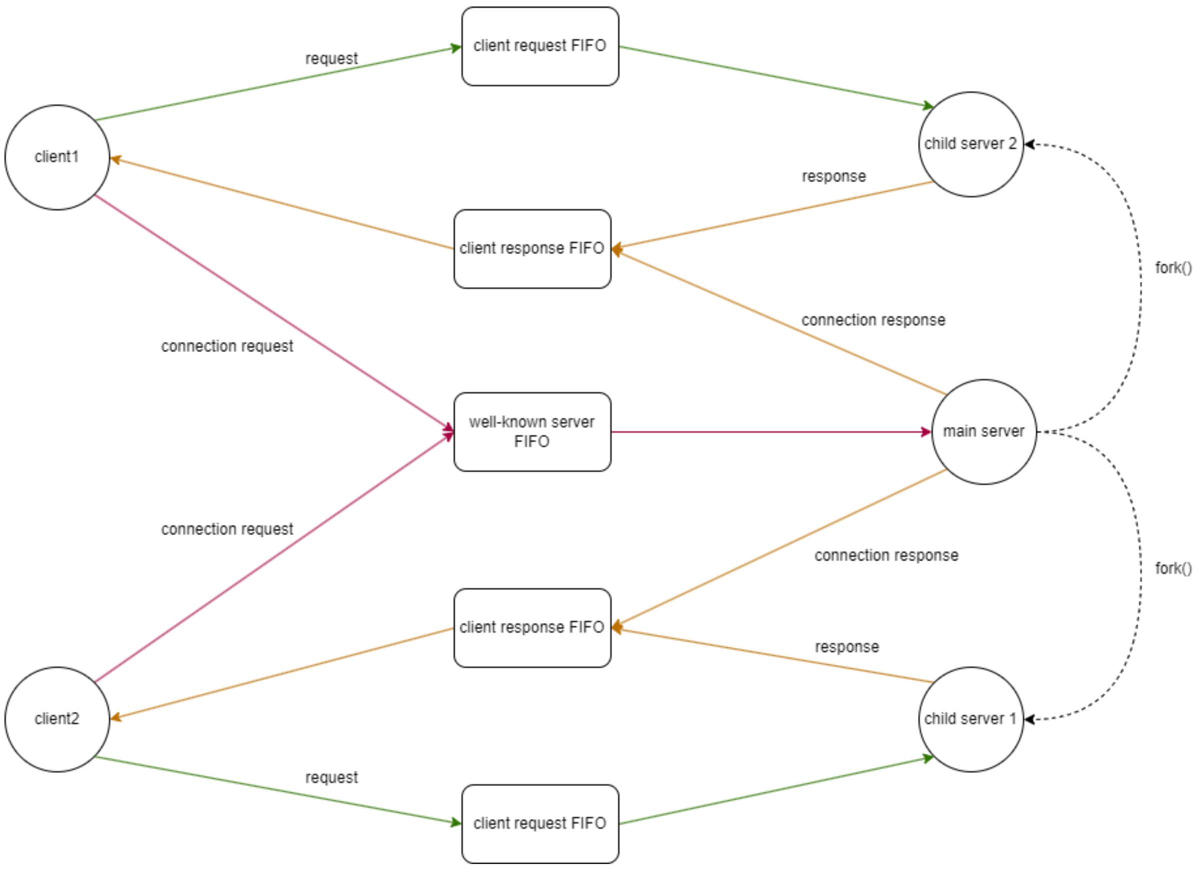
GTU Department of Computer Engineering
CSE 344 - Spring 2023
Midterm Project Report

Emirkan Burak Yılmaz
1901042659

MybiboBox
A Concurrent File Access System

Contents

1 System Architecture



1.1 Connecting to the MyBiboBox Server

The communication between client and server is handled via unnamed pipes (FIFOs). There is one well known server FIFO which is used to connect to the MyBiboBox system. Client first prepares a packet which contains its PID, current working directory and the wait bit. The wait bit indicates whether the client prefers waiting in the wait queue or not. Client makes a connection request with the prepared packet to the main server via well-known server FIFO and the main server identifies the client request, check the wait queue and the status of child servers. According to the status of wait queue and number of children, main server prepares a response to indicate if it is possible to serve the client immediately or wait it a little bit. According to coming response from the main server, the client waits or gives up according to it connect option (connect/tryConnect) which is passed as command line argument to the client program. If the client is open with connect option, it opens the client request FIFO and blocked until main server forks a child server and that child opens the other and of client request FIFO. If the client chose to give up, then it directly terminates without connecting to the server.

1.2 Requests and Responses

The PIDs of child servers are kept in array and it is controlled by the main server. Whenever a new child is forked it adds the child PID to the array When the number of child server is less than the # of max child and there is a waiting client, a new child server is forked from the main server and the necessary informations about the child is taken. The child server opens the write end of the client FIFO which is unique for the client and the client unblocked and ready for listening user commands. Each user command converted to a request. Size of each request may change depend on the user command and arguments. For that reason, the size of the request data is sent with a header before sending the actual data. With this way, child server can read the request in prepare a response. A response is sent with the header and the actual data with the same way as client does.

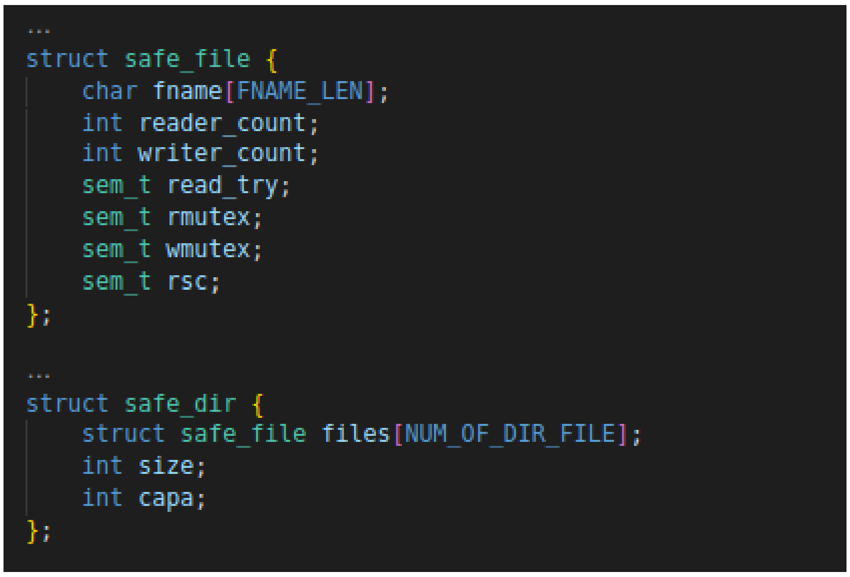
1.3 Disconnection

Client disconnects from the server either sending quit command or pressing CTRL-C and the signal handler takes cares the rest.

