CSE 344 - Final Project
Sena Özbelen - 1901042601

This program works on both sides: client and server

client.c

```
./BibakBOXClient client port_num ip_addr
```

Client part takes 3 arguments: directory path, port number, ip address of server.
Before execution, it creates the directory if it doesn't exist. If it does, then it clears the directory.

```c
/* if it exists, then remove it to recreate it */
char* argv[] = { "rm", "-rf", path, NULL };
char* envp[] = { NULL };

if(execve("/bin/rm", argv, envp) == -1)
{
    perror("Error executing command");
    exit(1);
}
```

After clearing the directory, server files are directly copied to client's directory (sync).
When client is connected to server at first, server starts to send its files through socket.

Before connecting, if all threads are working, then "FULL" is sent to client and client is terminated.

Until it gets a signal (SIGINT), it executes a loop. Firstly, it gets messages from server and updates its directory. It sleeps for 5 seconds because of performance issues. Then, it sends the changes to server through socket.

Messages have a pattern:
A1|path = the file on this path is added
D2|path = the file on this path is deleted
M3|path = the file on this path is modified

It gets messages until "END" or "EXIT" is received. "END" is an indicator that the message is ended. "EXIT" means that server is terminated so client should be terminated. While getting messages, if A1 or M3 is received, then it opens the file by extracting the path from the message and it sets the flag to 1. When this flag is 1, it writes the content of the file to the file in the client's directory because after A1 or M3, the content is sent. If D2 is received, it simply removes it from the directory.

After changing the files, client has to send changes. It gets the current directory state.

```c
struct file_info /* file information struct */
{
    char path[DEFAULT_SIZE];
    time_t modified_time;
};
```

```
struct dir_state /* directory state struct */
{
    struct file_info* files;
    int num_files;
};
```

It traverses the whole directory and saves the file_info for each file. After that, it compares the current state to the previous state. If there is a new file in the current state, then it sends A1 message. If a file is not found in the current state, then it sends D2 message. It also compares modified_time variable so that it can spot the modifications and send M3 message.
It sends the main message (A1, etc) and then sends the content of the file if necessary. After all messages are sent, it sends “END” so that server stops receiving the messages. Also, if a signal (SIGINT) is caught, client sends “EXIT” and server removes the client. It sets terminate flag to 1 and client program is terminated.

server.c

```
./BibakBOXServer server max_client port_num
```

Server takes 3 arguments, directory path, max client number and port number.
Server prints its IP addresses when it is started.

Server works in a similar way that client works in.

If a client wants to connect to server and server has a free thread, it assigns a thread for the client and starts executing.

It sends the files of server directory to client to be in sync. Then, it starts to execute a loop.
Firstly, it compares the current state of the directory to previous one **(similar to client)** and if there is a change, it sends it through socket. After that, it sleeps for 5 seconds and then it starts to receive changes from the client side. For every change, server modifies its directory. When “END” is received, it stops receiving messages and goes back to the beginning of the loop. If it receives “EXIT”, it removes the client.
If server catches a signal (SIGINT), then it sends “EXIT” to all clients and sets the terminate flag to 1.

Each thread and the main server have signal handlers to catch the signals.

Server and client send-receive mechanism are the same.
When a message is received from other side (valid for both client and server), it stores them in a temp array because directory state is changed so received files are also re-sent to the other side and this becomes an infinite loop. My solution is to store them and check if this file is in temp array before sending a message. After that, it resets the temp array for new messages.

