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.

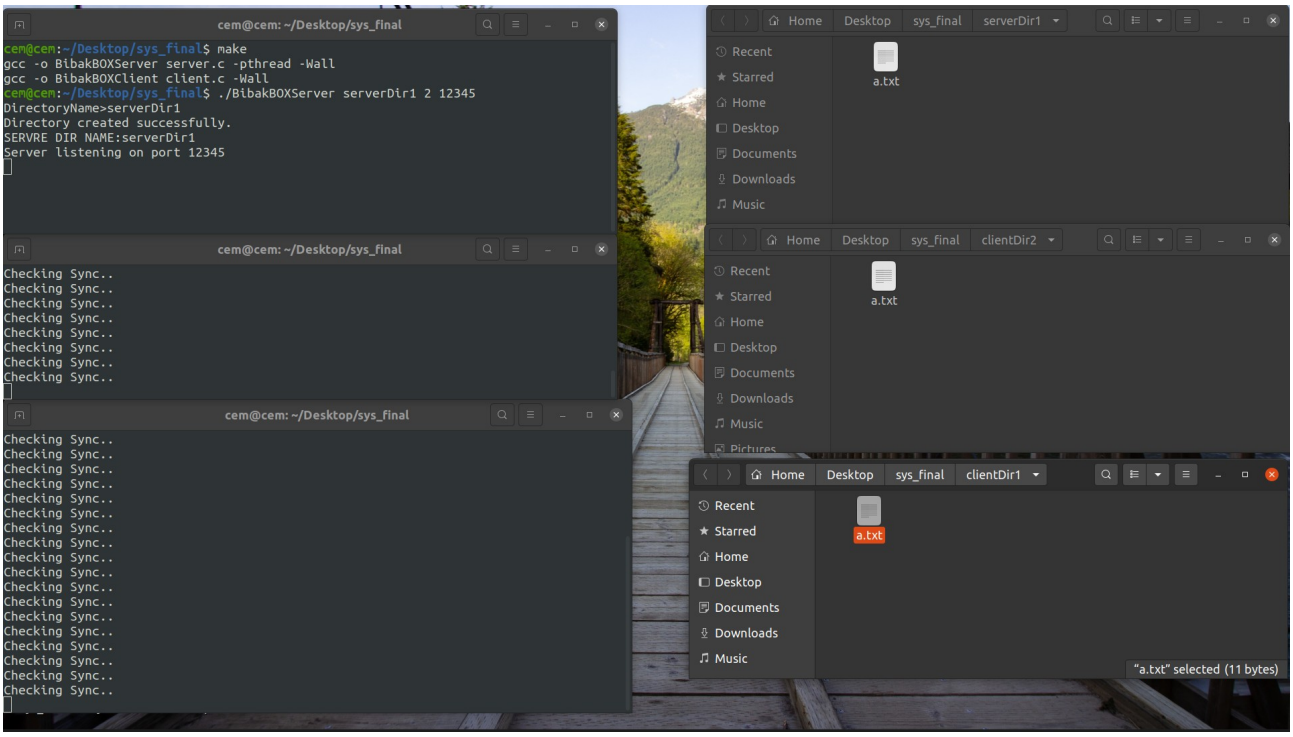
My synchronization method is checking by the server. Client send its current files to the server with their path and modified times only, it does not send the data that they contains.

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.

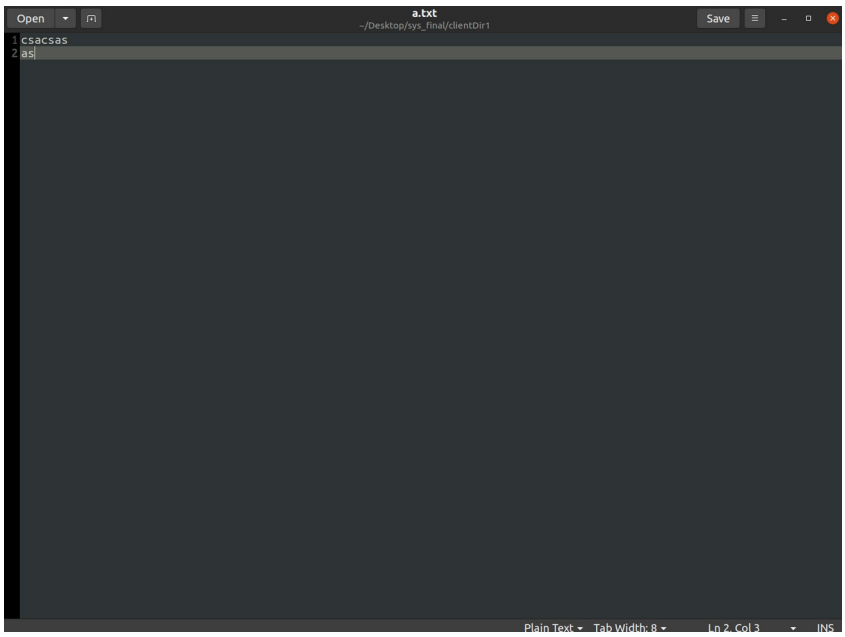
For other operation such like create, update I handle them also inside of this operation.
I also handle SIGINT signal, when SIGINT comes to client when there is file transfer it hold the SIGINT wait until transfer is done later it terminates the program gracefully. When the program is finished my thread on the server side understand that client is disconnecting so it makes it self available for the new clients.

When server side receive SIGINT signal it handle gracefully again, first it close it self but before it it send signal to the clients so that clients also closed gracefully.

