

CSE 344 System Programming

Final Project

1801042090 Cem Bozkurt

In this final project I implemented like a Dropbox system. My program can create, delete, and update file and this changes reflect on both side and vice versa as we asked to implement.

So first my server is started and client is connected to the server.

For the first ever interaction between client and server I have a queue structure to store socket file descriptor inside it after client is connected to server I store that descriptor inside of the queue and wake my worker threads. Total number of worker threads are equal to pool size that decided by the user when server is started through command line input.

So that all my client is handled by the one thread if there is more client than the thread than it will be wait until one of client that currently communicate with thread to finish.

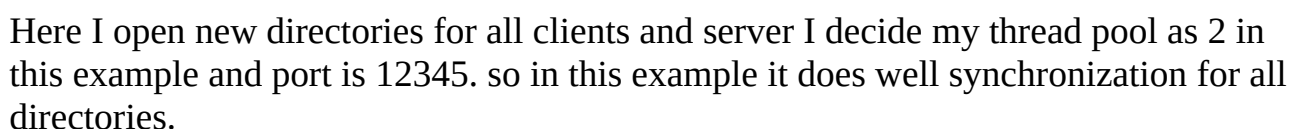
After successfully connect and start communicate between thread and client, I started to synchronize, but to not overheat the cpu I use sleep function inside the client so that it check for synchronization not all the time but with the 5second periods. I do NOT use sleep for the synchronization only use it for not the overheat cpu and it makes more sense to check it periodically.

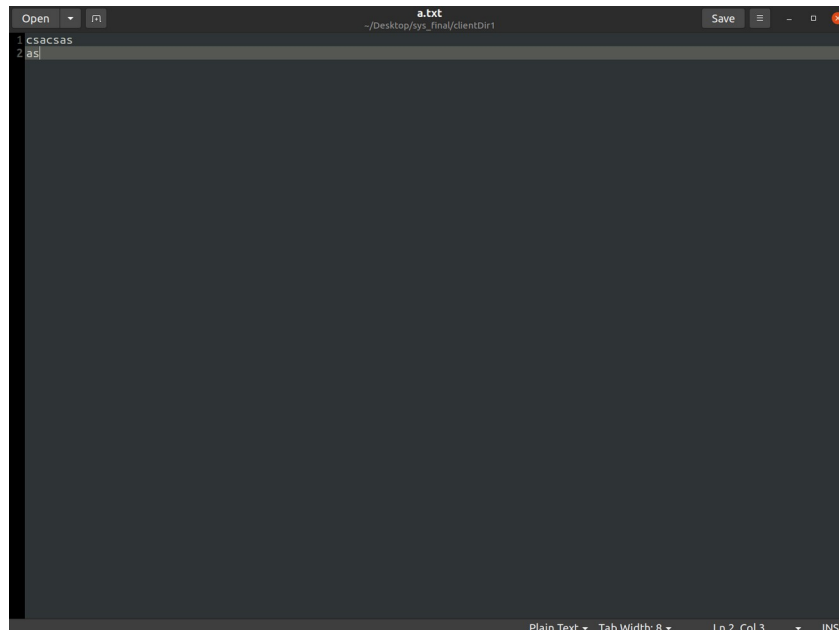
My synchronization method is checking by the server. Client send its current files to the server with their path and modified times only, it does not send the data that they contains.

Also in my server threads I keep the old files information both servers and the clients in case of there is delete operation. I understand there is delete operation if there is a file that not inside the new files but it is inside the old files than I understand there is delete operation and I perform that operation.

I also handle SIGINT signal, when SIGINT comes to client when there is file transfer it hold the SIGINT wait until transfer is done later it terminates the program gracefully. When the program is finished my thread on the server side understand that client is disconnecting so it makes it self available for the new clients.

