biboServer
log
Makefile
server.c
test0.txt
>>readF test0.txt
My name is cagri
>>quit
Sending write request to server log file
waiting for log file...
cagri@CSE344:~/Masaüstü/MT1/client$
```

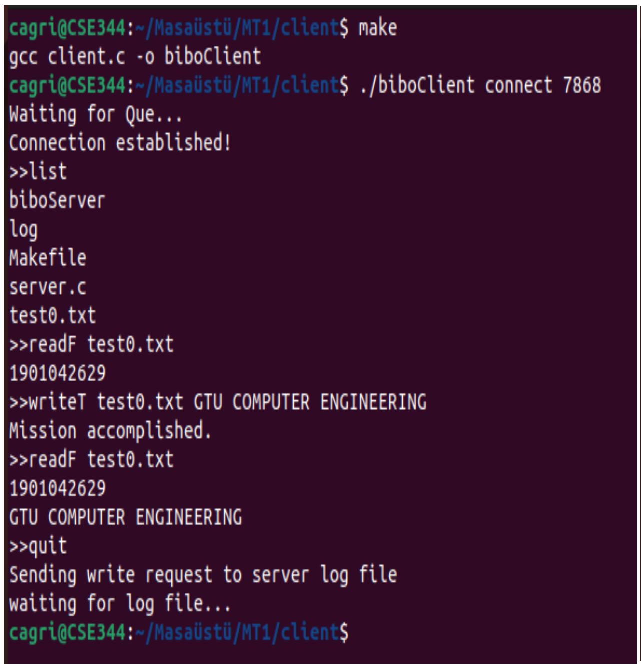
DOWNLOAD


```
cagri@CSE344:~/Masaüstü/MT1/client$ ls
biboClient  client.c  Makefile
cagri@CSE344:~/Masaüstü/MT1/client$ make
gcc client.c -o biboClient
cagri@CSE344:~/Masaüstü/MT1/client$ ./biboClient connect 7745
Waiting for Que...
Connection established!
>>list
biboServer
log
Makefile
server.c
test0.txt
>>readF test0.txt
My name is cagri
>>download test0.txt
Mission accomplished.
>>quit
Sending write request to server log file
waiting for log file...
cagri@CSE344:~/Masaüstü/MT1/client$ ls
biboClient  client.c  Makefile  test0.txt
cagri@CSE344:~/Masaüstü/MT1/client$ cat test0.txt
My name is cagri
cagri@CSE344:~/Masaüstü/MT1/client$
```

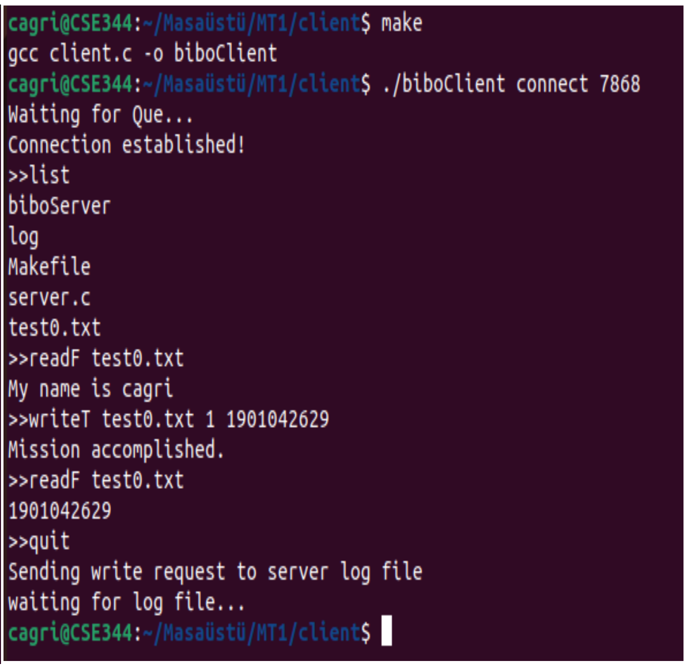
READF