2 Synchronization

2.1 Files in Server Directory

For enforcing mutual exclusion to prevent race conditions in the server directory, the readers-writers problem is followed. To integrate readers writers’ approach, safe_file and safe_dir structures are created.



```c
...
struct safe_file {
    char fname[FNAME_LEN];
    int reader_count;
    int writer_count;
    sem_t read_try;
    sem_t rmutex;
    sem_t wmutex;
    sem_t rsc;
};

...
struct safe_dir {
    struct safe_file files[NUM_OF_DIR_FILE];
    int size;
    int capa;
};
```

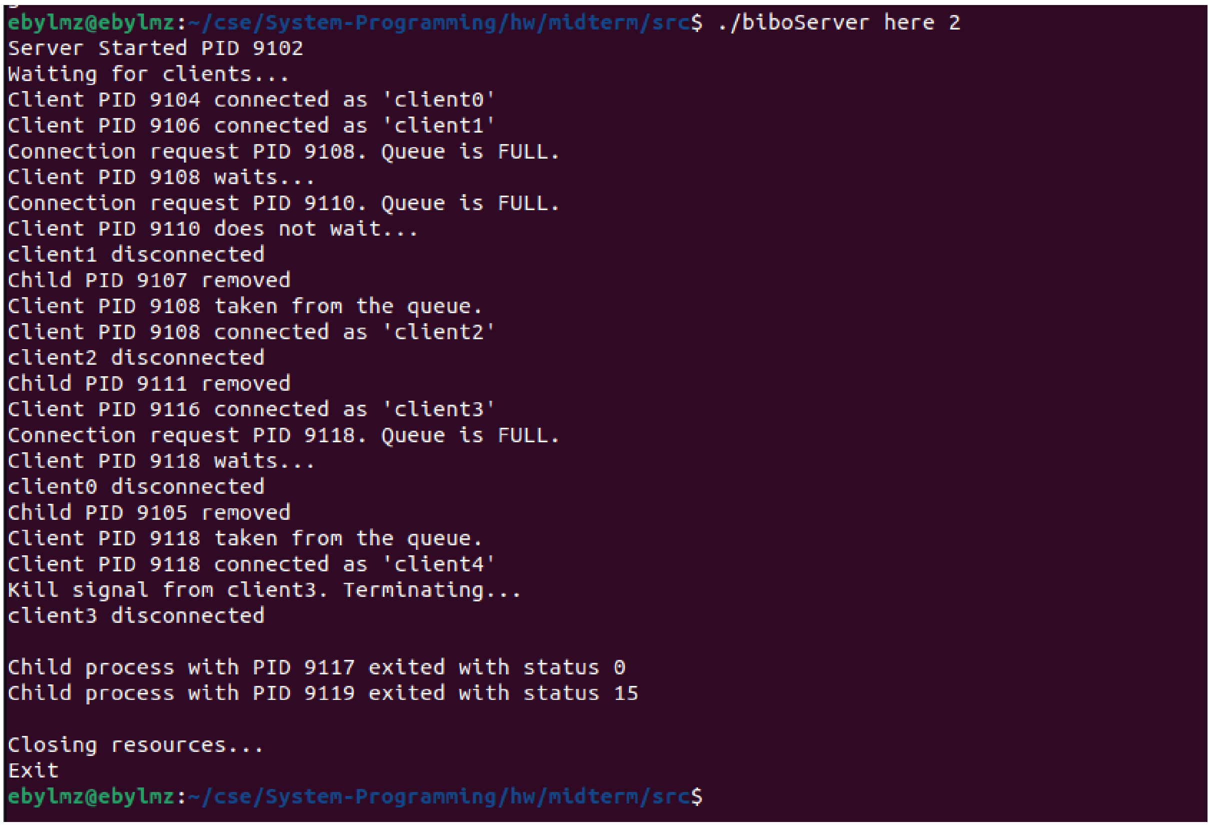
To share the semaphores between child processes, the main server creates a shared memory region and initialize the safe_dir structure in there. With this way each child process accesses the specific semaphores for a unique file by accessing the safe_dir structure inside this shared memory region. safe_file structure contains the name of the file, reader and writer counts and the semaphores for providing mutual exclusion and preventing race conditions.

2.2 Log File

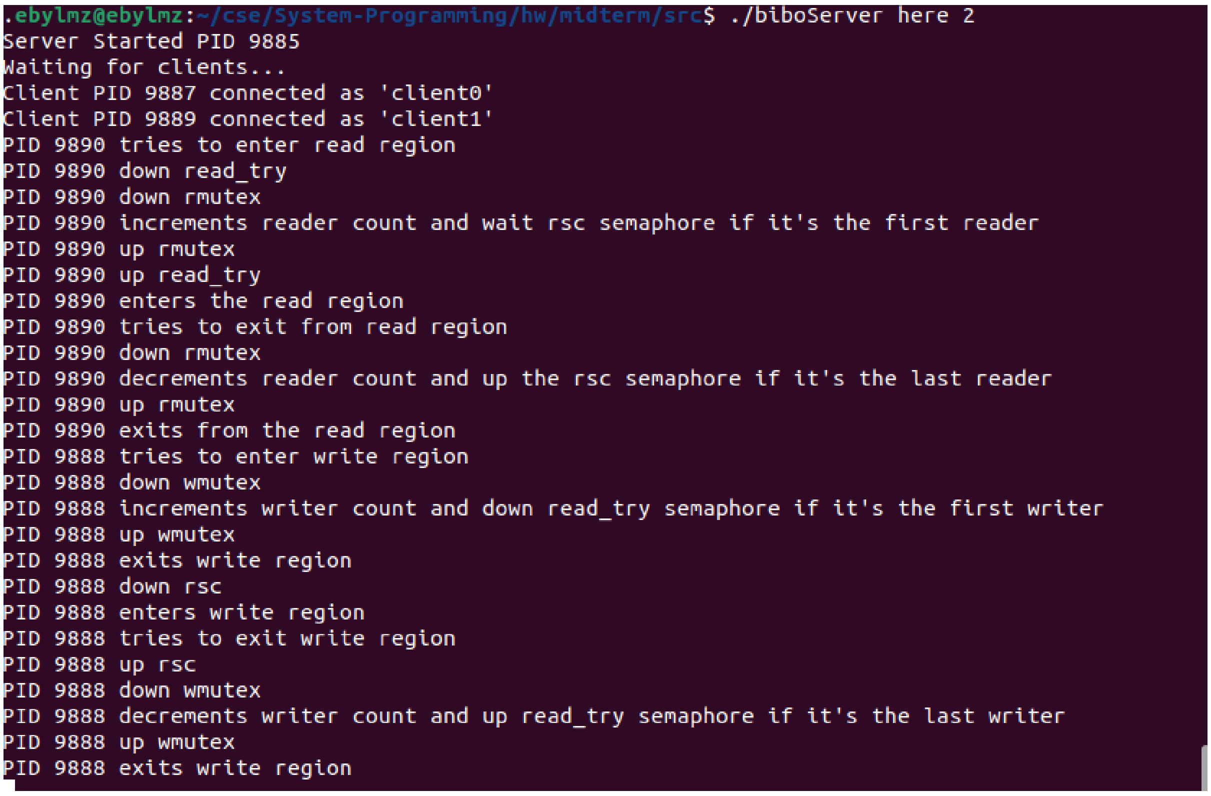
To provide mutual exclusion for writing the log file single binary semaphore is used.