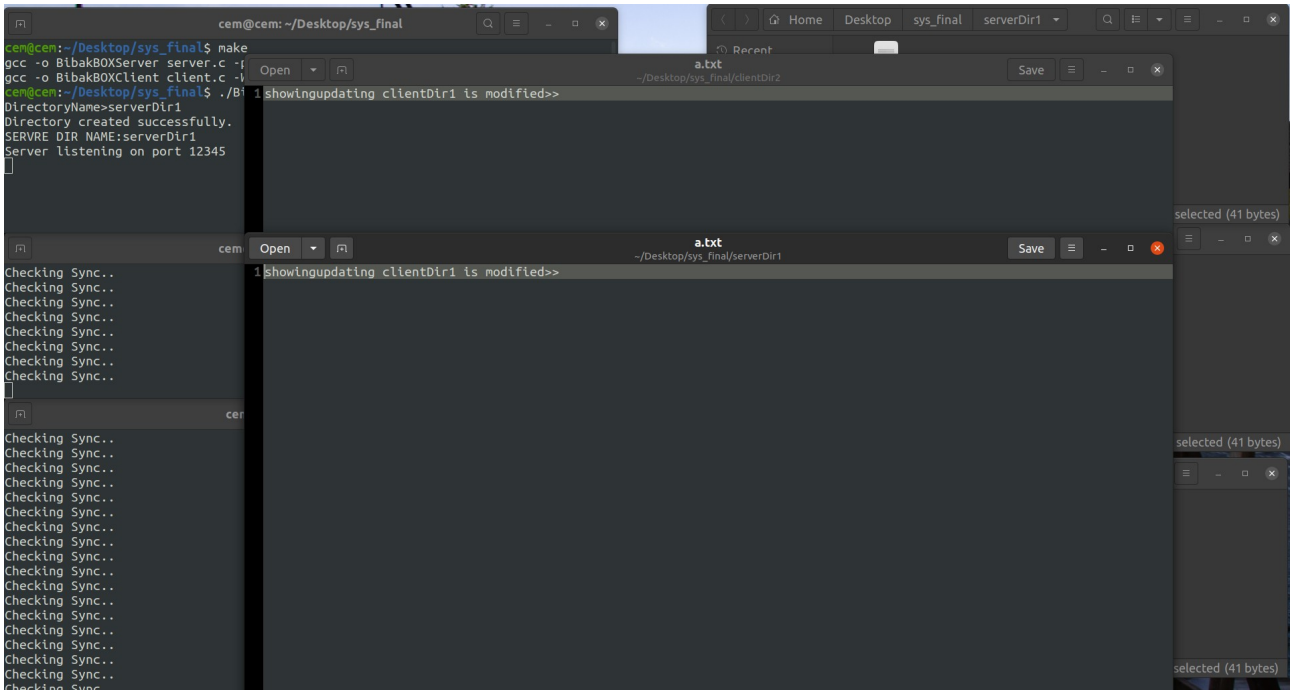
When I run my synchronization method I also write into my log file. For the naming of log files I use counter inside the critical section and name them client#number inside them I keep the information of which file is deleted and how this operation happens like from server to client server → client.



Here I open new directories for all clients and server I decide my thread pool as 2 in this example and port is 12345. so in this example it does well synchronization for all directories.

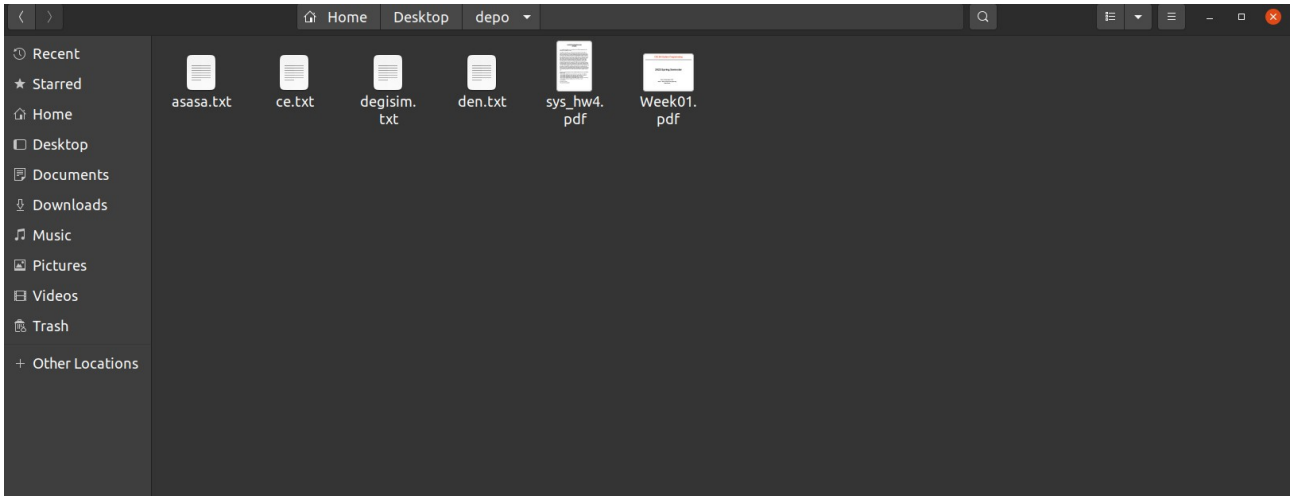


This is the content of a.txt I modified it here and show the other directors to prove my program also does the updating.