```
cagri@CSE344:~/Masaüstü/MT1/client$ make
gcc client.c -o biboClient
cagri@CSE344:~/Masaüstü/MT1/client$ ./biboClient connect 7868
Waiting for Que...
Connection established!
>>list
biboServer
log
Makefile
server.c
test0.txt
>>readF test0.txt
1901042629
GTU COMPUTER ENGINEERING
>>readF test0.txt 1
1901042629
>>readF test0.txt 2
GTU COMPUTER ENGINEERING
>>quit
Sending write request to server log file
waiting for log file...
cagri@CSE344:~/Masaüstü/MT1/client$
```

WRITET

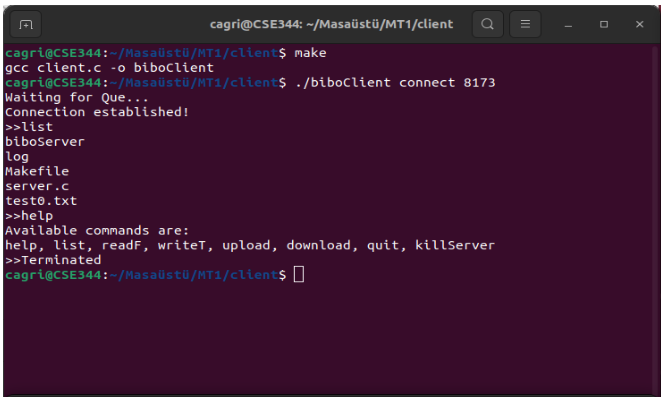


```
cagri@CSE344:~/Masaüstü/MT1/client$ make
gcc client.c -o biboClient
cagri@CSE344:~/Masaüstü/MT1/client$ ./biboClient connect 7868
Waiting for Que...
Connection established!
>>list
biboServer
log
Makefile
server.c
test0.txt
>>readF test0.txt
1901042629
>>writeT test0.txt GTU COMPUTER ENGINEERING
Mission accomplished.
>>readF test0.txt
1901042629
GTU COMPUTER ENGINEERING
>>quit
Sending write request to server log file
waiting for log file...
cagri@CSE344:~/Masaüstü/MT1/client$
```

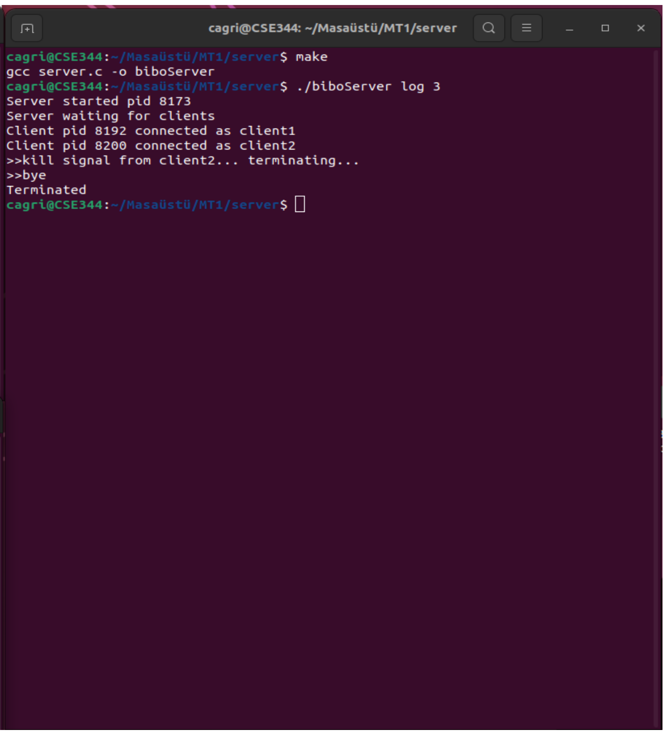


```
cagri@CSE344:~/Masaüstü/MT1/client$ make
gcc client.c -o biboClient
cagri@CSE344:~/Masaüstü/MT1/client$ ./biboClient connect 7868
Waiting for Que...
Connection established!
>>list
biboServer
log
Makefile
server.c
test0.txt
>>readF test0.txt
My name is cagri
>>writeT test0.txt 1 1901042629
Mission accomplished.
>>readF test0.txt
1901042629
>>quit
Sending write request to server log file
waiting for log file...
cagri@CSE344:~/Masaüstü/MT1/client$
```

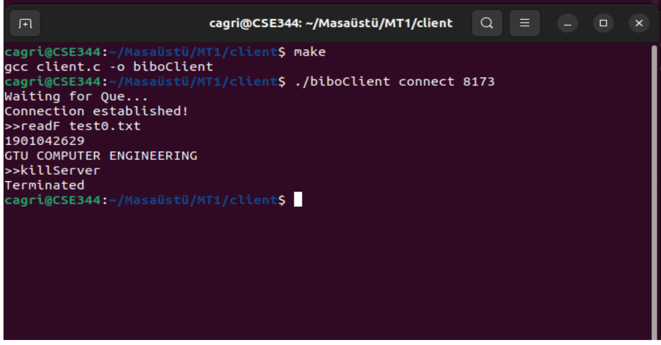
KILLSERVER



