

GEBZE TECHNICAL UNIVERSITY
COMPUTER ENGINEERING

SYSTEM PROGRAMING(CSE 344)

FINAL PROJECT

BERKAN AKIN

171044073

Problem Defination

In this project, your task is to implement a simplified version of Dropbox. The server side should be capable of handling multiple clients simultaneously, functioning as a multi-threaded internet server. Upon establishing a connection with the server, the directories on both the server and client sides need to be synchronized. This means that any new file created, deleted, or updated on the server should reflect the same changes on the client side, and vice versa.

Unlike the official Dropbox service, your server should also maintain a logfile under the respective client's directory. This logfile should record the names and access times of created, deleted, and updated files. Additionally, it is important to handle SIGINT signal on both the server and client sides.

An example call for the server should be in the following format:

```
BibakBOXServer [directory] [threadPoolSize] [portnumber]
```

where the directory is the server's specific area for file operations, threadPoolSize is the maximum number of threads active at a time, and portnumber is the port the server will wait for connection.

An example call from the client might be in the following format:

```
BibakBOXClient [dirName] [portnumber]
```

where dirName is the name of the directory on the server side, and portnumber is the connection port of the server.

Note that the client should return with a proper message when the server is down, and the server should prompt a message when a client connection is accepted (with the address of the connection) to the screen.

Test your code with multiple (10, 20, 50) clients, reconnect to see if the server updates the client information properly. Check what happens when a new file is added, edited, or removed on the client side when the client-server connection is still active. Write a report examining at least 5 different cases. Make sure to include testing scenarios where the server and client are running on separate machines.

How to Run the Program

Client

Compile

- `gcc -o BibakBOXClient BibakBOXClient.c`

Run

- `./BibakBOXClient /home/berkan/Desktop/Final/client/test 4545`

Server

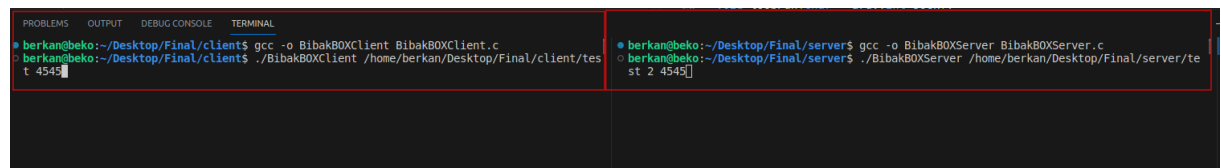
Compile

- `gcc -o BibakBOXServer BibakBOXServer.c`

Run

- `./BibakBOXServer /home/berkan/Desktop/Final/server/test 2 4545`

Example



The image shows two side-by-side terminal windows. The left window shows the compilation of BibakBOXClient and its execution. The right window shows the compilation of BibakBOXServer and its execution.

```
berkan@beko:~/Desktop/Final/clients$ gcc -o BibakBOXClient BibakBOXClient.c
berkan@beko:~/Desktop/Final/clients$ ./BibakBOXClient /home/berkan/Desktop/Final/client/test 4545

berkan@beko:~/Desktop/Final/servers$ gcc -o BibakBOXServer BibakBOXServer.c
berkan@beko:~/Desktop/Final/servers$ ./BibakBOXServer /home/berkan/Desktop/Final/server/test 2 4545
```

Requirements and Test Case

1) Copying files from client to server.

Copying the contents of the **test** directory on the client to the **test** directory on the server

First Situation in server test folder

```
client > test > folder1 > folder2 > E tmp.txt
1 istanbul
2 ankara
3 izmir

client > test > E berkan.txt
1 istanbul
2 ankara
3 izmir

berkan@berko:~/Desktop/Final/servers$ gcc -o BibakBOXServer BibakBOXServer.c
berkan@berko:~/Desktop/Final/servers$ ./server /home/berkan/Desktop/Final/server/test 2 454
clear
bind() error: Permission denied
berkan@berko:~/Desktop/Final/servers$ ./server /home/berkan/Desktop/Final/server/test 2 454
bind() error: Address already in use
berkan@berko:~/Desktop/Final/servers$ ./server /home/berkan/Desktop/Final/server/test 2 555
```

Last Situation in server test folder

```
3)Client Request:folder1
folder1: File exists.
3)Server Response: var
childDir:folder1
2)Client Request: dir
2)Server response: geldi
3)Client Request:folder2
folder2: File exists.
3)Server Response: var
childDir:folder2
2)Client Request: tmp.txt
2)Server response: geldi
tmp.txt: File exists.
2)Server Request: var
2)Client Request: return
2)Server response: bitti
*****Server to Client Control Done*****
2)Client Request: return
2)Server response: bitti
*****Server to Client Control Done*****
2)Client Request: berkan.txt
2)Server response: geldi
berkan.txt: File exists.
2)Server Request: var
Server Request: var
Client Request: return
Server response: bitti
*****Client to Server Control Done*****
*****Server to Client Control Done*****
Path:test
Client Request: folder1
Server response:var
Path:test/folder1
Client Request: folder2
Server response:var
Path:test/folder1/folder2
4)Client Request: tmp.txt
4)Server response:geldi
*****Server to Client Control Done*****
7)Client Request: return
7)Server response:bitti
*****Server to Client Control Done*****
7)Client Request: return
7)Server response:bitti
```

2) Copying files from server to client.