3 Test Cases and Results

3.1 Client Connection Options (connect/tryConnect)



```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboServer here 2
Server Started PID 9102
Waiting for clients...
Client PID 9104 connected as 'client0'
Client PID 9106 connected as 'client1'
Connection request PID 9108. Queue is FULL.
Client PID 9108 waits...
Connection request PID 9110. Queue is FULL.
Client PID 9110 does not wait...
client1 disconnected
Child PID 9107 removed
Client PID 9108 taken from the queue.
Client PID 9108 connected as 'client2'
client2 disconnected
Child PID 9111 removed
Client PID 9116 connected as 'client3'
Connection request PID 9118. Queue is FULL.
Client PID 9118 waits...
client0 disconnected
Child PID 9105 removed
Client PID 9118 taken from the queue.
Client PID 9118 connected as 'client4'
Kill signal from client3. Terminating...
client3 disconnected

Child process with PID 9117 exited with status 0
Child process with PID 9119 exited with status 15

Closing resources...
Exit
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$
```

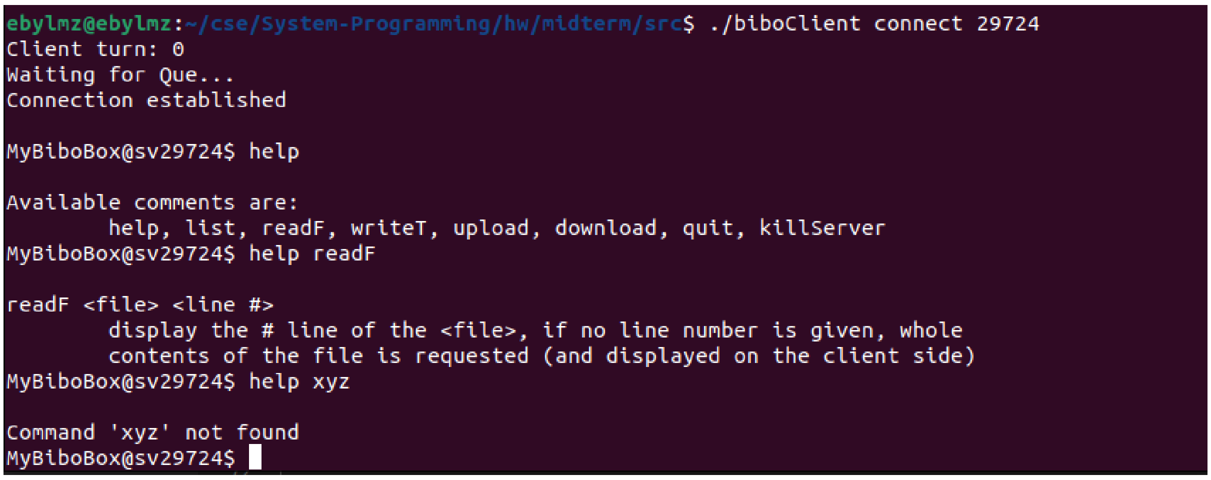


```
.ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboServer here 2
Server Started PID 9885
Waiting for clients...
Client PID 9887 connected as 'client0'
Client PID 9889 connected as 'client1'
PID 9890 tries to enter read region
PID 9890 down read_try
PID 9890 down rmutex
PID 9890 increments reader count and wait rsc semaphore if it's the first reader
PID 9890 up rmutex
PID 9890 up read_try
PID 9890 enters the read region
PID 9890 tries to exit from read region
PID 9890 down rmutex
PID 9890 decrements reader count and up the rsc semaphore if it's the last reader
PID 9890 up rmutex
PID 9890 exits from the read region
PID 9888 tries to enter write region
PID 9888 down wmutex
PID 9888 increments writer count and down read_try semaphore if it's the first writer
PID 9888 up wmutex
PID 9888 exits write region
PID 9888 down rsc
PID 9888 enters write region
PID 9888 tries to exit write region
PID 9888 up rsc
PID 9888 down wmutex
PID 9888 decrements writer count and up read_try semaphore if it's the last writer
PID 9888 up wmutex
PID 9888 exits write region
```

3.2 Basic User Commands

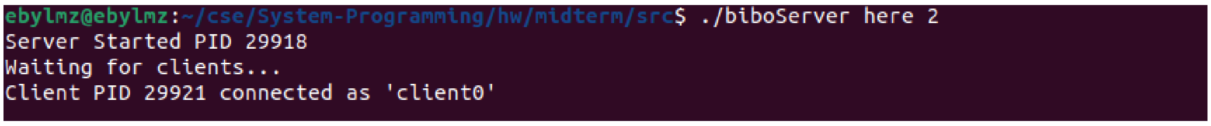
The user commands help, list, readF, writeT, upload, download, quit and killServer should satisfy the user needed.

3.2.1.1 help

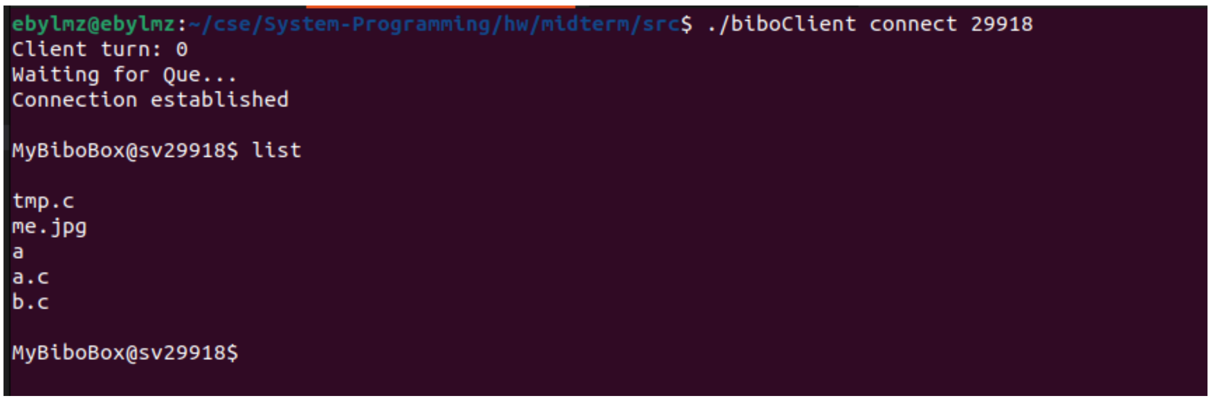


```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboClient connect 29724
Client turn: 0
Waiting for Que...
Connection established

MyBiboBox@sv29724$ help

Available comments are:
        help, list, readF, writeT, upload, download, quit, killServer
MyBiboBox@sv29724$ help readF

readF <file> <line #>
        display the # line of the <file>, if no line number is given, whole
        contents of the file is requested (and displayed on the client side)
MyBiboBox@sv29724$ help xyz

Command 'xyz' not found
MyBiboBox@sv29724$
```

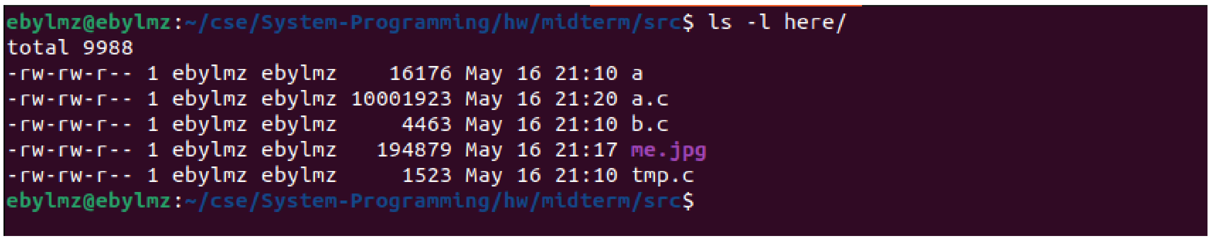
3.2.1.2 list



```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboServer here 2
Server Started PID 29918
Waiting for clients...
Client PID 29921 connected as 'client0'
```