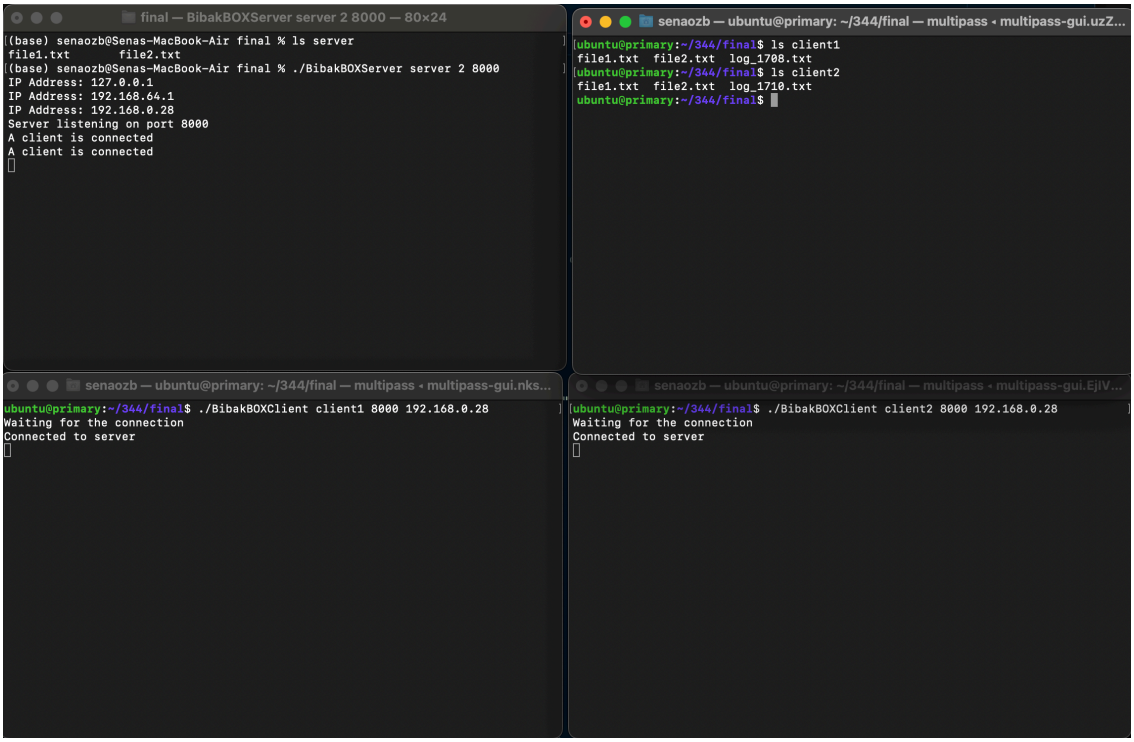
Dataflow : server -> clients , sleep , clients -> server

Test Cases

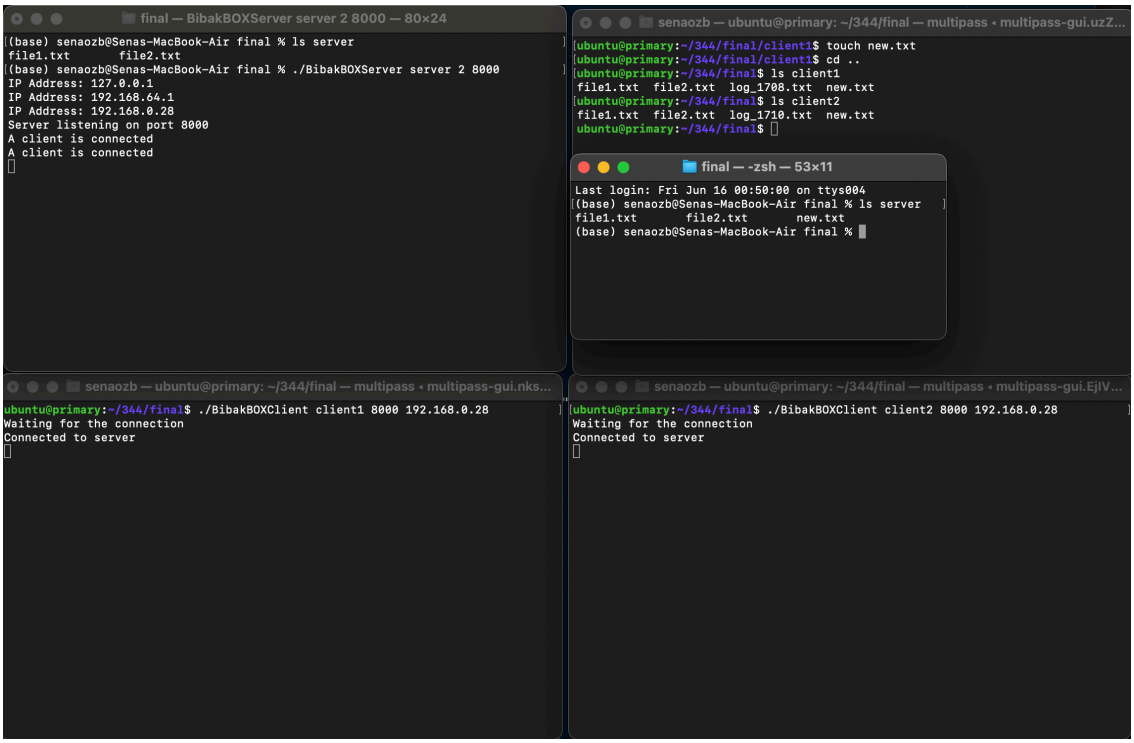
*Program can’t handle subdirectories.
*Program sometimes can’t send file contents properly from client side on different machines (MacOS - Ubuntu)