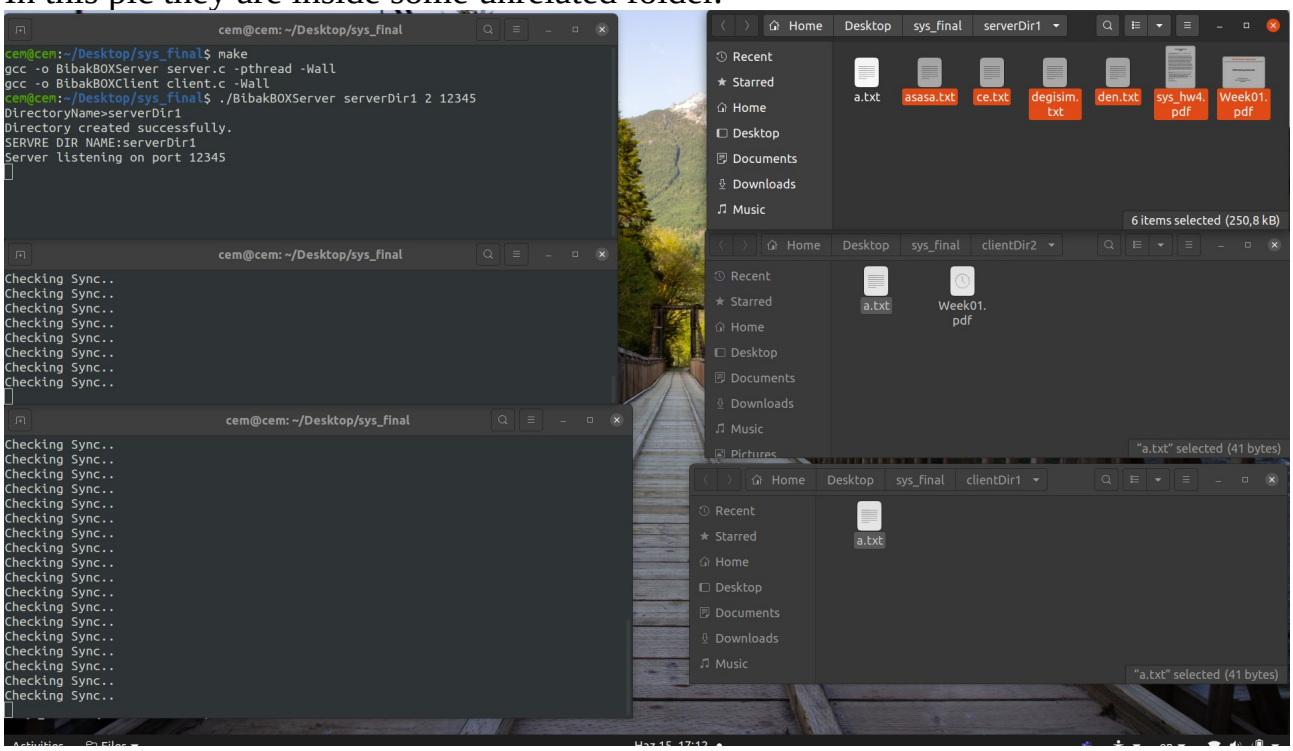


Both on serverDir1 and clientDir2 changes reflect the here so update operation is successful.

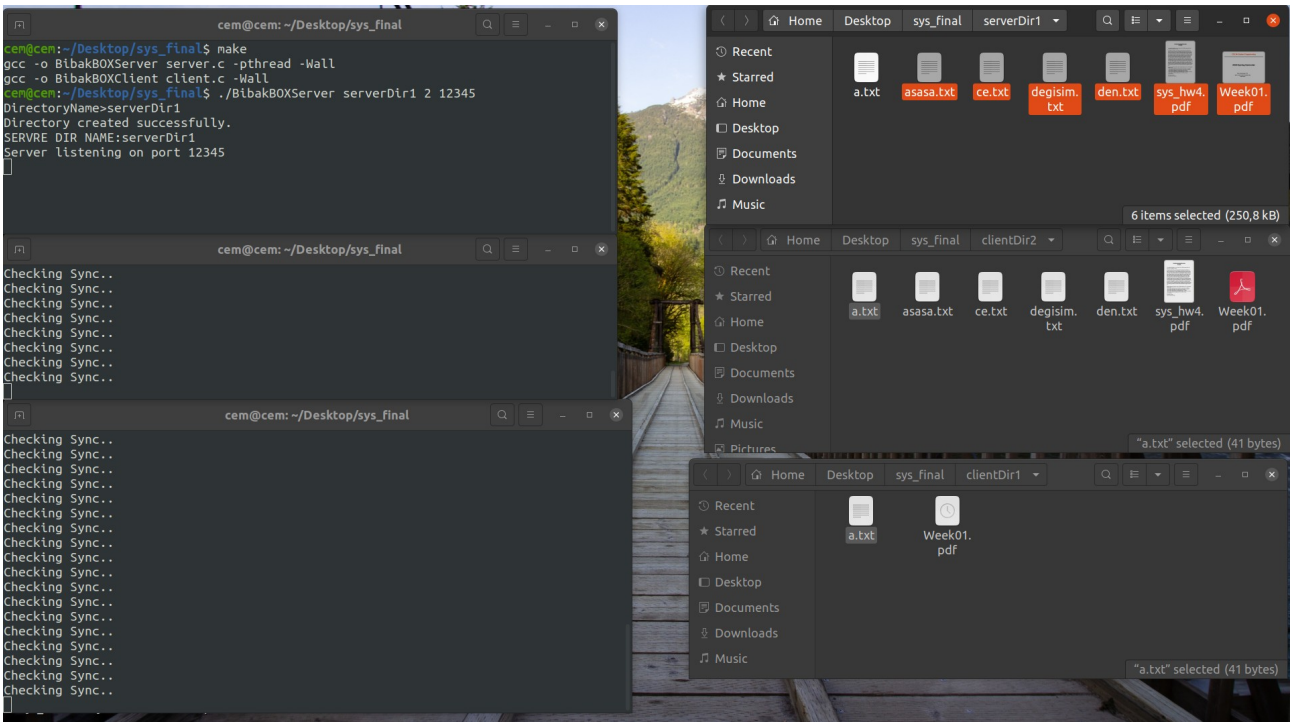
Multiple file download. Below I will show I put multiple file inside to the server and program synchronize the all directories.



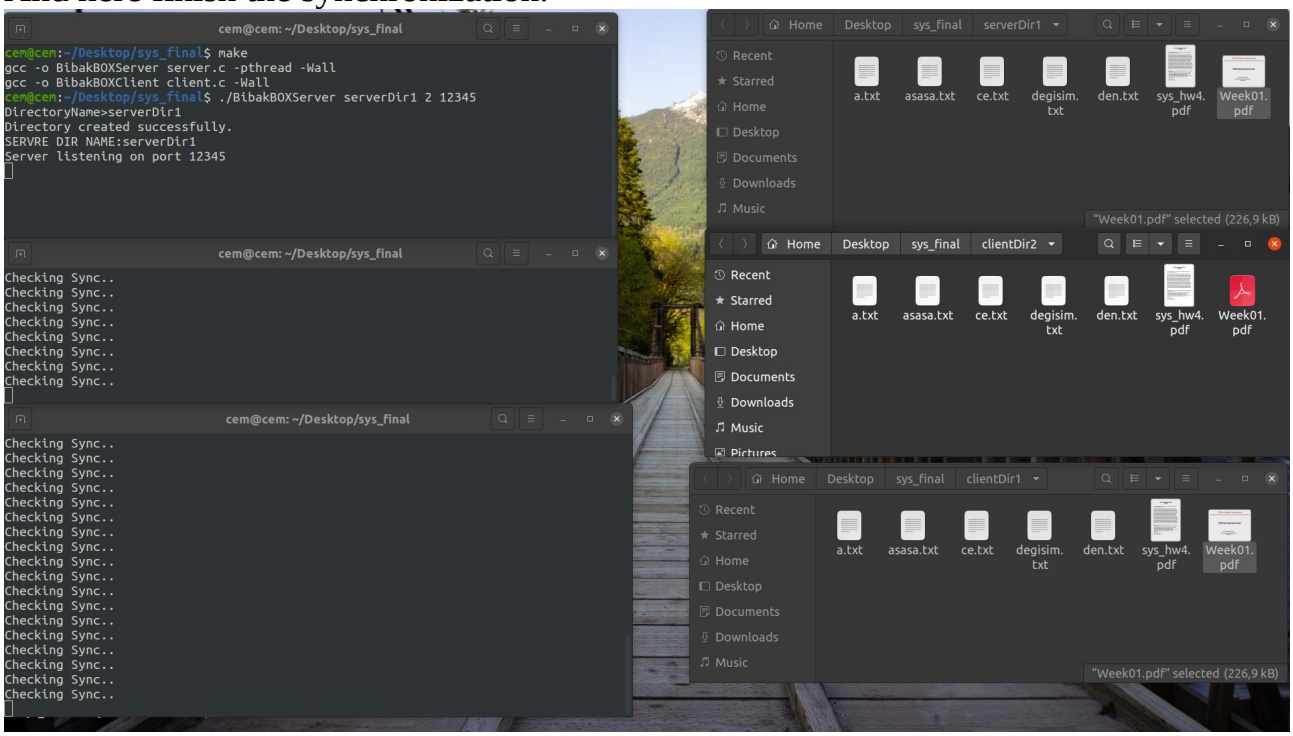
In this pic they are inside some unrelated folder.