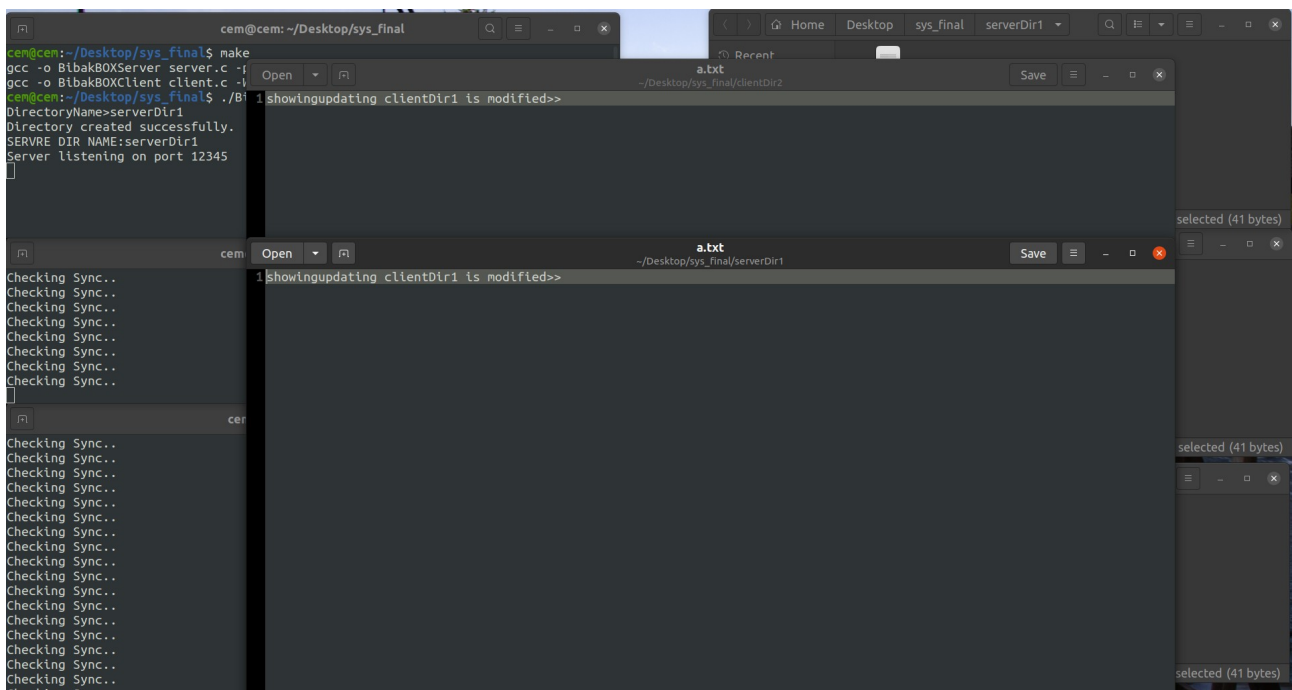
When I run my synchronization method I also write into my log file. For the naming of log files I use counter inside the critical section and name them client#number inside them I keep the information of which file is deleted and how this operation happens like from server to client server \rightarrow client.





The screenshot shows a text editor window titled 'a.txt' with the file path '~/Desktop/sys_final/clientDir1'. The editor contains two lines of text: '1 csacsas' and '2 asl'. The status bar at the bottom indicates 'Plain Text', 'TabWidth: 8', 'Ln 2, Col 3', and 'INS'.

This is the content of a.txt I modified it here and show the other directors to prove my program also does the updating.



The screenshot shows a terminal window with the following output:

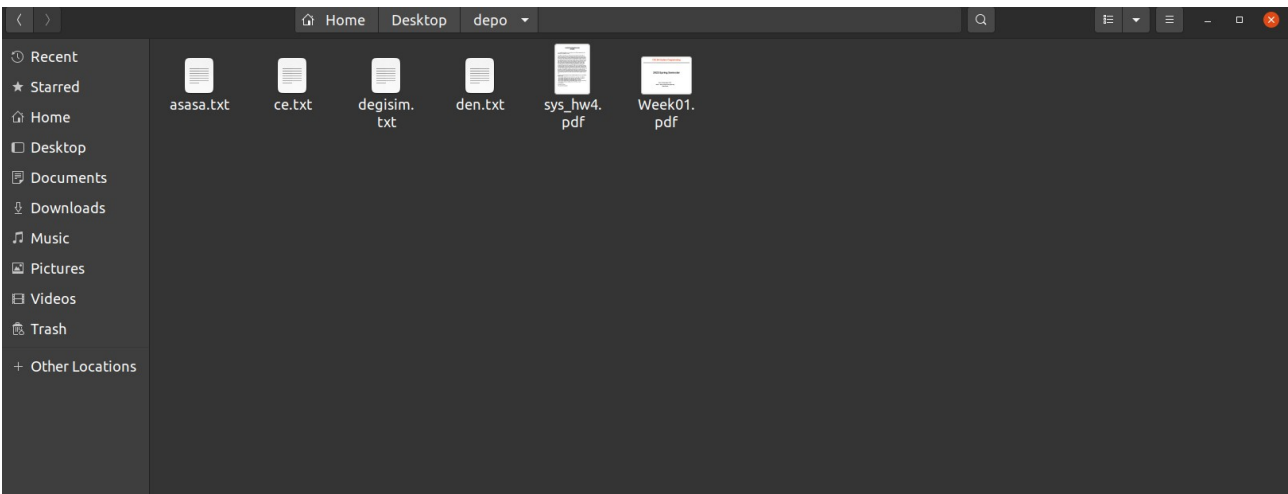
```
cem@cem: ~/Desktop/sys_final$ make
gcc -o BibakBOXServer server.c -f
gcc -o BibakBOXClient client.c -f
cem@cem: ~/Desktop/sys_final$ ./BibakBOXServer
DirectoryNameserverDir1
Directory created successfully.
SERVRE DIR NAME:serverDir1
Server listening on port 12345

cem@cem: ~/Desktop/sys_final$ ./BibakBOXClient
1showingupdating clientDir1 is modified>>
```