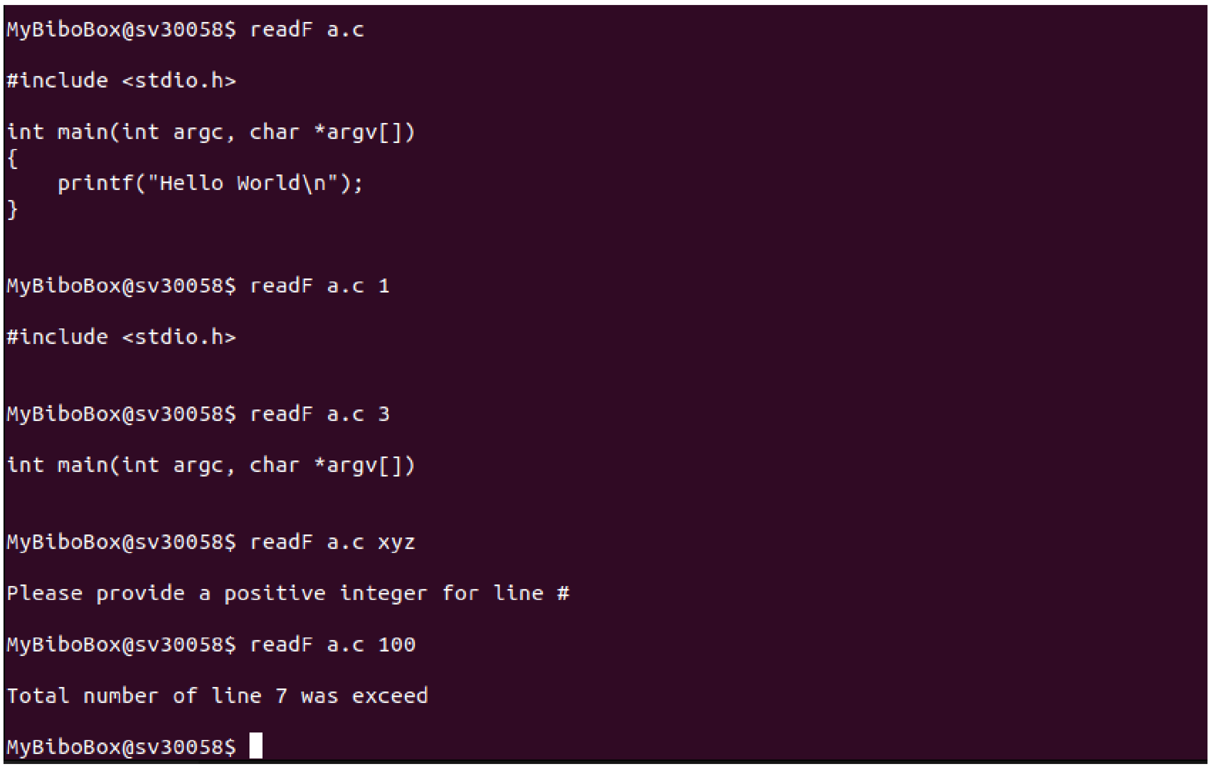


```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboClient connect 29918
Client turn: 0
Waiting for Que...
Connection established

MyBiboBox@sv29918$ list

tmp.c
me.jpg
a
a.c
b.c

MyBiboBox@sv29918$
```

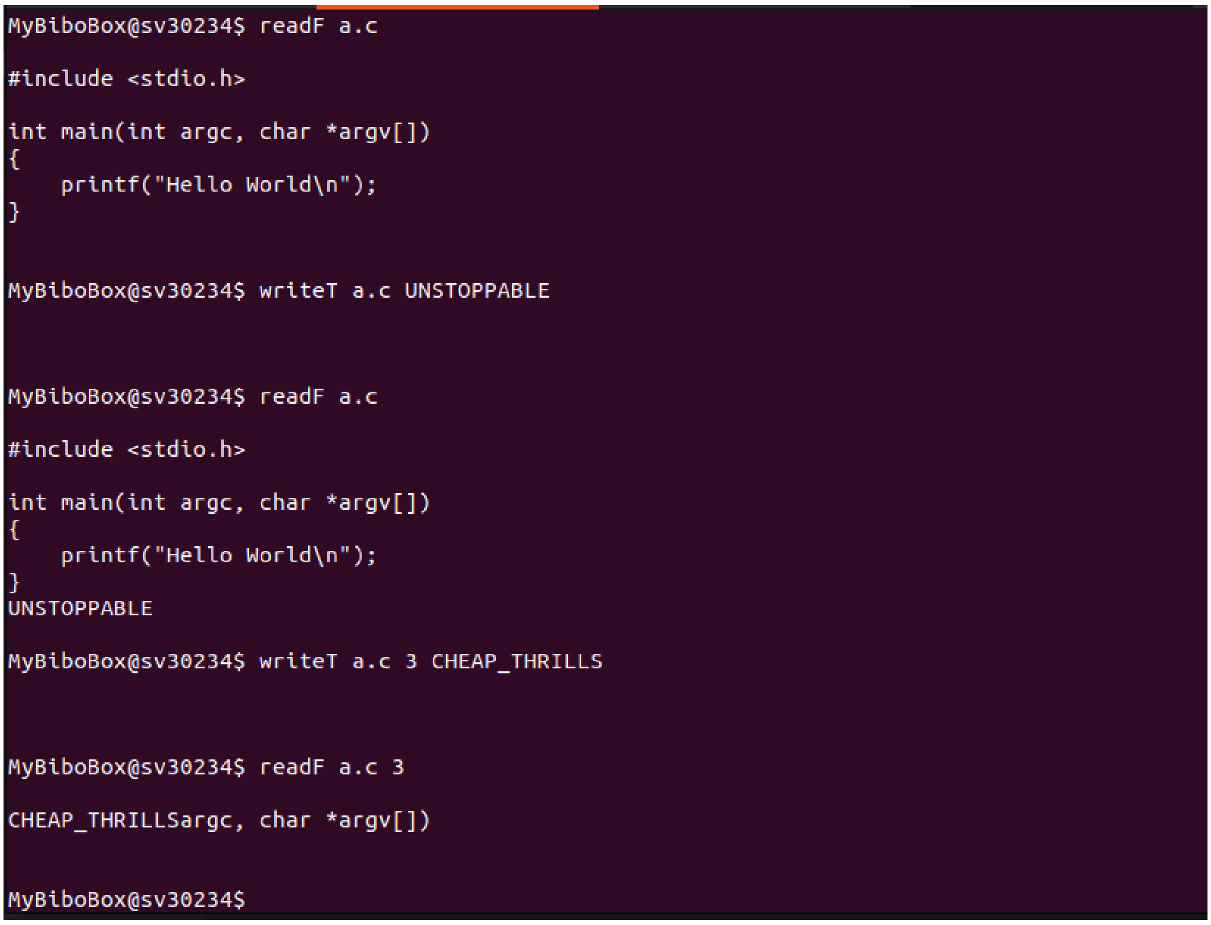


```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ls -l here/
total 9988
-rw-rw-r-- 1 ebylmz ebylmz    16176 May 16 21:10 a
-rw-rw-r-- 1 ebylmz ebylmz 10001923 May 16 21:20 a.c
-rw-rw-r-- 1 ebylmz ebylmz     4463 May 16 21:10 b.c
-rw-rw-r-- 1 ebylmz ebylmz   194879 May 16 21:17 me.jpg
-rw-rw-r-- 1 ebylmz ebylmz     1523 May 16 21:10 tmp.c
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$
```

3.2.1.3 readF


```
MyBiboBox@sv30058$ readF a.c

#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("Hello World\n");
}


MyBiboBox@sv30058$ readF a.c 1

#include <stdio.h>


MyBiboBox@sv30058$ readF a.c 3

int main(int argc, char *argv[])


MyBiboBox@sv30058$ readF a.c xyz

Please provide a positive integer for line #

MyBiboBox@sv30058$ readF a.c 100

Total number of line 7 was exceed

MyBiboBox@sv30058$
```