Here it first update the clientDir2

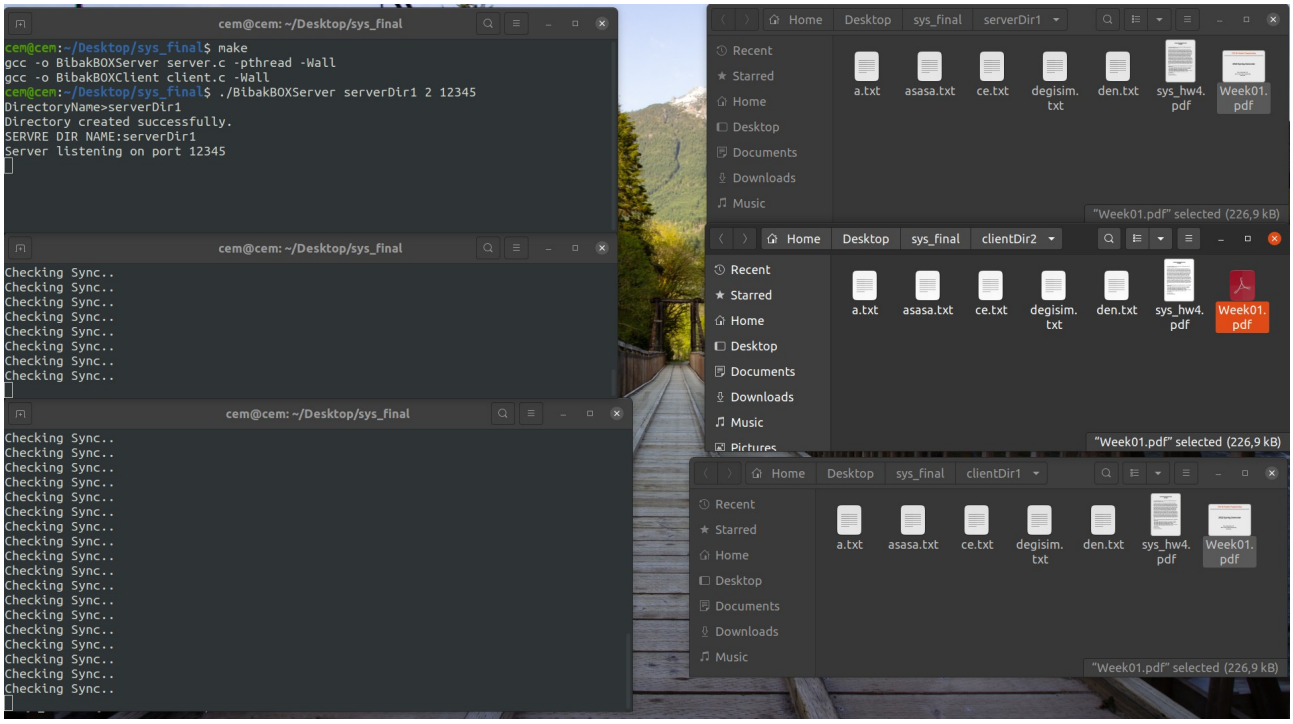


And here finish the synchronization.