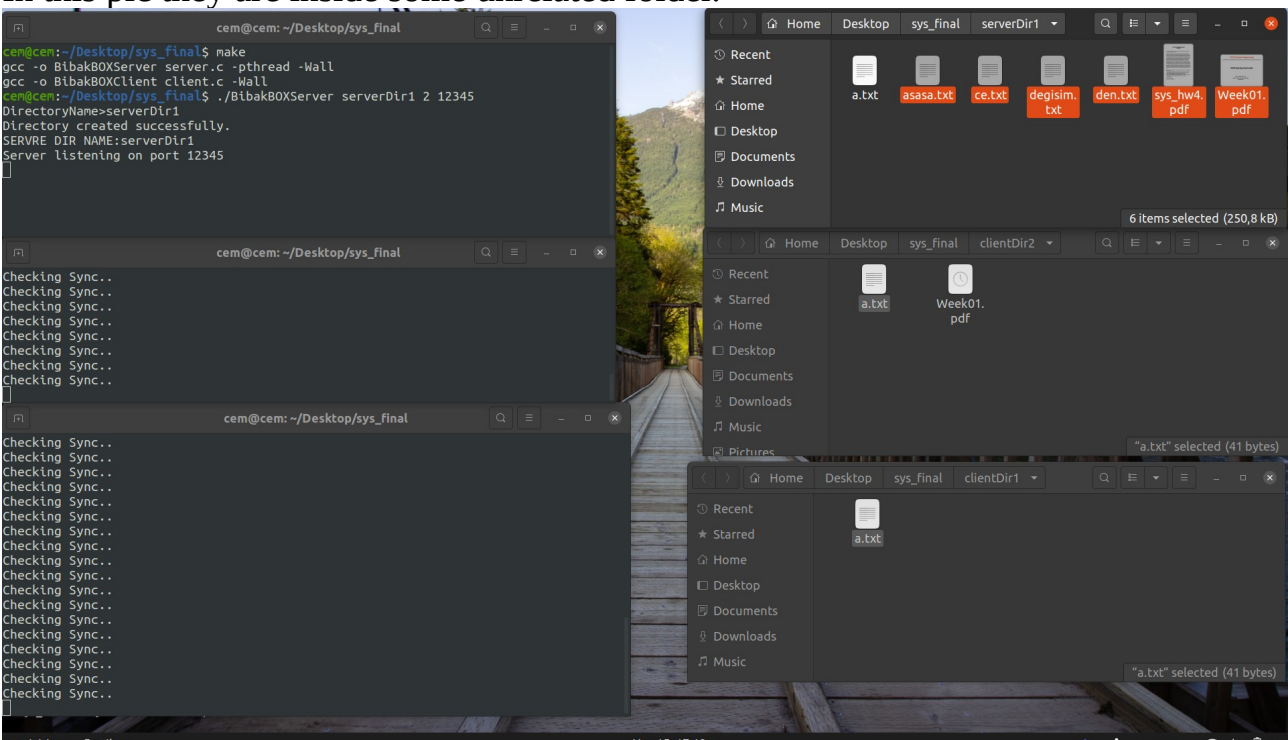
The terminal also shows a list of 'Checking Sync..' messages. To the right, three instances of the 'a.txt' text editor are visible, each showing the same content: '1showingupdating clientDir1 is modified>>'. The status bar on the right indicates 'selected (41 bytes)'.

Both on serverDir1 and clientDir2 changes reflect the here so update operation is successful.