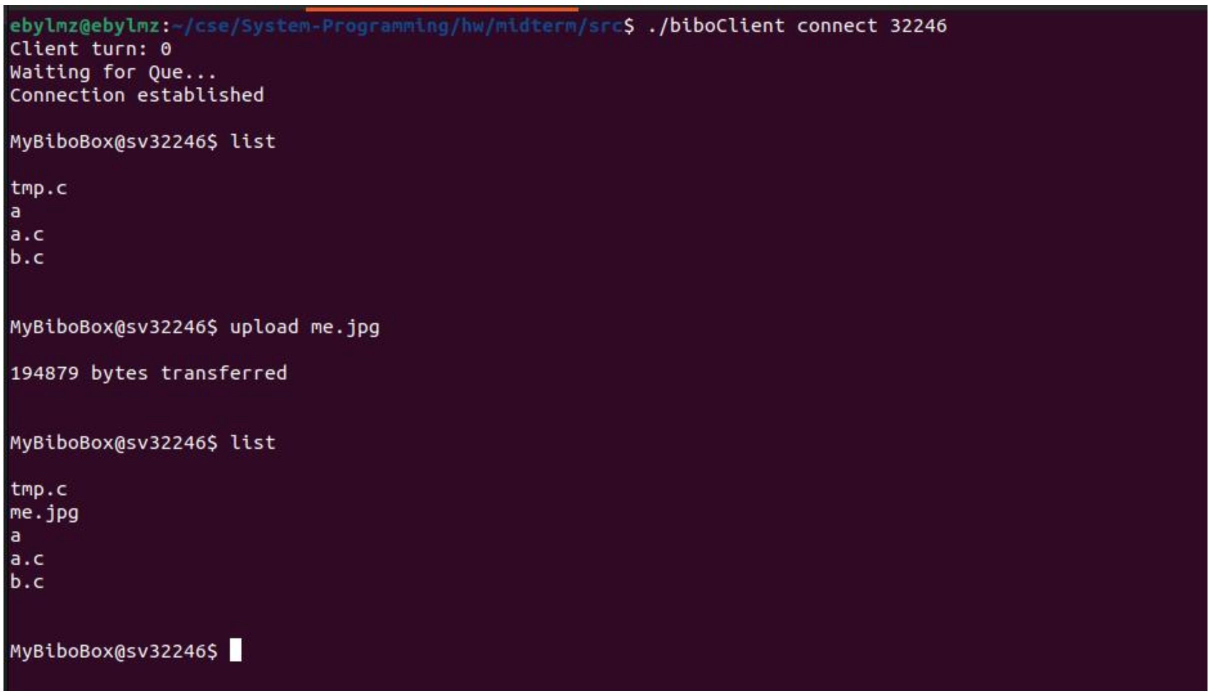
3.2.1.4 writeT


```
MyBiboBox@sv30234$ readF a.c

#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("Hello World\n");
}


MyBiboBox@sv30234$ writeT a.c UNSTOPPABLE


MyBiboBox@sv30234$ readF a.c

#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("Hello World\n");
}
UNSTOPPABLE

MyBiboBox@sv30234$ writeT a.c 3 CHEAP_THRILLS


MyBiboBox@sv30234$ readF a.c 3

CHEAP_THRILLSargc, char *argv[])


MyBiboBox@sv30234$
```

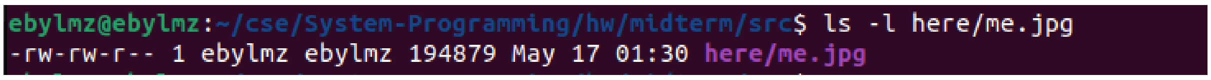
Unfortunately, writeT does not detect the string in the quotes :(

If the given line is exceeding the current number of line, then it returns error response. It could also be implemented as adding newlines and going to the target line and writing the string.

3.2.1.5 upload

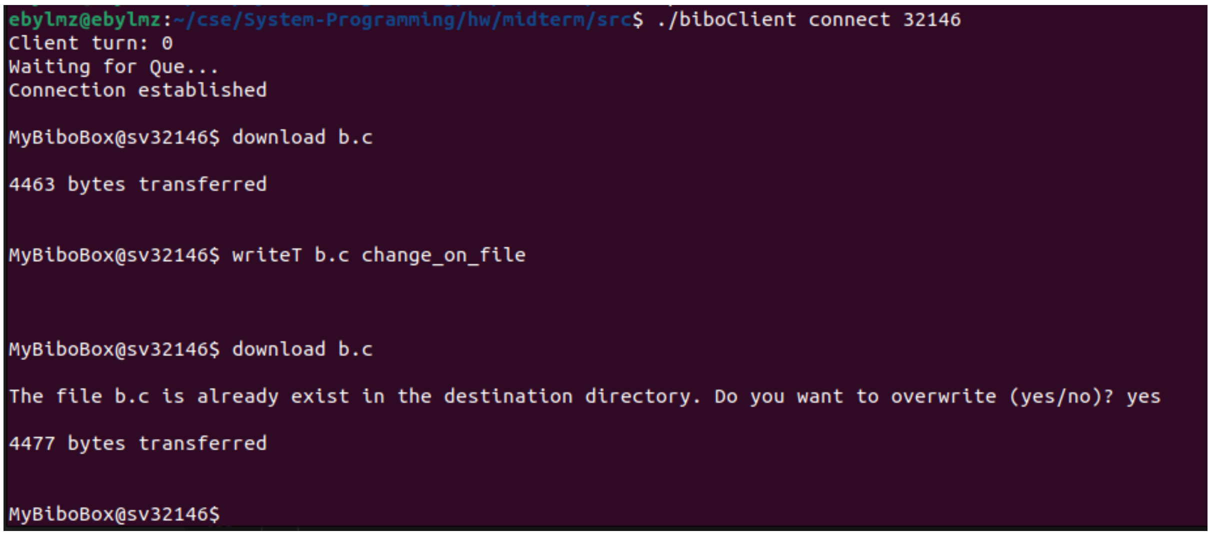


```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboClient connect 32246
Client turn: 0
Waiting for Que...
Connection established

MyBiboBox@sv32246$ list

tmp.c
a
a.c
b.c


MyBiboBox@sv32246$ upload me.jpg

194879 bytes transferred


MyBiboBox@sv32246$ list

tmp.c
me.jpg
a
a.c
b.c


MyBiboBox@sv32246$
```

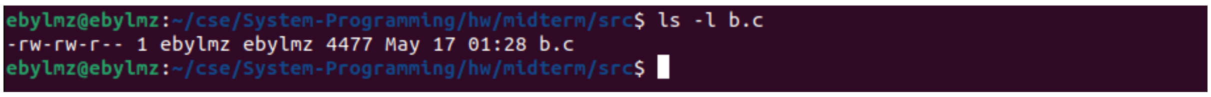


```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ls -l here/me.jpg
-rw-rw-r-- 1 ebylmz ebylmz 194879 May 17 01:30 here/me.jpg
```