```
cagri@CSE344:~/Masaüstü/MT1/client$ make
gcc client.c -o biboClient
cagri@CSE344:~/Masaüstü/MT1/client$ ./biboClient connect 8173
Waiting for Que...
Connection established!
>>list
biboServer
log
Makefile
server.c
test0.txt
>>help
Available commands are:
help, list, readF, writeT, upload, download, quit, killServer
>>Terminated
cagri@CSE344:~/Masaüstü/MT1/client$
```

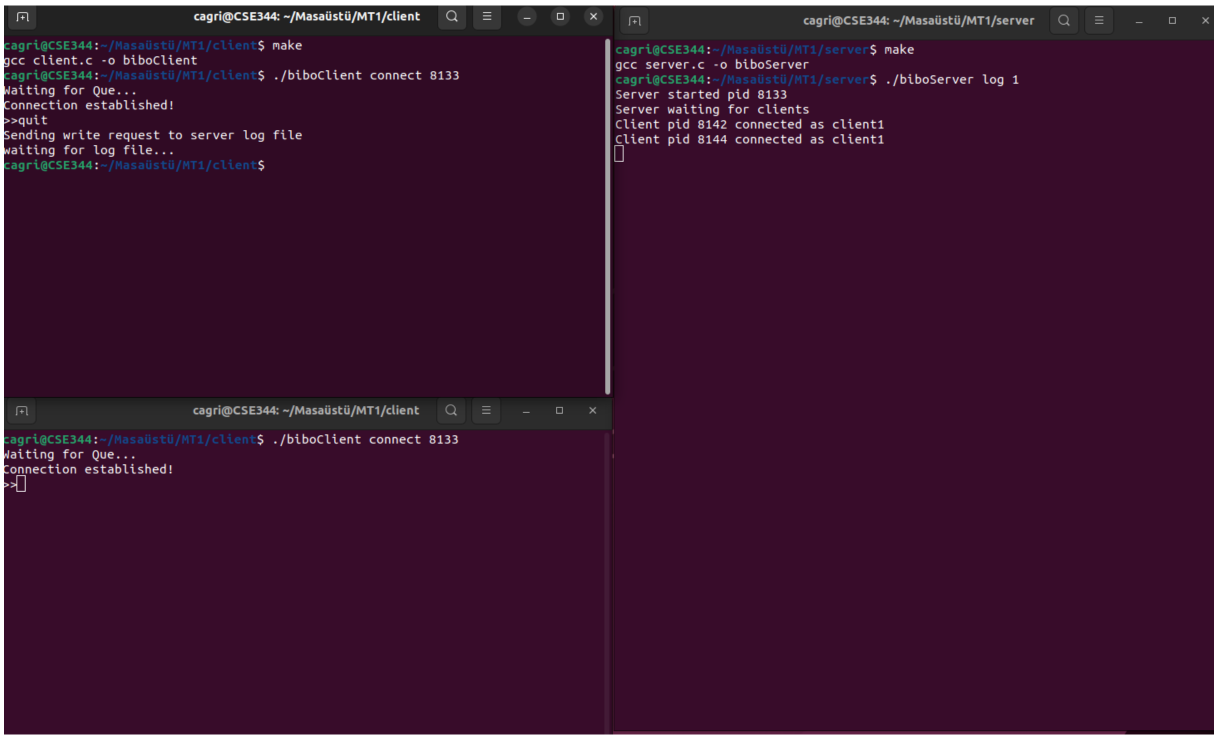


```
cagri@CSE344:~/Masaüstü/MT1/server$ make
gcc server.c -o biboServer
cagri@CSE344:~/Masaüstü/MT1/server$ ./biboServer log 3
Server started pid 8173
Server waiting for clients
Client pid 8192 connected as client1
Client pid 8200 connected as client2
>>kill signal from client2... terminating...
>>bye
Terminated
cagri@CSE344:~/Masaüstü/MT1/server$
```