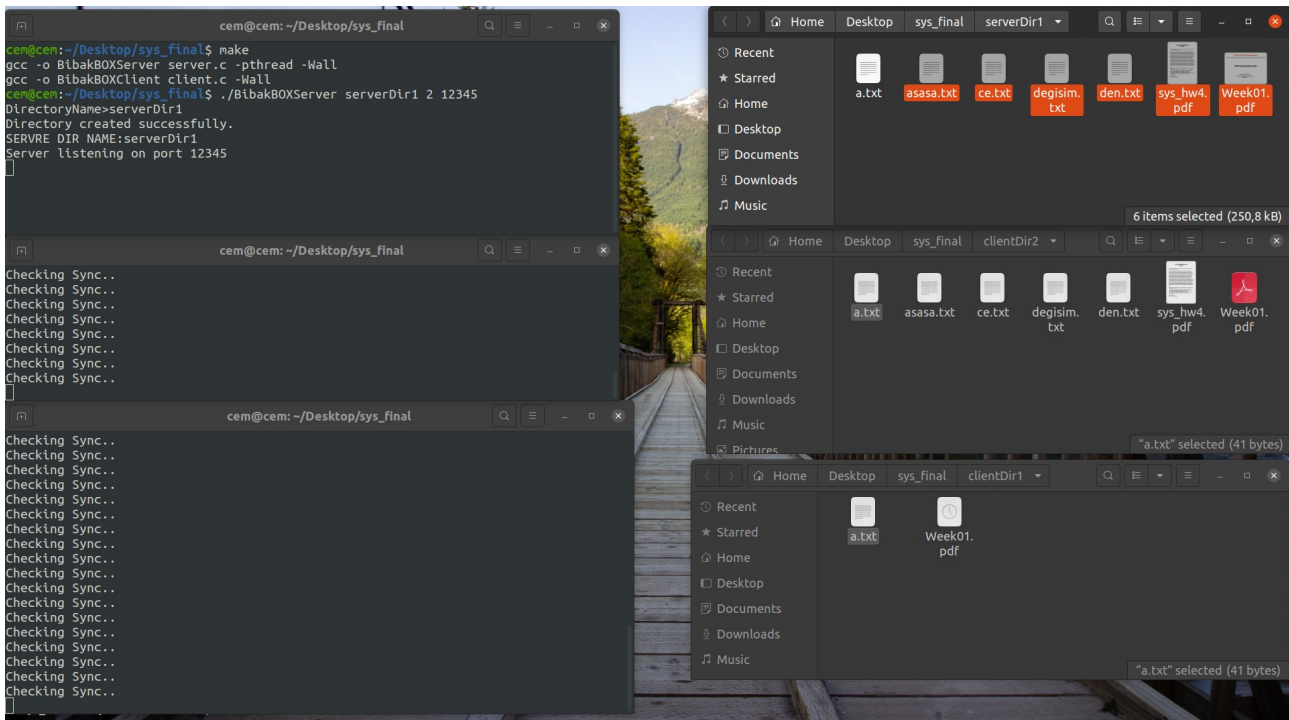
Multiple file download. Below I will show I put multiple file inside to the server and program synchronize the all directories.



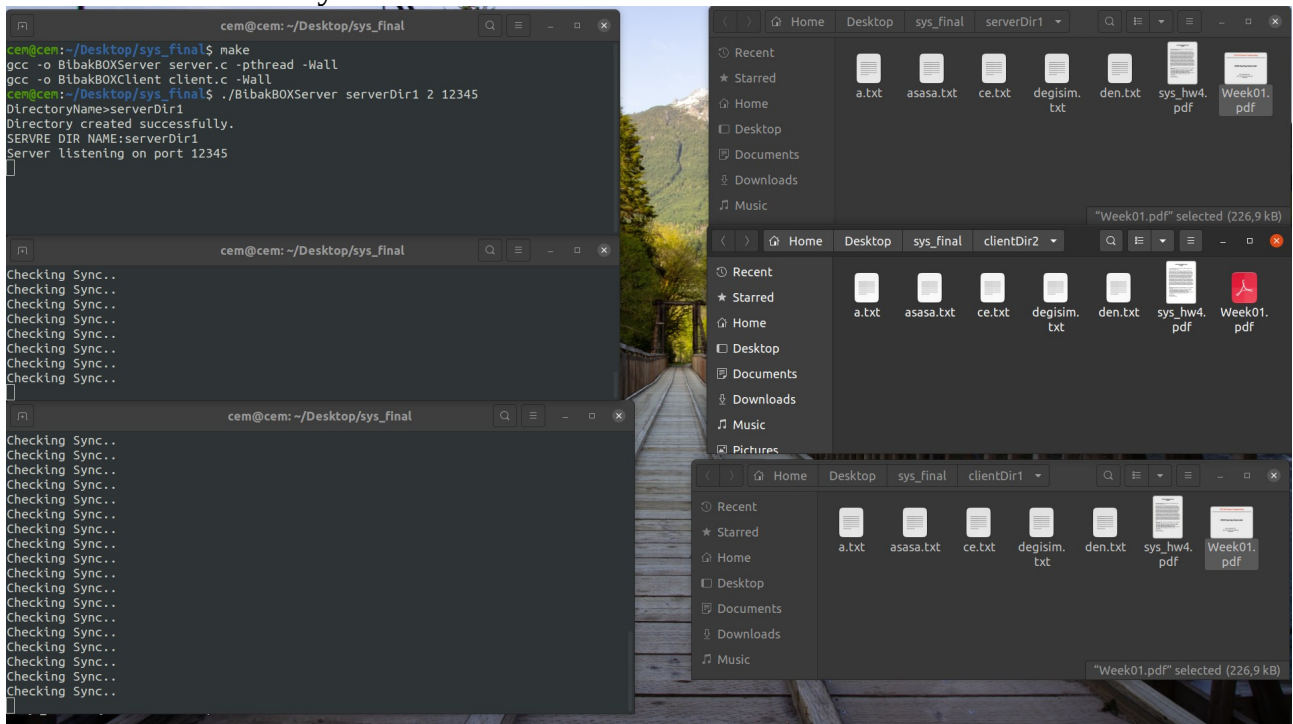
In this pic they are inside some unrelated folder.