3.2.1.6 download

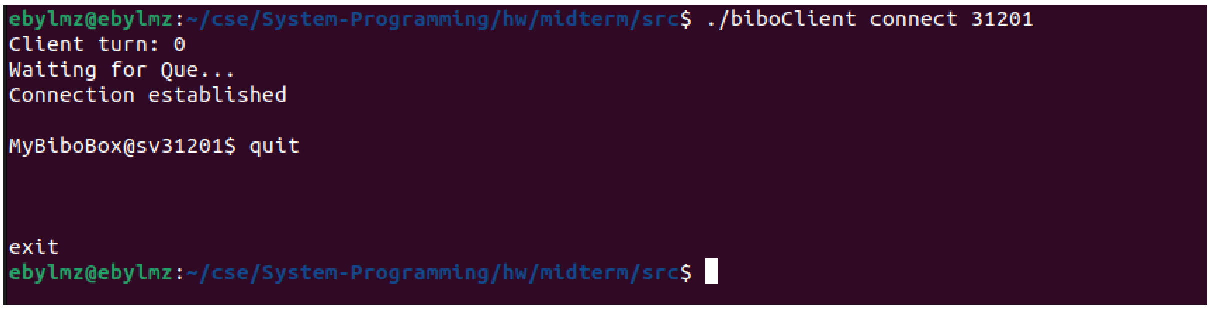


```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboClient connect 32146
Client turn: 0
Waiting for Que...
Connection established

MyBiboBox@sv32146$ download b.c

4463 bytes transferred


MyBiboBox@sv32146$ writeT b.c change_on_file



MyBiboBox@sv32146$ download b.c

The file b.c is already exist in the destination directory. Do you want to overwrite (yes/no)? yes

4477 bytes transferred


MyBiboBox@sv32146$
```



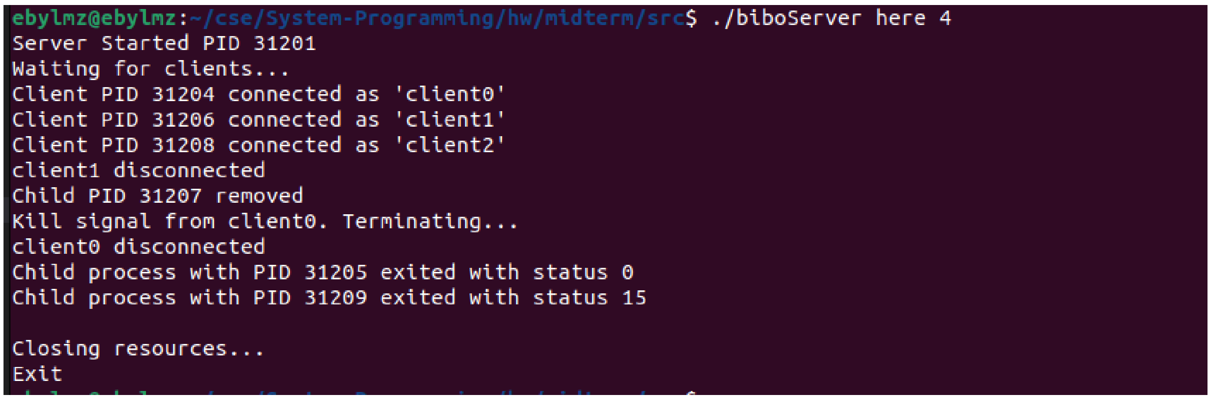
```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ls -l b.c
-rw-rw-r-- 1 ebylmz ebylmz 4477 May 17 01:28 b.c
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$
```

3.2.1.7 quit



```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboClient connect 31201
Client turn: 0
Waiting for Que...
Connection established

MyBiboBox@sv31201$ quit



exit
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$
```

3.2.1.8 killServer

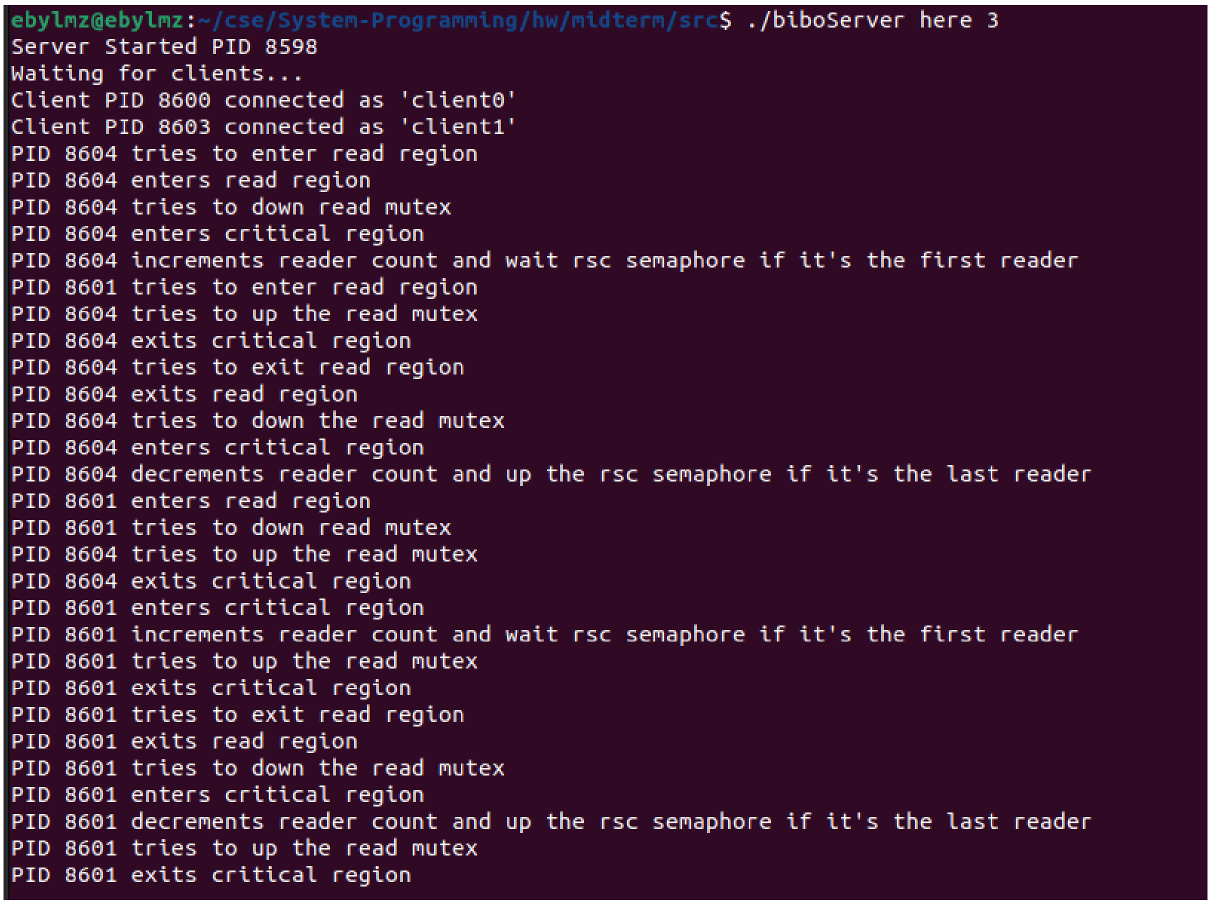


```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboServer here 4
Server Started PID 31201
Waiting for clients...
Client PID 31204 connected as 'client0'
Client PID 31206 connected as 'client1'
Client PID 31208 connected as 'client2'
client1 disconnected
Child PID 31207 removed
Kill signal from client0. Terminating...
client0 disconnected
Child process with PID 31205 exited with status 0
Child process with PID 31209 exited with status 15

Closing resources...
Exit
```

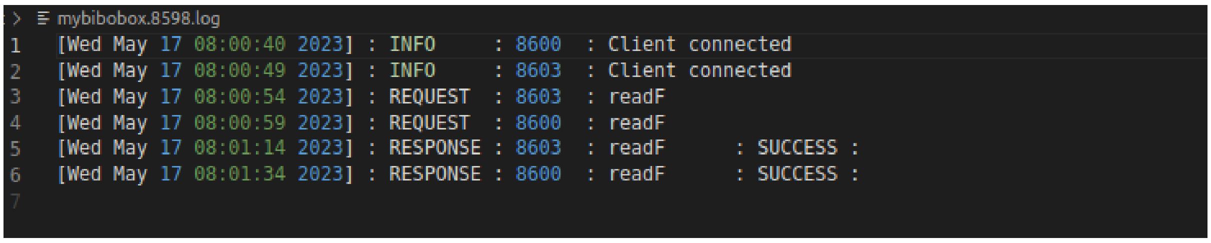
3.3 Providing Mutual Exclusion and Preventing Race Conditions

- More than one process may want to write log.
- readF and writeT requests may happen on the same file at the same time.
- During readF or writeT requests download or upload request may occur.
- During upload request list request may happen.