Running on MacOS as server and Ubuntu(Multipass) as clients

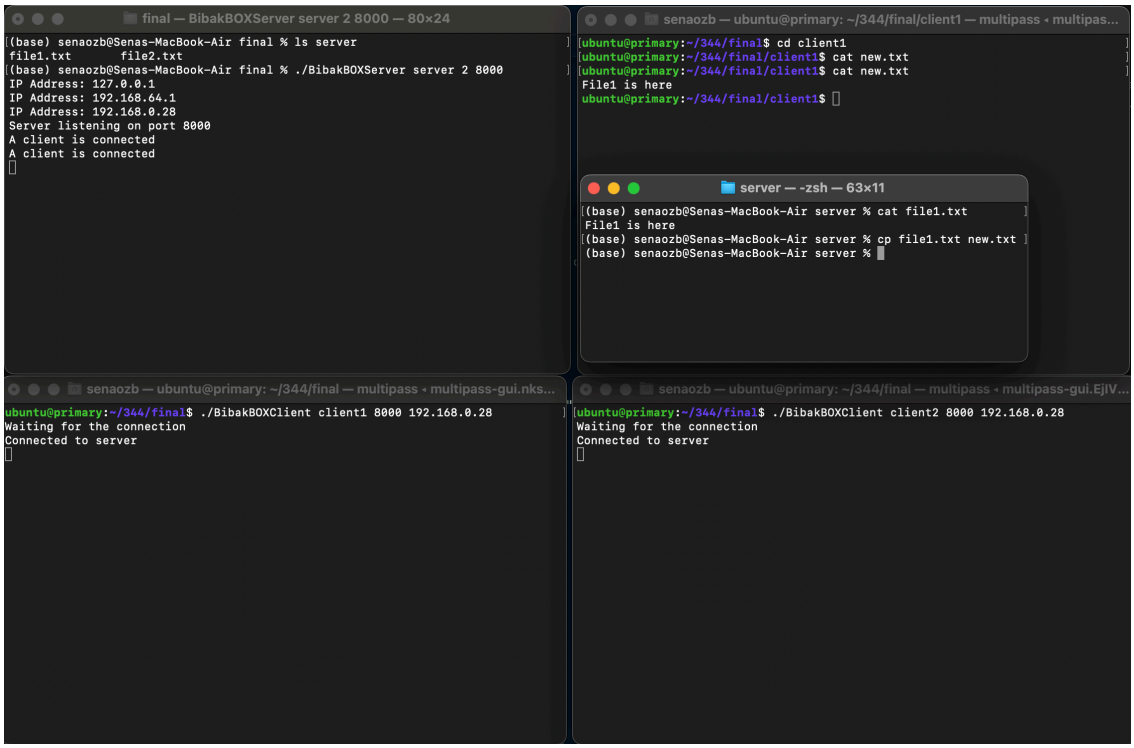
```
./BibakBOXServer server 2 8000
./BibakBOXClient client1 8000 ip_add
./BibakBOXClient client2 8000 ip_add
```