Here it first update the clientDir2

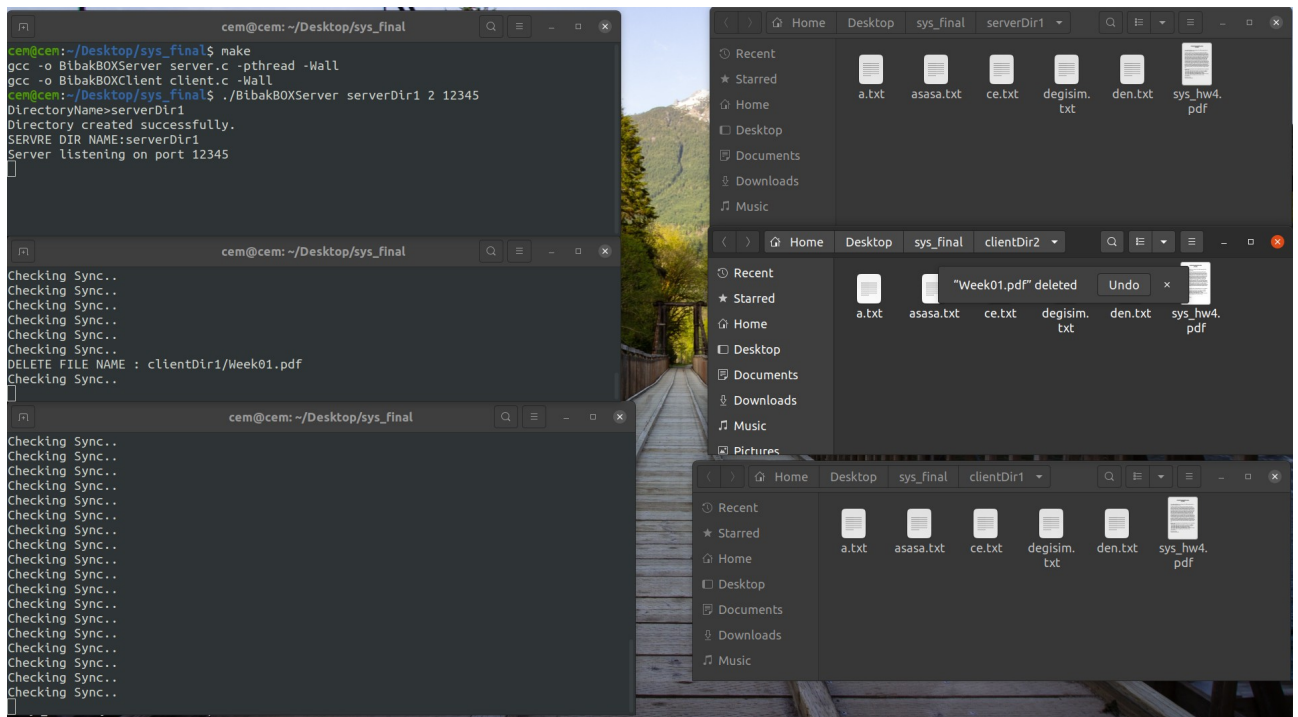


And here finish the synchronization.



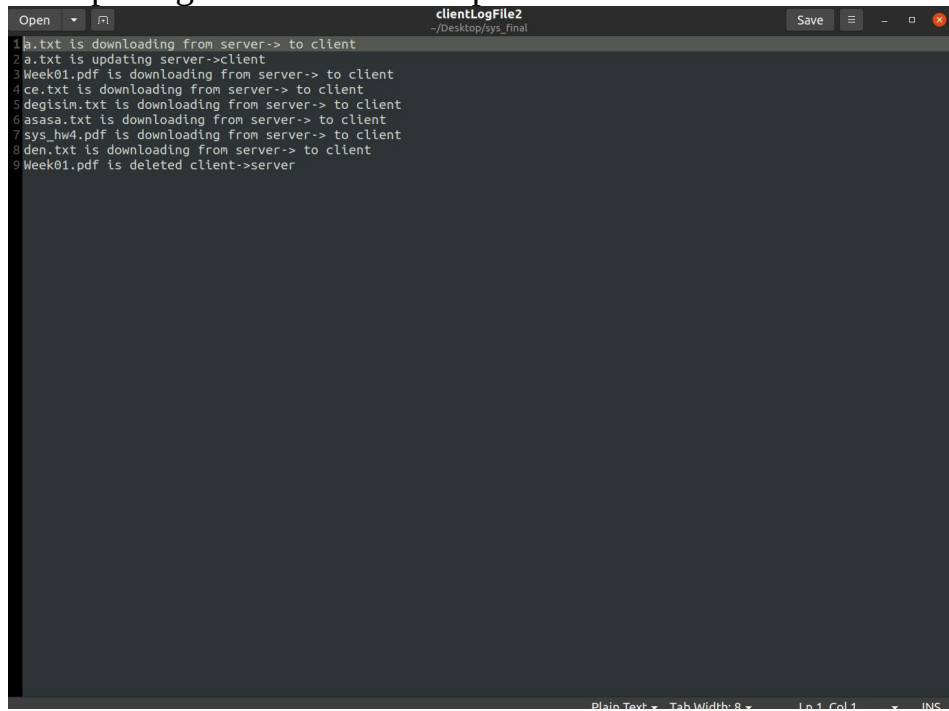
[illegible]