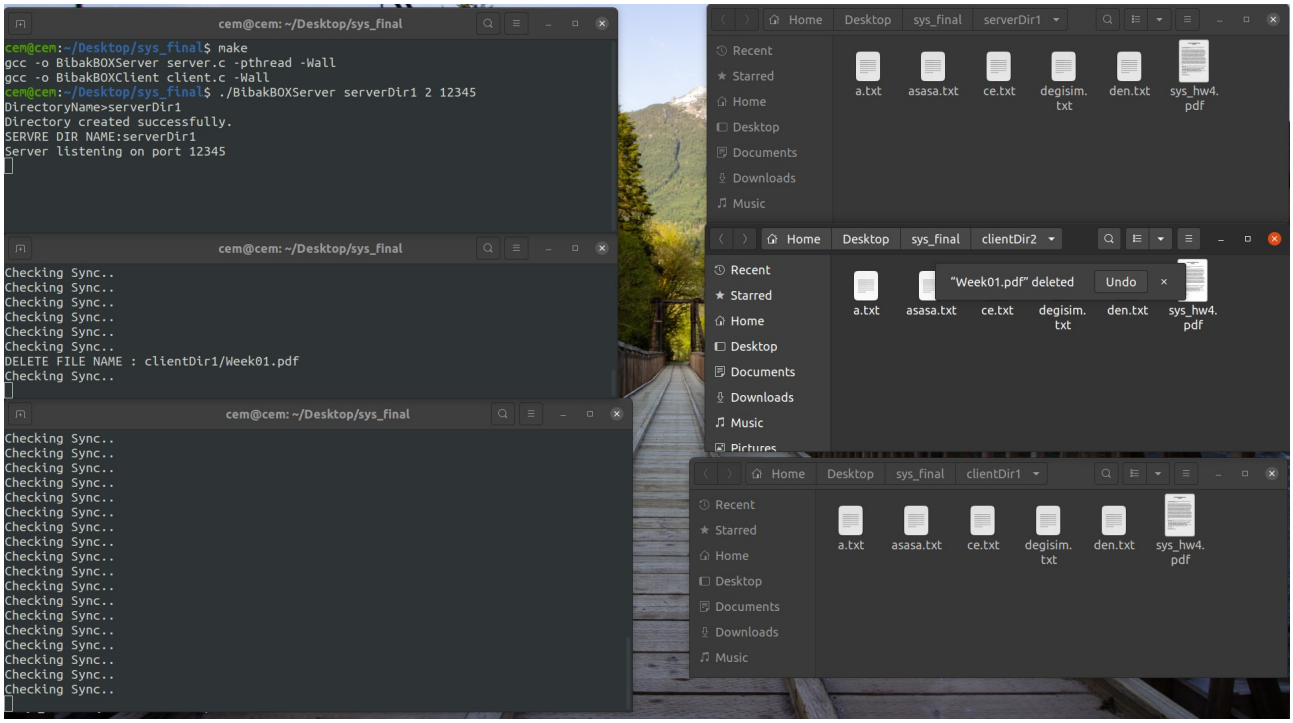


.

Here I will show deletion.



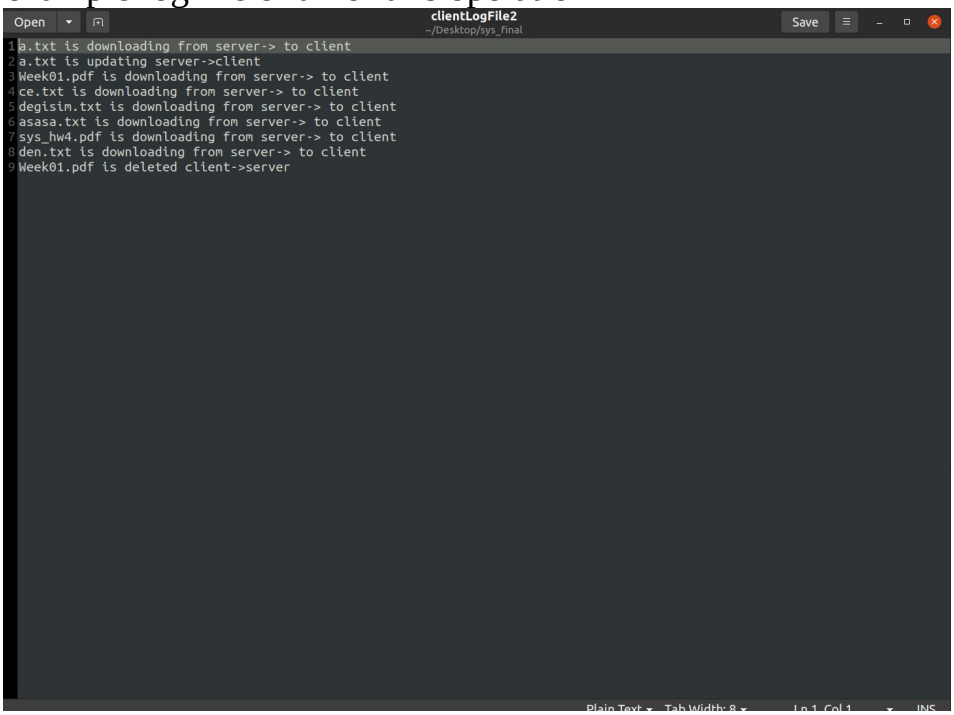
I delete week01.pdf from clientDir2 and then

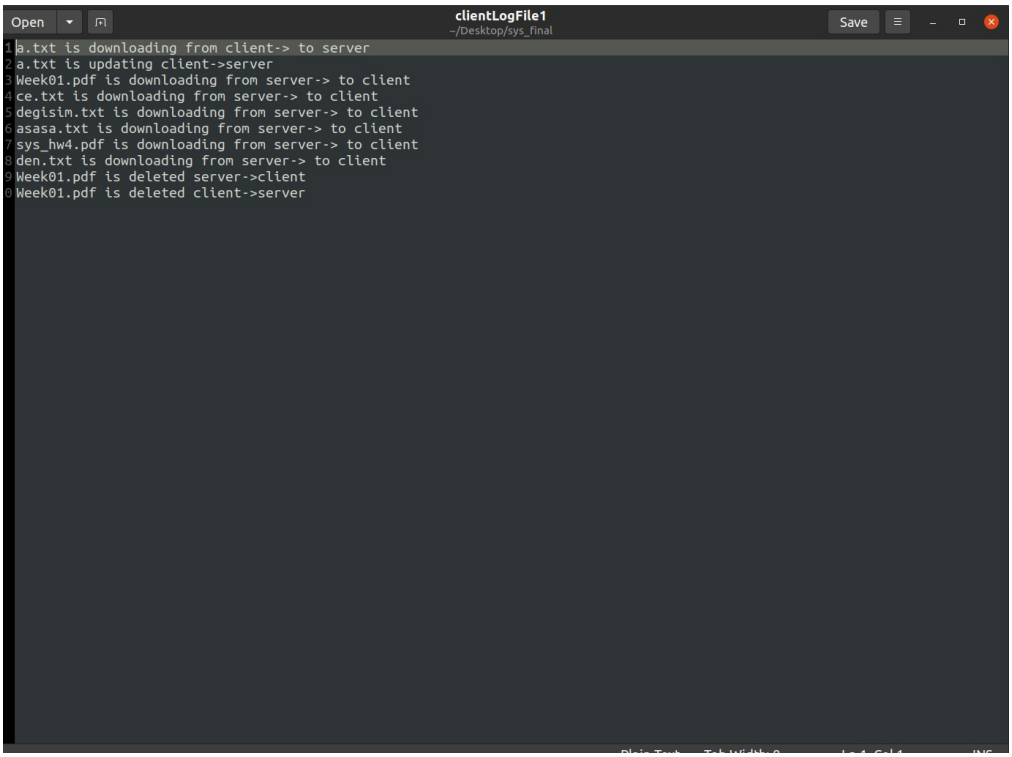


it removed from all directories.