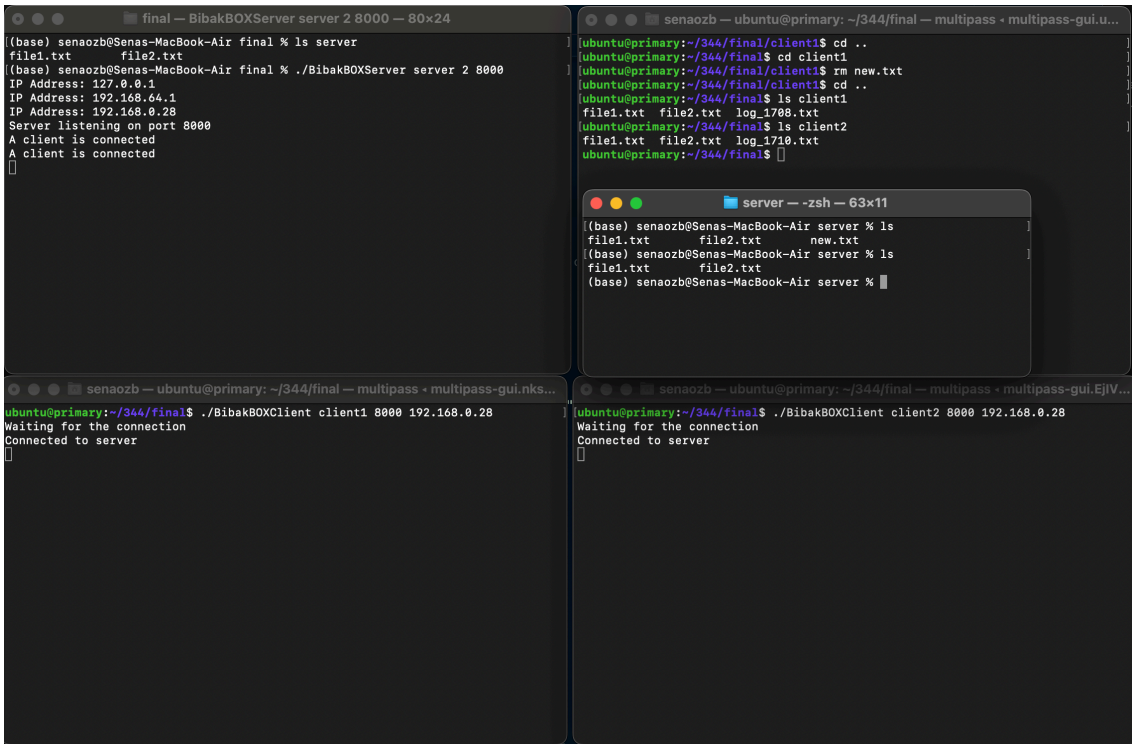
2 clients are connected to server and their directories are in sync now.



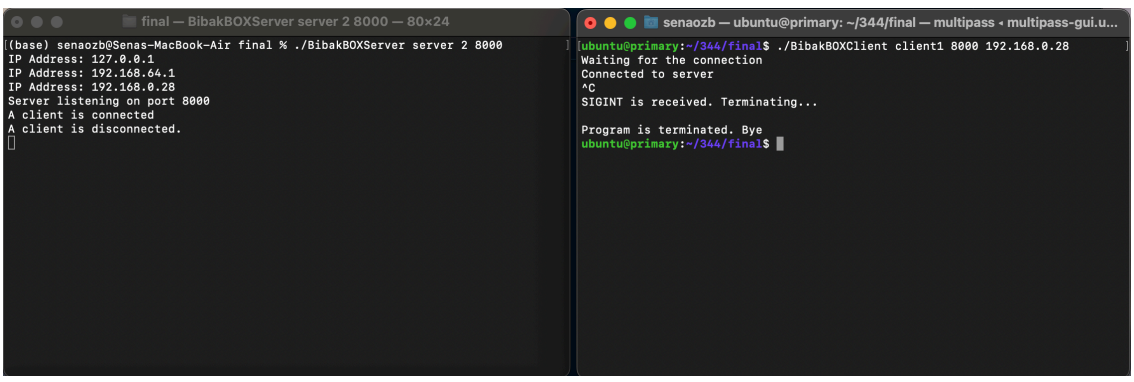
In client1’s directory, a file “new.txt” is created. Now, every directory has this new file.