I delete week01.pdf from clientDir2 and then



it removed from all directories.

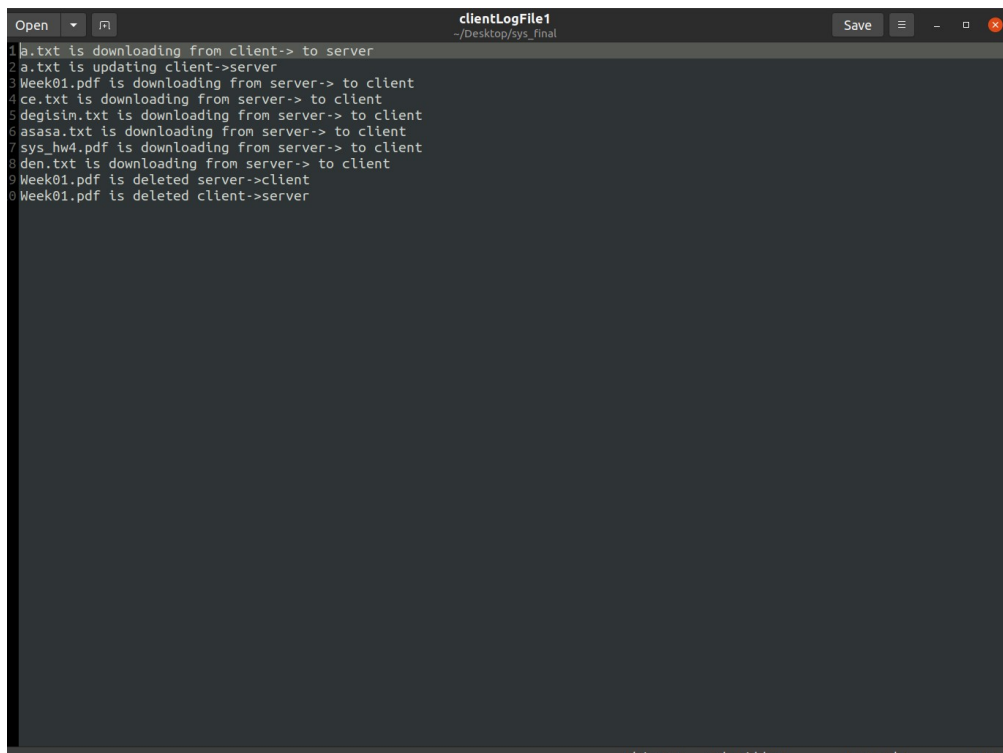
Here is the example log file of all of this operation
asd



```
Open  clientLogFile2  Save
~/Desktop/sys_final

1 a.txt is downloading from server-> to client
2 a.txt is updating server->client
3 Week01.pdf is downloading from server-> to client
4 ce.txt is downloading from server-> to client
5 degisim.txt is downloading from server-> to client
6 asasa.txt is downloading from server-> to client
7 sys_hw4.pdf is downloading from server-> to client
8 den.txt is downloading from server-> to client
9 Week01.pdf is deleted client->server

Plain Text  Tab Width: 8  Ln: 1, Col: 1  IN5
```