Two clients want to read the same file at the same time (For testing purposes the details are printed on the server screen, but they are commented out on the provided source code). sleep() function is used to see the results.



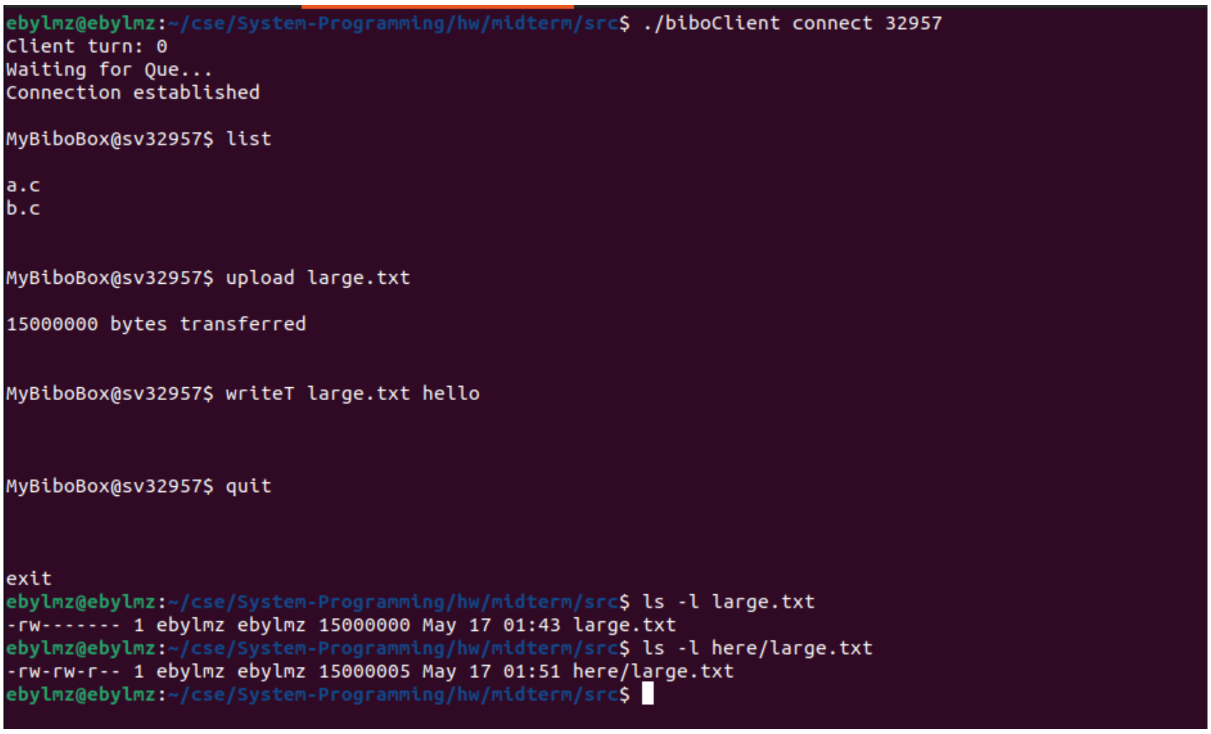
```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboServer here 3
Server Started PID 8598
Waiting for clients...
Client PID 8600 connected as 'client0'
Client PID 8603 connected as 'client1'
PID 8604 tries to enter read region
PID 8604 enters read region
PID 8604 tries to down read mutex
PID 8604 enters critical region
PID 8604 increments reader count and wait rsc semaphore if it's the first reader
PID 8601 tries to enter read region
PID 8604 tries to up the read mutex
PID 8604 exits critical region
PID 8604 tries to exit read region
PID 8604 exits read region
PID 8604 tries to down the read mutex
PID 8604 enters critical region
PID 8604 decrements reader count and up the rsc semaphore if it's the last reader
PID 8601 enters read region
PID 8601 tries to down read mutex
PID 8604 tries to up the read mutex
PID 8604 exits critical region
PID 8601 enters critical region
PID 8601 increments reader count and wait rsc semaphore if it's the first reader
PID 8601 tries to up the read mutex
PID 8601 exits critical region
PID 8601 tries to exit read region
PID 8601 exits read region
PID 8601 tries to down the read mutex
PID 8601 enters critical region
PID 8601 decrements reader count and up the rsc semaphore if it's the last reader
PID 8601 tries to up the read mutex
PID 8601 exits critical region
```



```
mybibobox.8598.log
[Wed May 17 08:00:40 2023] : INFO     : 8600  : Client connected
[Wed May 17 08:00:49 2023] : INFO     : 8603  : Client connected
[Wed May 17 08:00:54 2023] : REQUEST  : 8603  : readF
[Wed May 17 08:00:59 2023] : REQUEST  : 8600  : readF
[Wed May 17 08:01:14 2023] : RESPONSE : 8603  : readF      : SUCCESS :
[Wed May 17 08:01:34 2023] : RESPONSE : 8600  : readF      : SUCCESS :
```