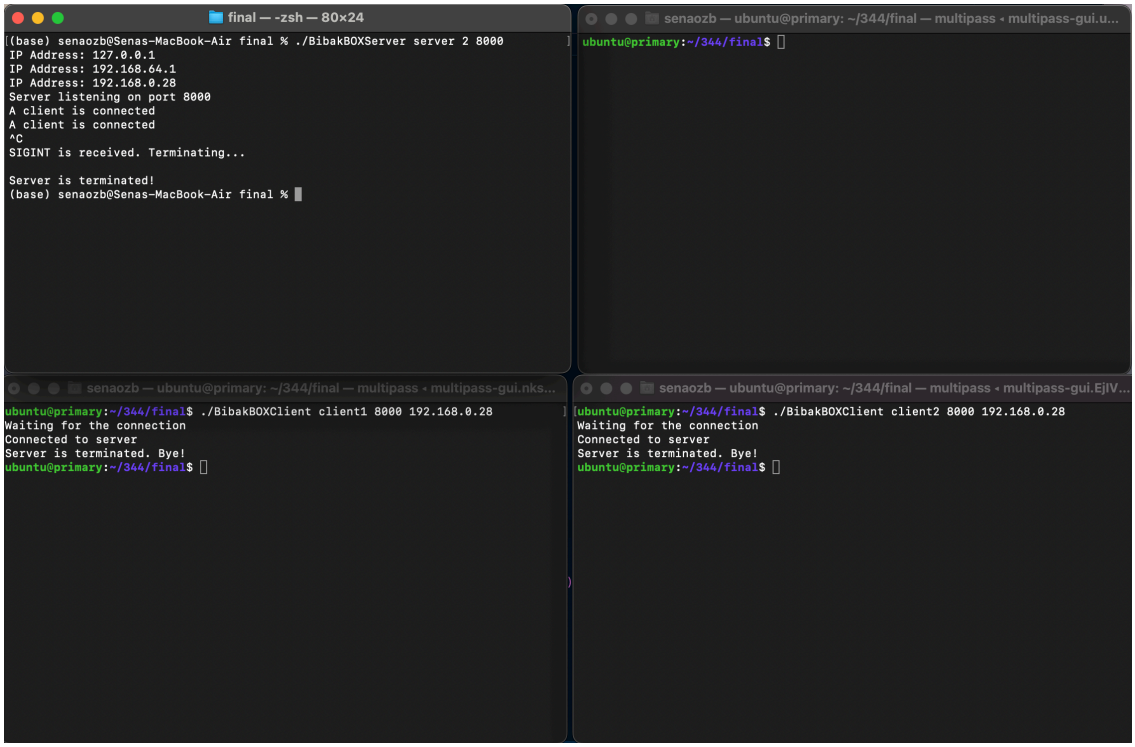
The content of file1.txt is copied to new.txt in server's directory. We can see that clients modified their files as well. It was empty before copying the content. After copy operation, cat command shows the new content.



new.txt is removed from client1's directory. We can see that client2 and server also removed the file.