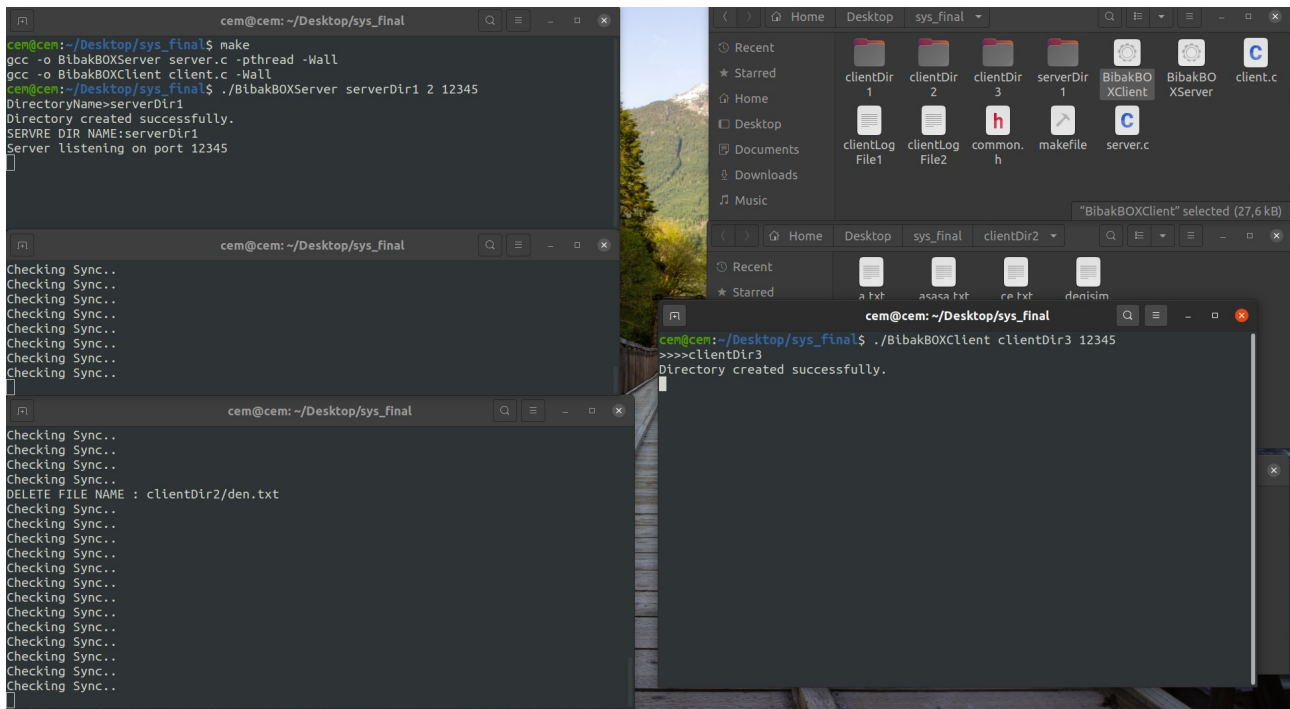
```
Open  clientLogFile1  Save
~/Desktop/sys_final

1 a.txt is downloading from client-> to server
2 a.txt is updating client->server
3 Week01.pdf is downloading from server-> to client
4 ce.txt is downloading from server-> to client
5 degisim.txt is downloading from server-> to client
6 asasa.txt is downloading from server-> to client
7 sys_hw4.pdf is downloading from server-> to client
8 den.txt is downloading from server-> to client
9 Week01.pdf is deleted server->client
10 Week01.pdf is deleted client->server

Plain Text  Tab Width: 8  Ln: 1, Col: 1  IN5
```

so I have been show you when all directors are empty, creation, updating, deletion scenario.

What happens if all threads are occupied and new client is arrived?



Here as you can see clientDir3 is connected but not handled by the worker thread yet you can understand from it does not print Checking Sync... and log file for the 3th client you can see it from sys_final directory which is my work space.

The screenshot displays a Kali Linux desktop environment with a mountain landscape wallpaper. Three terminal windows and one file manager window are open.

- Top-left terminal:** Shows the execution of `make` and `BibakBOXClient` commands. The output indicates that the directory was created successfully, the server name is `serverDir1`, and the server is listening on port `12345`. It also shows that one client is leaving.
- Bottom-left terminal:** Shows the execution of `BibakBOXClient` with arguments `clientDir2 12345`. The output shows multiple "Checking Sync.." messages and a "DELETE FILE NAME : clientDir2/den.txt" message.
- Right-side window:** A file manager showing the contents of the `sys_final` directory. It lists subdirectories `clientDir` (1, 2, 3), `serverDir` (1), `BibakBOXClient`, `BibakBOXServer`, and `client.c`. It also shows files `clientLog` (File1, File2, File3), `common.h`, `makefile`, and `server.c`. The `BibakBOXClient` file is selected.
- Bottom-right terminal:** Shows the execution of `BibakBOXClient` with arguments `clientDir3 12345`. The output shows multiple "Checking Sync.." messages and a "Directory created successfully." message.

```
clientLogFile3
~/Desktop/sys_final

1 ce.txt is downloading from server-> to client
2 degisim.txt is downloading from server-> to client
3 a.txt is downloading from server-> to client
4 asasa.txt is downloading from server-> to client
```

So in the end this is show how server and clients work when server receive SIGINT signal.