```
cagri@CSE344:~/Masaüstü/MT1/client$ make
gcc client.c -o biboClient
cagri@CSE344:~/Masaüstü/MT1/client$ ./biboClient connect 8173
Waiting for Que...
Connection established!
>>readF test0.txt
1901042629
GTU COMPUTER ENGINEERING
>>killServer
Terminated
cagri@CSE344:~/Masaüstü/MT1/client$
```

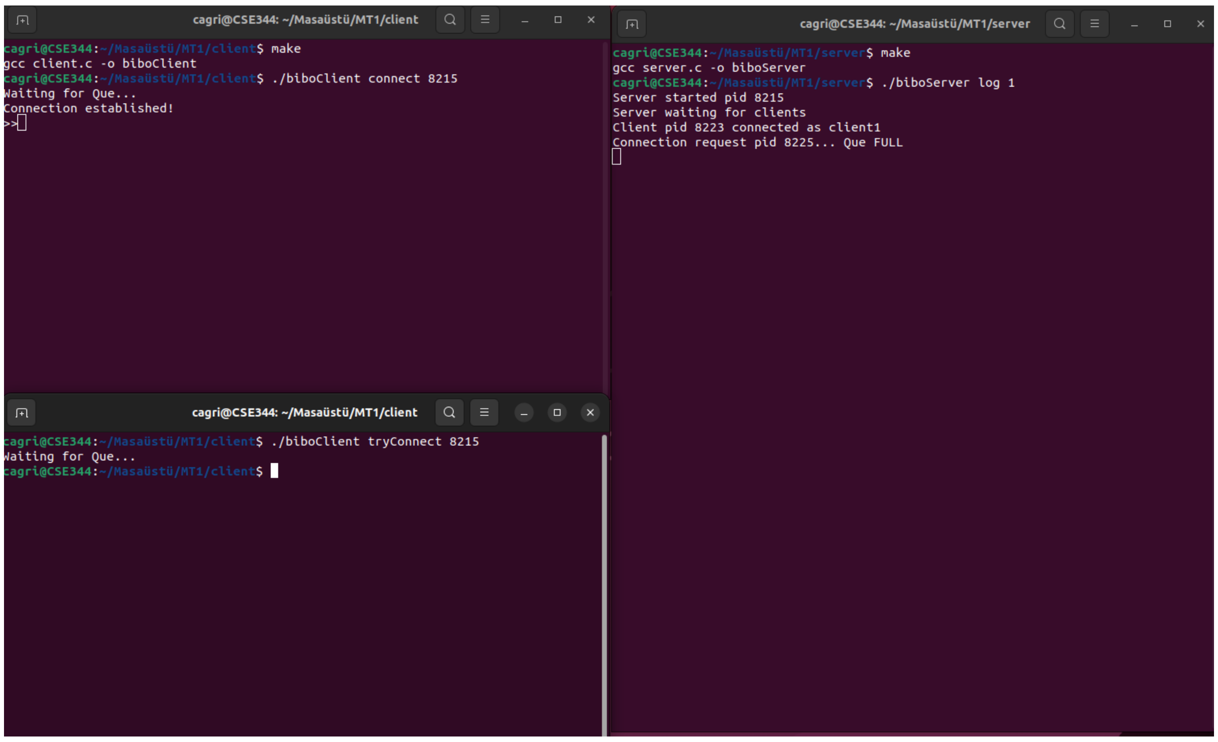
BUSY WAITING


cagri@CSE344: ~/Masaüstü/MT1/client
cagri@CSE344: ~/Masaüstü/MT1/server
cagri@CSE344: ~/Masaüstü/MT1/client

```
cagri@CSE344:~/Masaüstü/MT1/client$ make
gcc client.c -o biboClient
cagri@CSE344:~/Masaüstü/MT1/client$ ./biboClient connect 8133
Waiting for Que...
Connection established!
>>quit
Sending write request to server log file
waiting for log file...
cagri@CSE344:~/Masaüstü/MT1/client$
```

```
cagri@CSE344:~/Masaüstü/MT1/client$ ./biboClient connect 8133
Waiting for Que...
Connection established!
>>
```

```
cagri@CSE344:~/Masaüstü/MT1/server$ make
gcc server.c -o biboServer
cagri@CSE344:~/Masaüstü/MT1/server$ ./biboServer log 1
Server started pid 8133
Server waiting for clients
Client pid 8142 connected as client1
Client pid 8144 connected as client1
```


cagri@CSE344: ~/Masaüstü/MT1/client
cagri@CSE344: ~/Masaüstü/MT1/server
cagri@CSE344: ~/Masaüstü/MT1/client

```
cagri@CSE344:~/Masaüstü/MT1/client$ make
gcc client.c -o biboClient
cagri@CSE344:~/Masaüstü/MT1/client$ ./biboClient connect 8215
Waiting for Que...
Connection established!
>>
```

```
cagri@CSE344:~/Masaüstü/MT1/client$ ./biboClient tryConnect 8215
Waiting for Que...
cagri@CSE344:~/Masaüstü/MT1/client$
```

```
cagri@CSE344:~/Masaüstü/MT1/server$ make
gcc server.c -o biboServer
cagri@CSE344:~/Masaüstü/MT1/server$ ./biboServer log 1
Server started pid 8215
Server waiting for clients
Client pid 8223 connected as client1
Connection request pid 8225... Que FULL
```