Here is the example log file of all of this operation

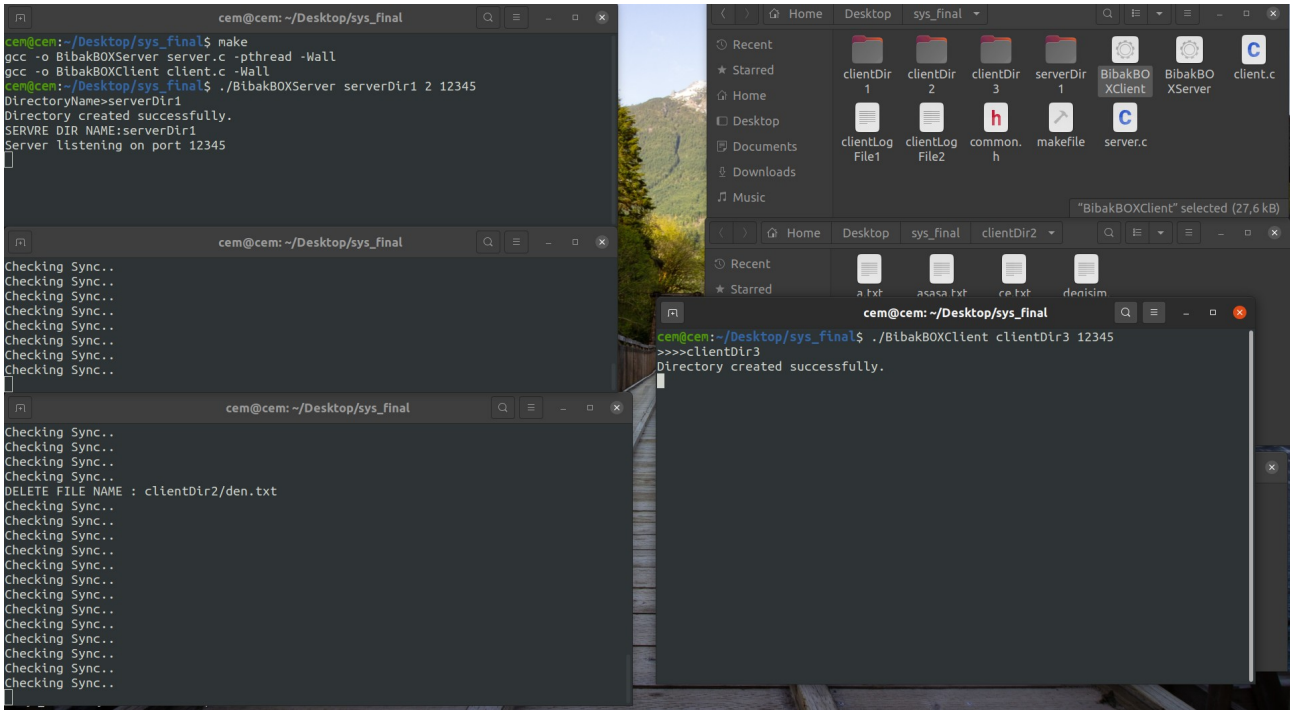
asd





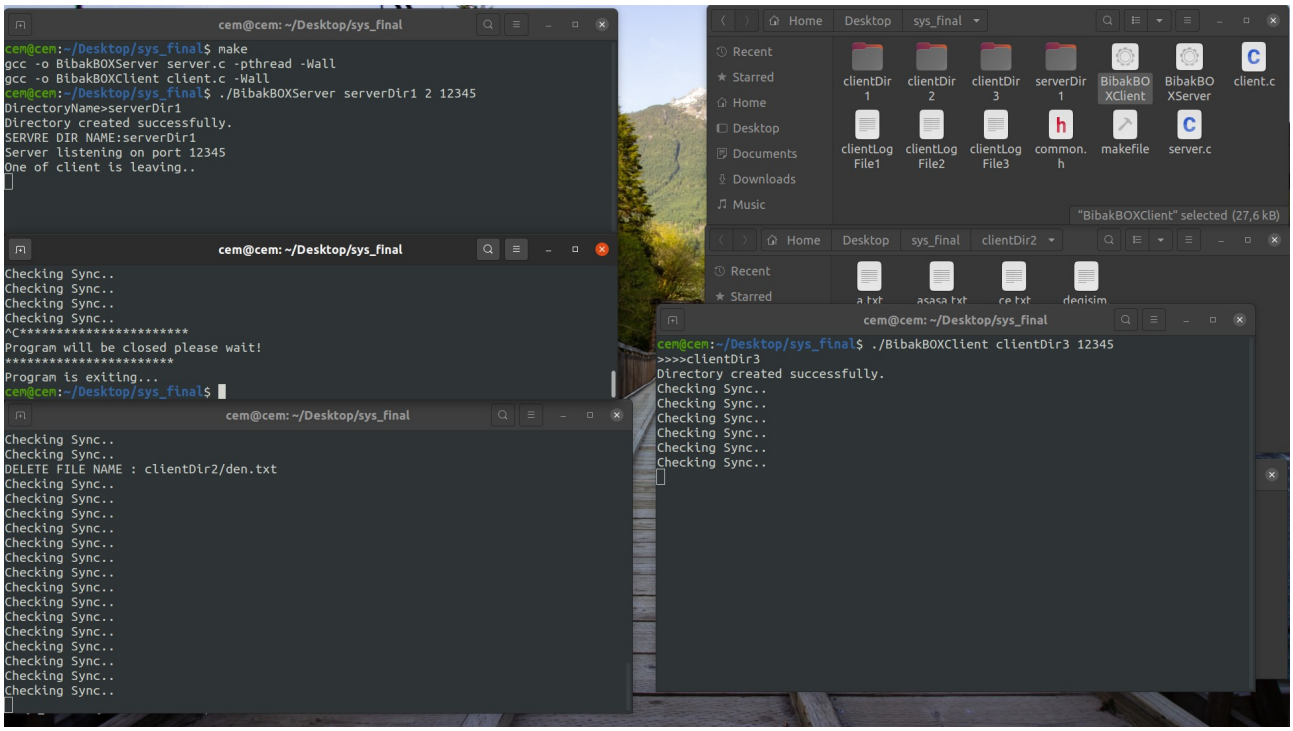
so I have been show you when all directors are empty, creation, updating, deletion scenario.