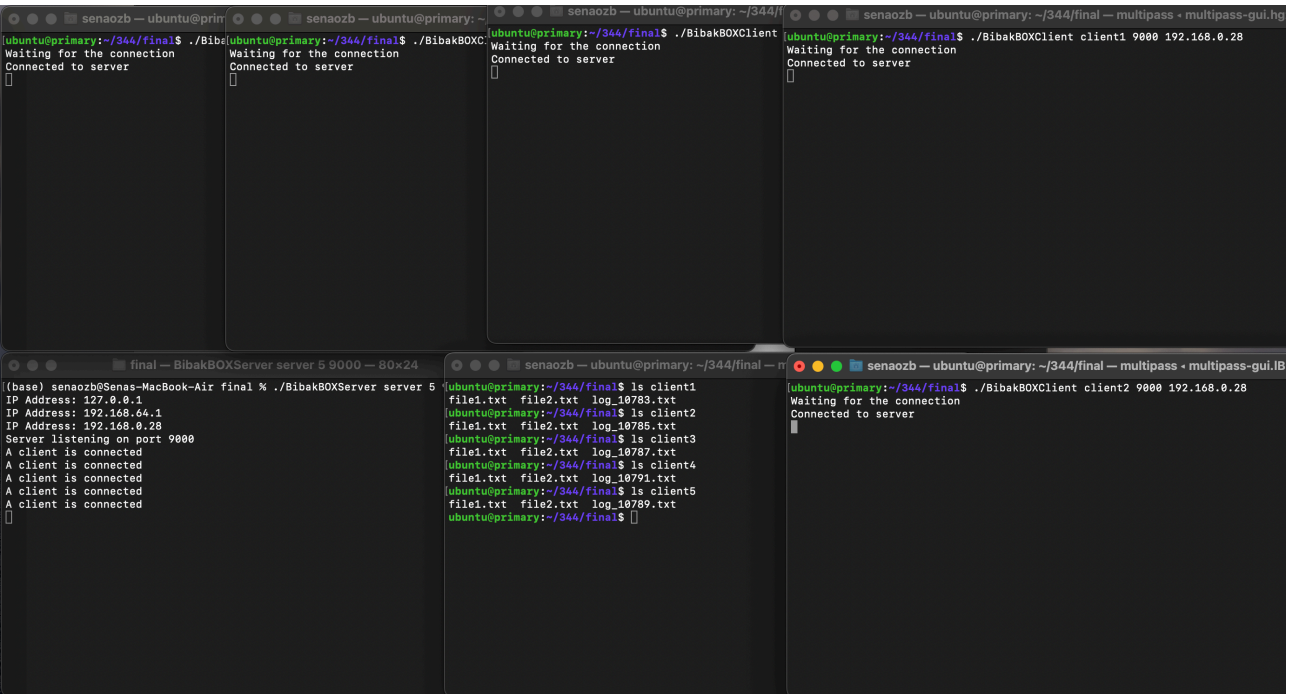


Client gets CTRL-C so client is removed.