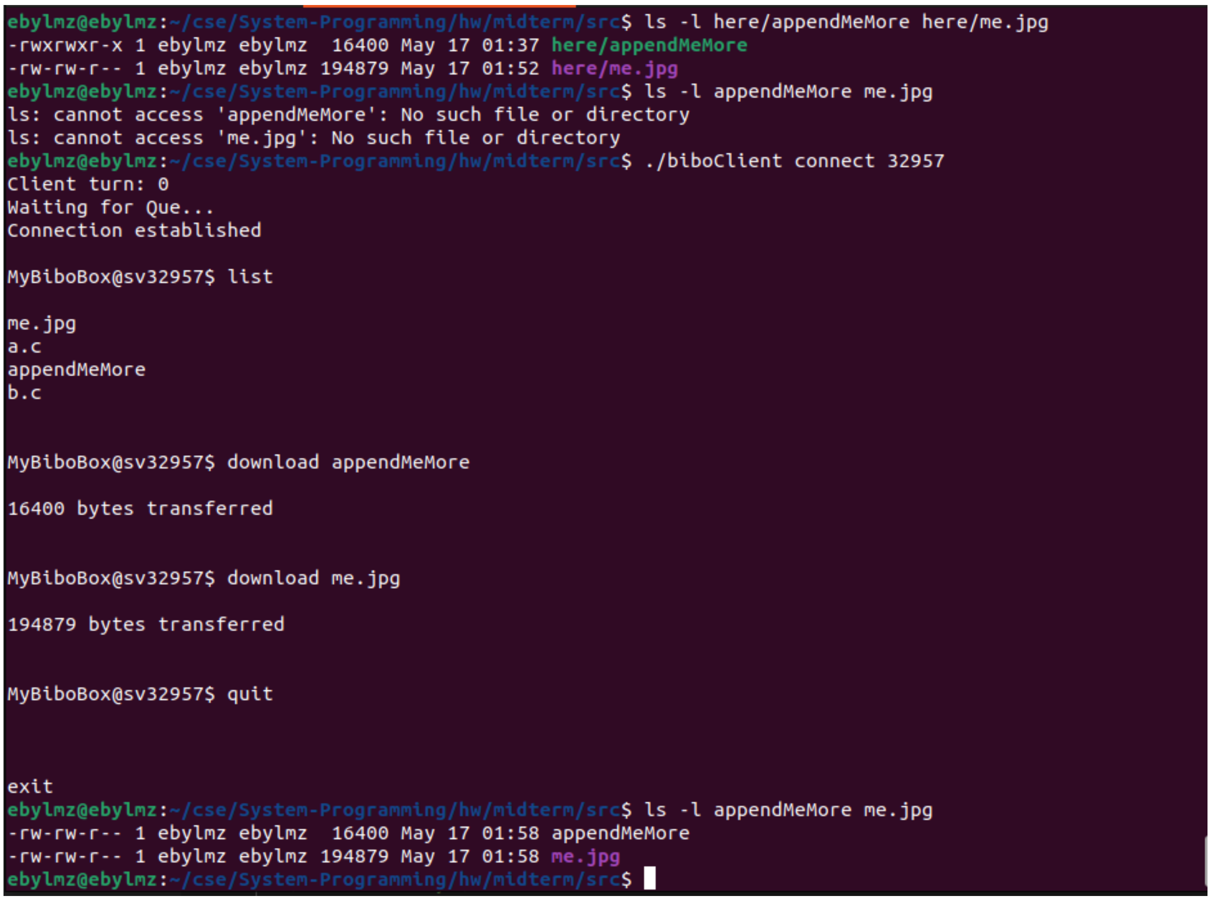
3.4 Large Files

The user commands such as upload, download, readF and writeT should be able to handle large files (i.e., > 10 MB)


```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboClient connect 32957
Client turn: 0
Waiting for Que...
Connection established

MyBiboBox@sv32957$ list

a.c
b.c


MyBiboBox@sv32957$ upload large.txt

15000000 bytes transferred


MyBiboBox@sv32957$ writeT large.txt hello



MyBiboBox@sv32957$ quit



exit
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ls -l large.txt
-rw------- 1 ebylmz ebylmz 15000000 May 17 01:43 large.txt
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ls -l here/large.txt
-rw-rw-r-- 1 ebylmz ebylmz 15000005 May 17 01:51 here/large.txt
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ 
```

3.5 Different File Formats (text, binary)

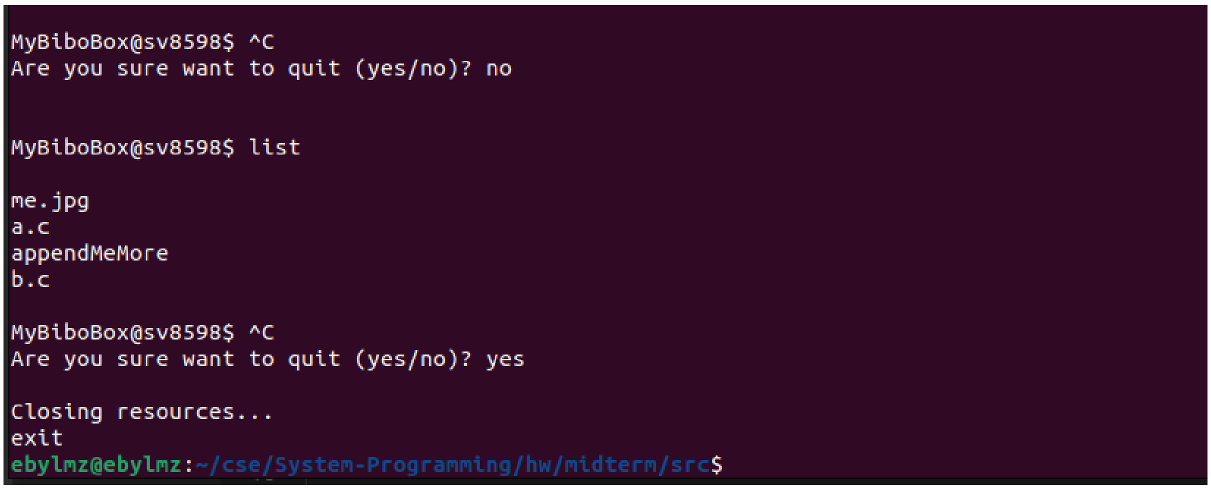
Different file formats (i.e., text, binary) should be handled.


```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ls -l here/appendMeMore here/me.jpg
-rwxrwxr-x 1 ebylmz ebylmz  16400 May 17 01:37 here/appendMeMore
-rw-rw-r-- 1 ebylmz ebylmz 194879 May 17 01:52 here/me.jpg
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ls -l appendMeMore me.jpg
ls: cannot access 'appendMeMore': No such file or directory
ls: cannot access 'me.jpg': No such file or directory
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboClient connect 32957
Client turn: 0
Waiting for Que...
Connection established

MyBiboBox@sv32957$ list

me.jpg
a.c
appendMeMore
b.c


MyBiboBox@sv32957$ download appendMeMore

16400 bytes transferred


MyBiboBox@sv32957$ download me.jpg

194879 bytes transferred


MyBiboBox@sv32957$ quit



exit
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ls -l appendMeMore me.jpg
-rw-rw-r-- 1 ebylmz ebylmz  16400 May 17 01:58 appendMeMore
-rw-rw-r-- 1 ebylmz ebylmz 194879 May 17 01:58 me.jpg
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ 
```

3.6 Signals

Both server and client should be able to handle the signals properly.

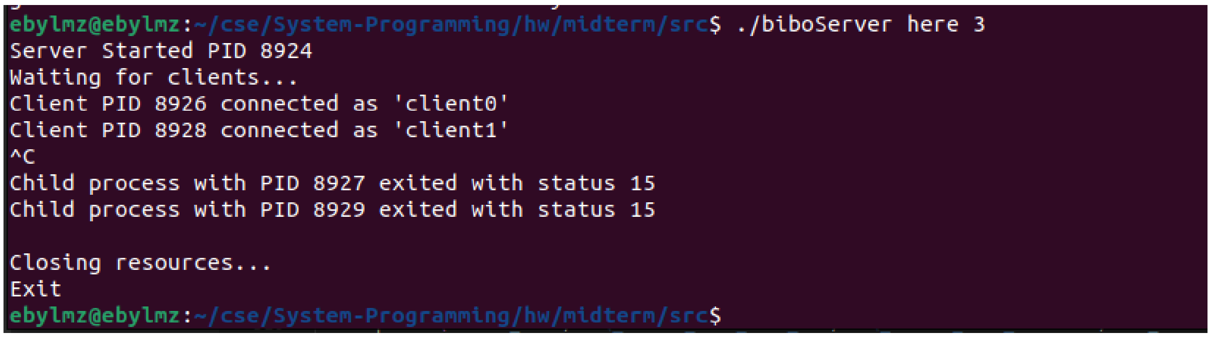
3.6.1 SIGINT/SIGTERM in Client



```
MyBiboBox@sv8598$ ^C
Are you sure want to quit (yes/no)? no


MyBiboBox@sv8598$ list

me.jpg
a.c
appendMeMore
b.c

MyBiboBox@sv8598$ ^C
Are you sure want to quit (yes/no)? yes

Closing resources...
exit
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$
```

Resources are closed properly and exited gracefully.

3.6.2 SIGINT/SIGTERM in Server



```
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$ ./biboServer here 3
Server Started PID 8924
Waiting for clients...
Client PID 8926 connected as 'client0'
Client PID 8928 connected as 'client1'
^C
Child process with PID 8927 exited with status 15
Child process with PID 8929 exited with status 15

Closing resources...
Exit
ebylmz@ebylmz:~/cse/System-Programming/hw/midterm/src$
```

Childs servers are killed, resources are closed properly and exited gracefully.