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

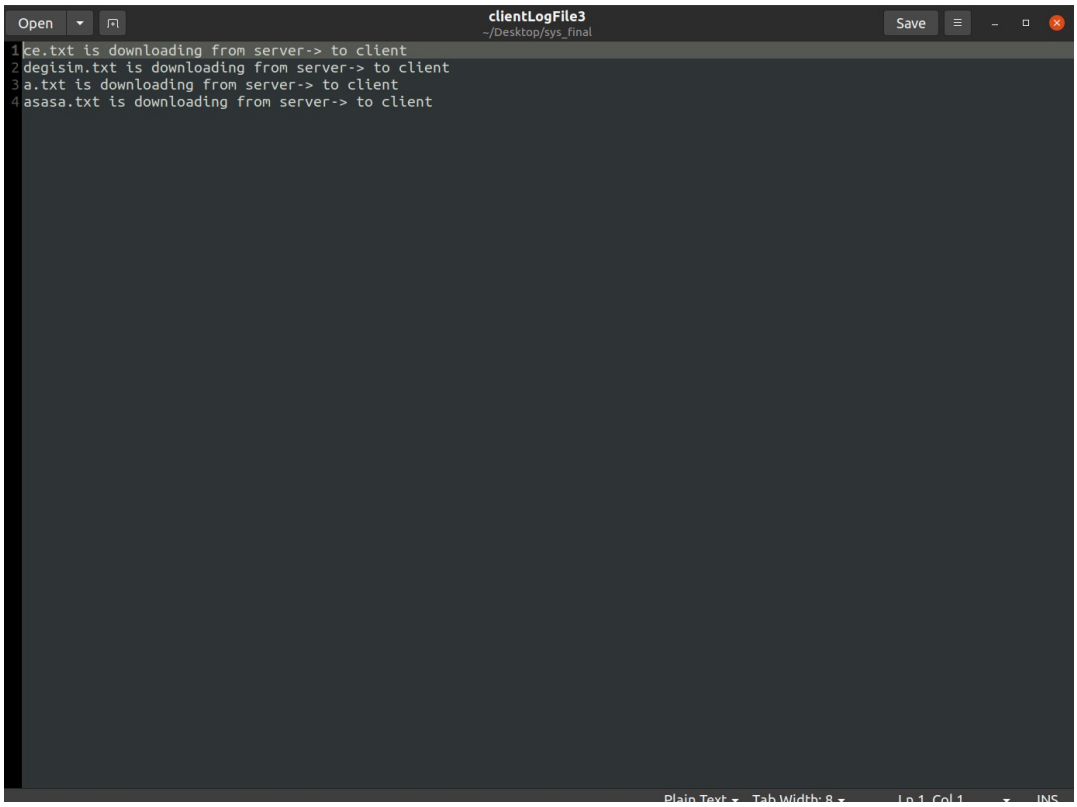


Here as you can see clientDir3 is connected but not handled by the worker thread yet you can understand from it does not print Checking Sync… and log file for the 3th client you can see it from sys_final directory which is my work space.

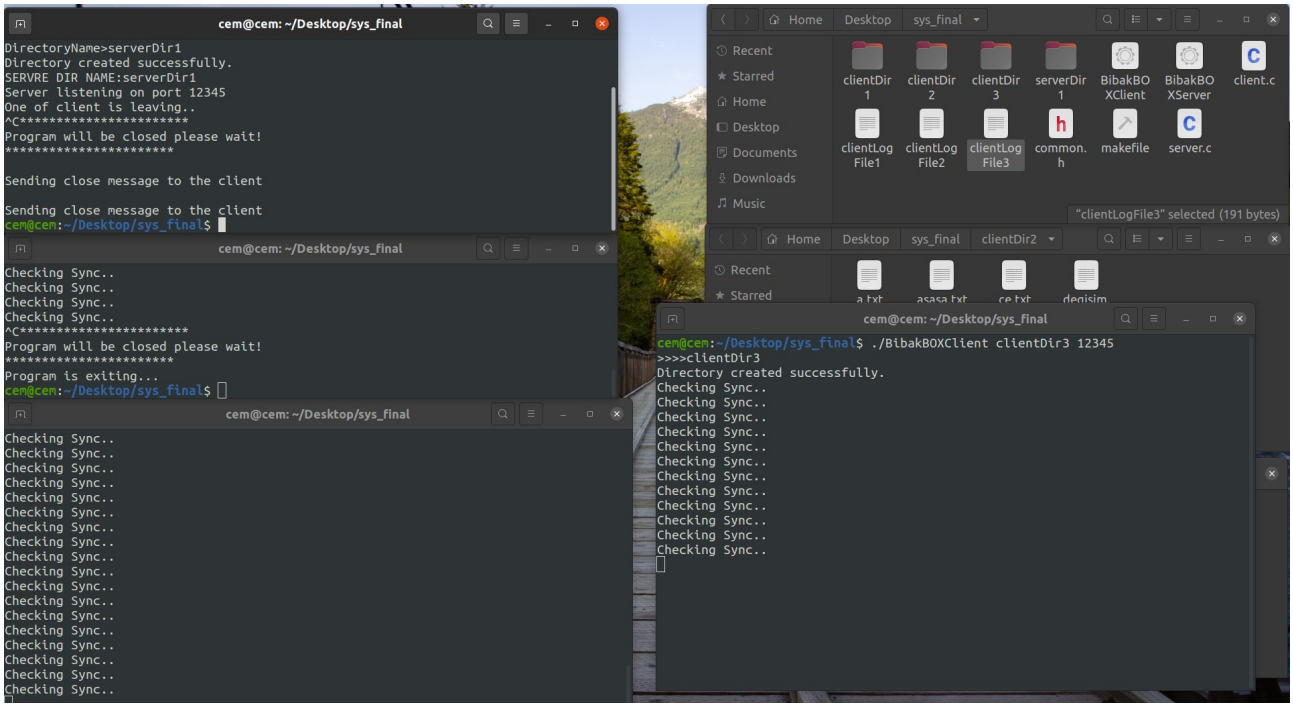
Here on the left mid you can see I send SIGINT to one of my client after it exit the other client now can access to the synchronization.