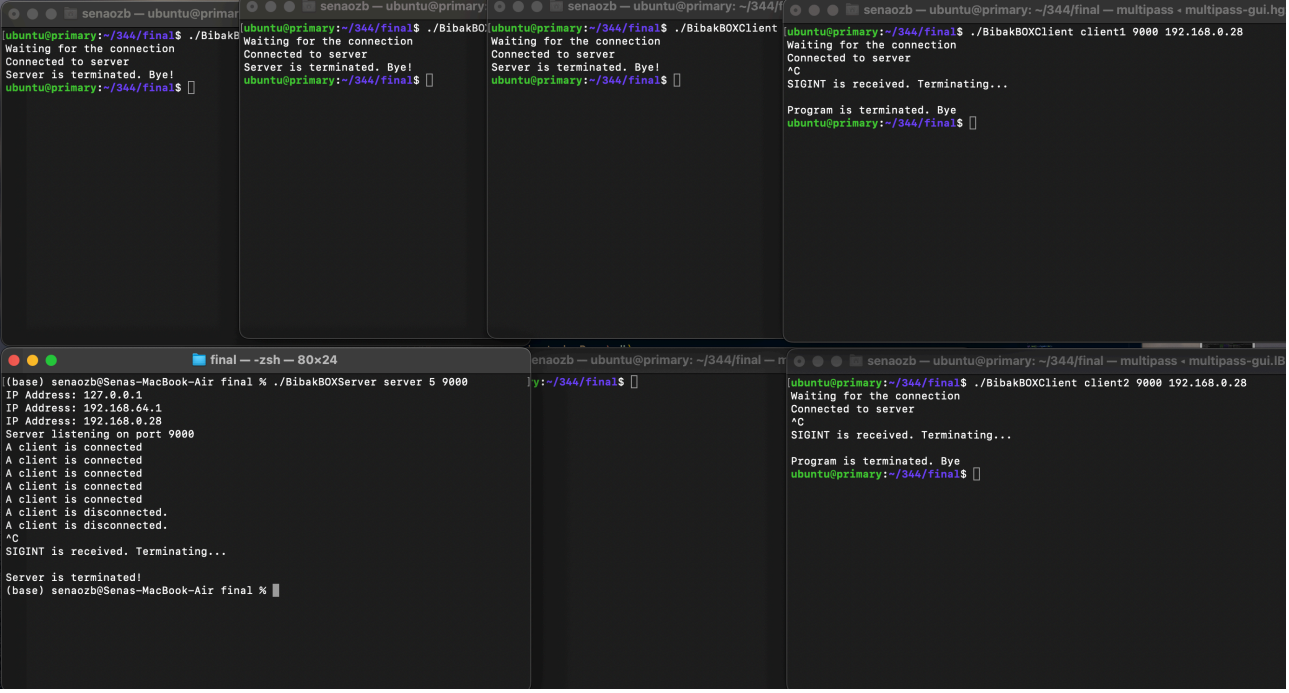


Server caught CTRL-C. Therefore, it sent EXIT message to clients and clients are terminated.

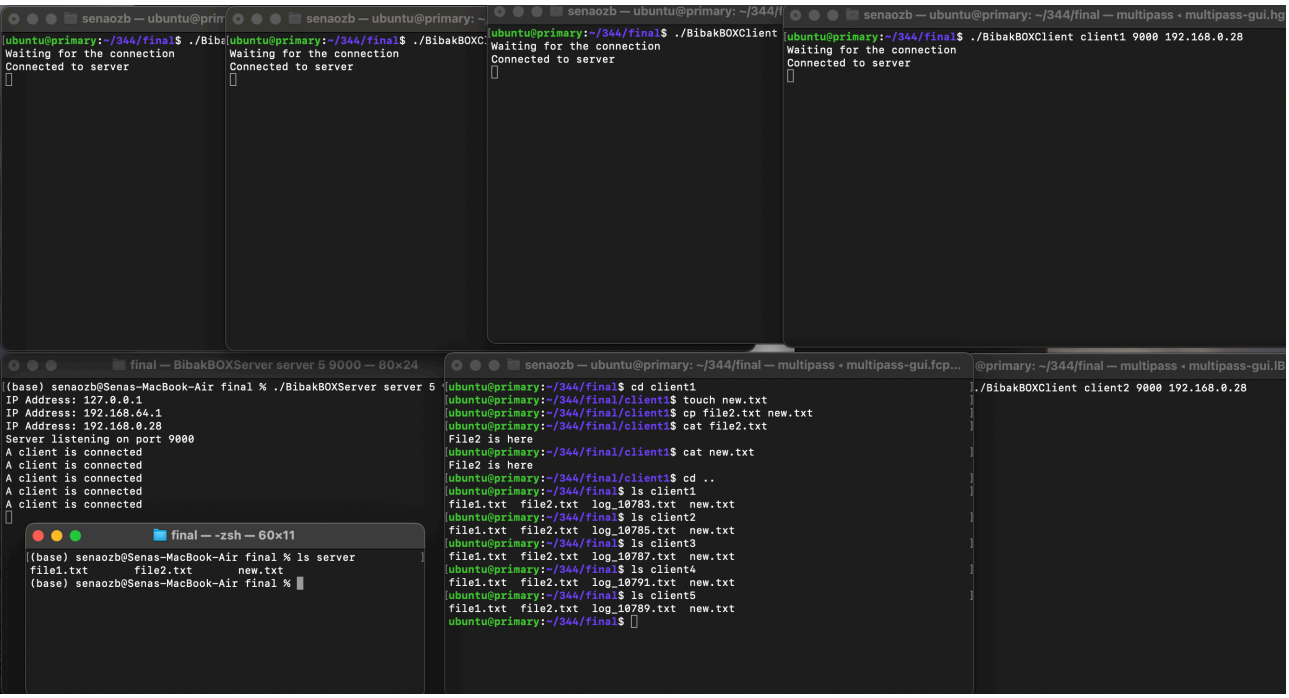
```
./BibakBOXServer server 5 9000
./BibakBOXClient client1 9000 ip_add
./BibakBOXClient client2 9000 ip_add
./BibakBOXClient client3 9000 ip_add
./BibakBOXClient client4 9000 ip_add
./BibakBOXClient client5 9000 ip_add
```



5 clients are connected to server



2 clients terminated with CTRL-C and rest terminated by server



Add a new file and then remove it