The screenshot shows a terminal window with the following output:

```

cem@cem: ~/Desktop/sys_final
DirectoryNames=serverDir1
Directory created successfully.
SERVRE DIR NAME:serverDir1
Server listening on port 12345
One of client is leaving..
AC*****
Program will be closed please wait!
*****
Sending close message to the client
Sending close message to the client
cem@cem:~/Desktop/sys_final$

```

The file explorer shows the directory structure:

- clientDir 1, clientDir 2, clientDir 3
- serverDir 1
- BibakBO XClient, BibakBO XServer
- client.c
- clientLog File1, clientLog File2, clientLog File3
- common.h, makefile, server.c

The terminal window shows the server checking sync status and sending close messages to clients.

The screenshot shows the terminal window with the following output:

```

cem@cem:~/Desktop/sys_final
DirectoryNames=serverDir1
Directory created successfully.
SERVRE DIR NAME:serverDir1
Server listening on port 12345
One of client is leaving..
AC*****
Program will be closed please wait!
*****
Sending close message to the client
Sending close message to the client
cem@cem:~/Desktop/sys_final$

```

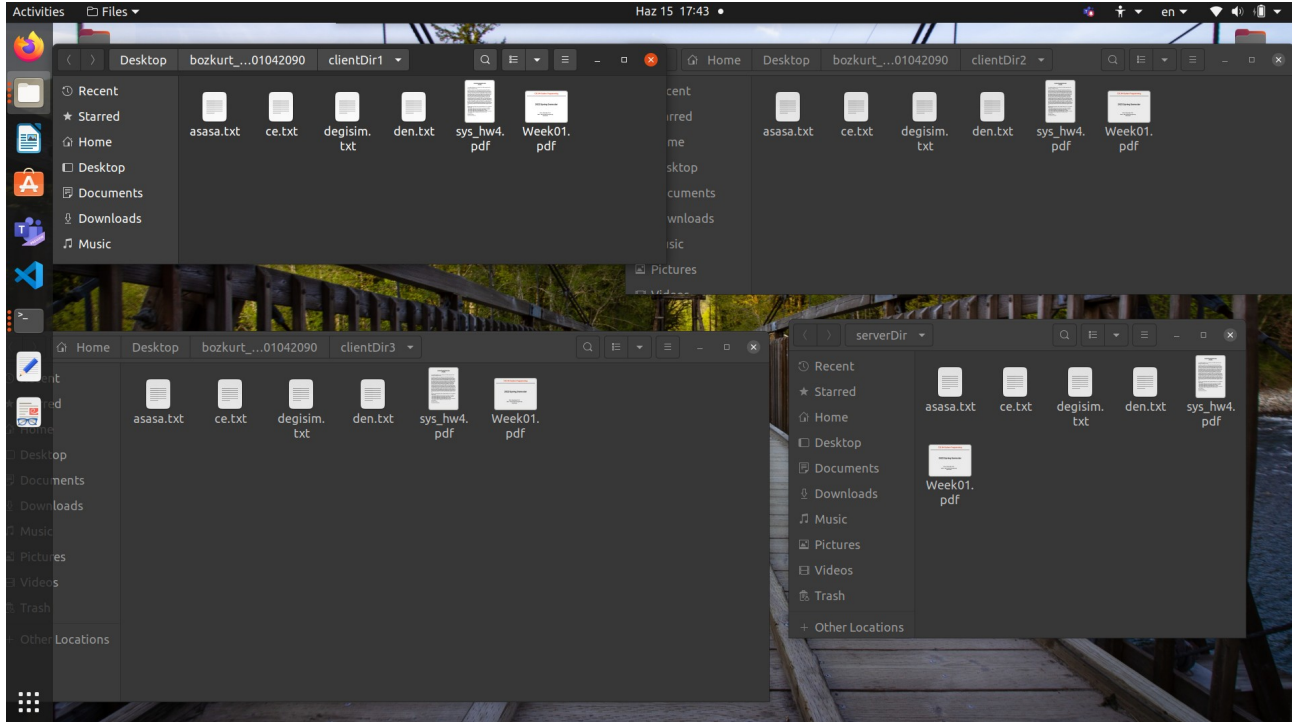
The file explorer shows the directory structure:

- clientDir 1, clientDir 2, clientDir 3
- serverDir 1
- BibakBO XClient, BibakBO XServer
- client.c
- clientLog File1, clientLog File2, clientLog File3
- common.h, makefile, server.c

The terminal window shows the server checking sync status and sending close messages to clients. The output indicates that the server is closed and the program is terminating.

As you can see it sends all clients to message to close and before closing they complete the transition in case of they were in the middle of it.

For the multiple file and larger file it takes time for program the synchronize all of them.



For example all of them was empty and then I copy paste all files from another directory to inside clientDir1. And here it takes 1-2 minute to complete synchronize the all directories. (This is different example here I make my pool size 3 this is why 3 client can work at the same time)