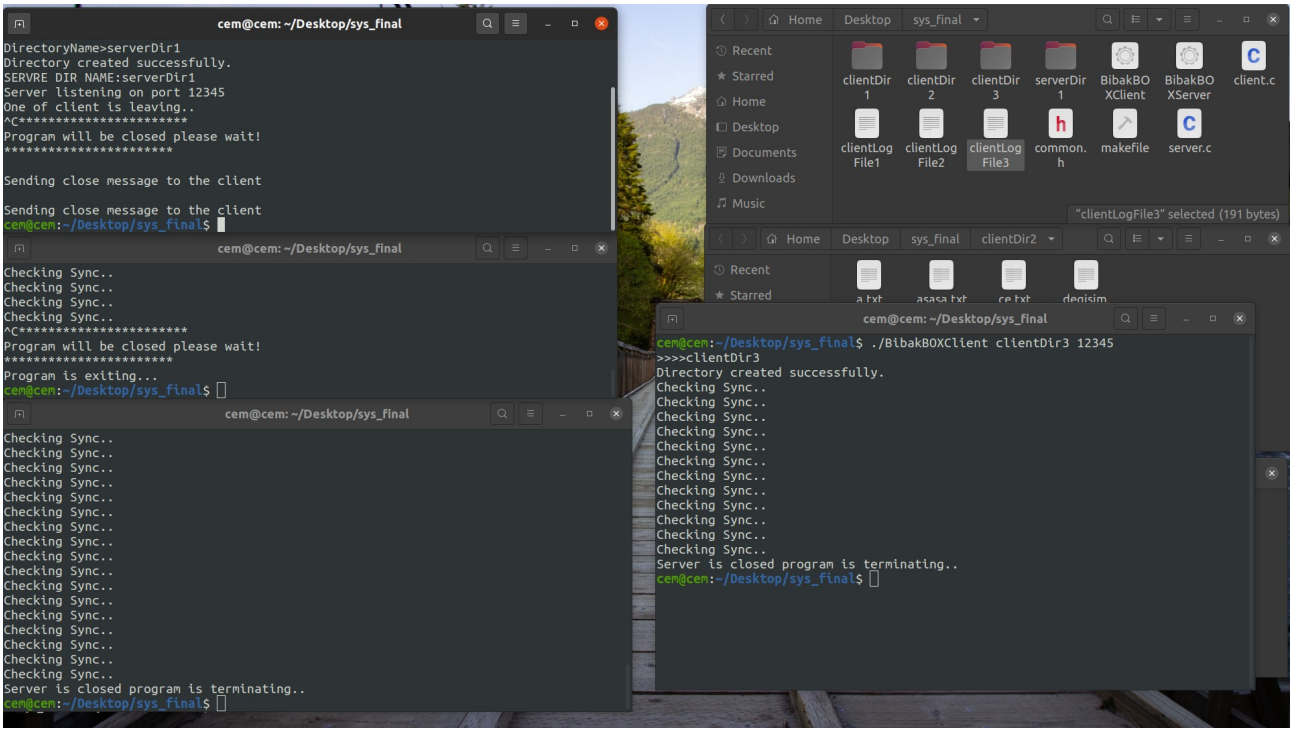


There was only 4 file and new clients directory is just created so thats the log file of the new client.



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

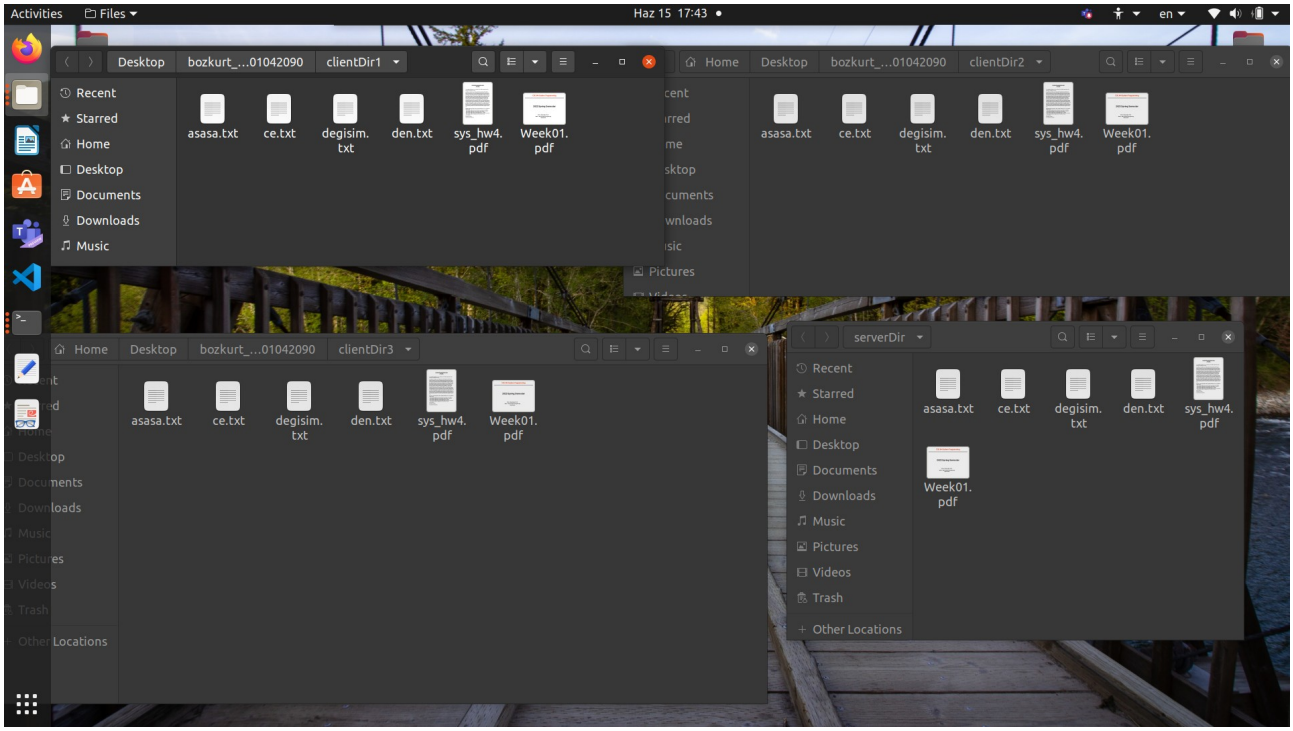




As you can see it sends all clients to message to close and before closing they complete the transition in case of they were in the middle of it.

***

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)