Copying the contents of the test directory on the server to the test directory on the client

First Situation

```
server > test > folder1 > folder2 > E tmp.txt
1 istanbul
2 ankara
3 izmir

server > test > E berkan.txt
1 gtu
2 gebze
3 istanbul
```

```
2)Client Request: return
2)Server response: bittl
*****Server to Client Control Done*****
2)Client Request: berkan.txt
2)Server response: geldi
berkan.txt: File exists.
2)Server Request: var
2)Client Request: return
2)Server response: bittl
*****Server to Client Control Done*****
^Take CTRL+C Signal
berkan@beko:~/Desktop/Final/client$ ^C
berkan@beko:~/Desktop/Final/client$ ^C
berkan@beko:~/Desktop/Final/client$
```

```
4)Client Request: tmp.txt
4)Server response: geldi
*****Server to Client Control Done*****
7)Client Request: return
7)Server response: bittl
*****Server to Client Control Done*****
7)Client Request: return
7)Server response: bittl
4)Client Request: berkan.txt
4)Server response: geldi
*****Server to Client Control Done*****
^Take CTRL+C Signal
berkan@beko:~/Desktop/Final/servers$ ^C
berkan@beko:~/Desktop/Final/servers$
```

Last Situation

```
server > test > folder1 > folder2 > E tmp.txt
1 istanbul
2 ankara
3 izmir

server > test > E berkan.txt
1 gtu
2 gebze
3 istanbul
```

```
2)Server response: bittl
*****Server to Client Control Done*****
2)Client Request: return
2)Server response: bittl
*****Server to Client Control Done*****
2)Client Request: berkan.txt
2)Server response: geldi
berkan.txt: File exists.
2)Server Request: var
2)Client Request: return
2)Server response: bittl
*****Server to Client Control Done*****
^Take CTRL+C Signal
berkan@beko:~/Desktop/Final/client$ ^C
berkan@beko:~/Desktop/Final/client$
```

```
Server response: var
Path: test/folder1/folder2
4)Client Request: tmp.txt
4)Server response: geldi
*****Server to Client Control Done*****
7)Client Request: return
7)Server response: bittl
*****Server to Client Control Done*****
7)Client Request: return
7)Server response: bittl
4)Client Request: berkan.txt
4)Server response: geldi
*****Server to Client Control Done*****
^Take CTRL+C Signal
berkan@beko:~/Desktop/Final/servers$ ^C
berkan@beko:~/Desktop/Final/servers$
```

3) Copying Files While the Program is Running

During the execution of the program, the files in the test directory where the client is running were deleted. They were then copied again with the files received from the server.

First Situation

The screenshot shows the VS Code interface with the Explorer view on the left. The 'test' directory is expanded, showing 'folder1' and 'folder2'. The 'folder2' directory is selected, and its contents are visible in the right pane: 'tmp.txt', 'berkan.txt', 'BibakBOXClient.c', 'BibakBOXServer.c', and 'client'. The 'client' file is highlighted. The Output view at the bottom shows the following log:

```
berkan.txt: File exists.
2)Server Request: var
.....
2)Client Request: return
2)Server response: bitt1
.....
*****Server to Client Control Done*****
.....
.....Client to Server Control.....
Client Request: folder1
Server response: var
.....
Client Request: folder2
Server response: var
.....
```

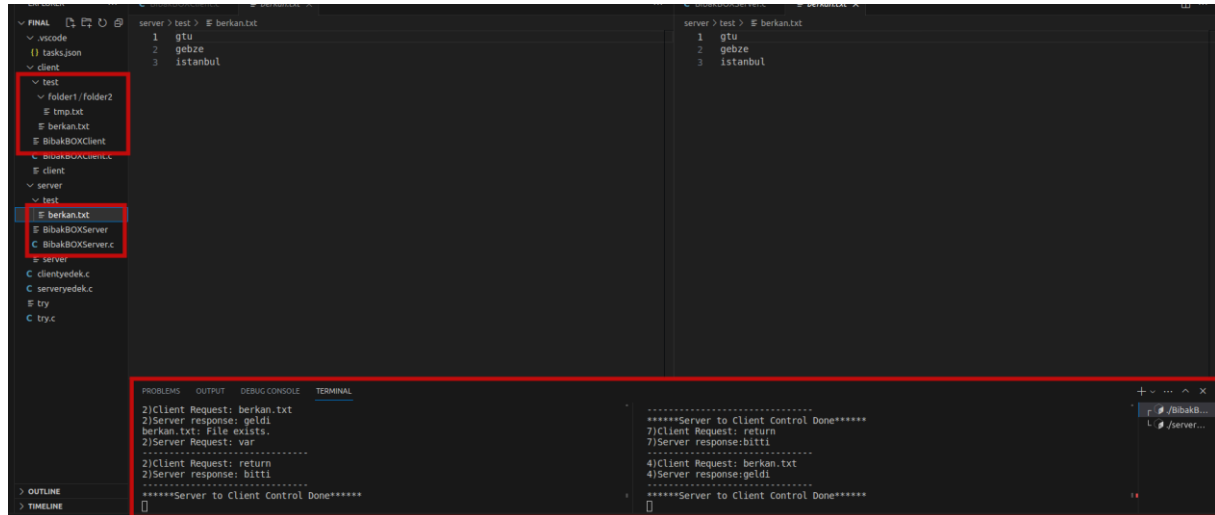
Last Situation

The screenshot shows the VS Code interface with the Explorer view on the left. The 'test' directory is expanded, showing 'folder1' and 'folder2'. The 'folder2' directory is selected, and its contents are visible in the right pane: 'tmp.txt', 'berkan.txt', 'BibakBOXClient.c', 'BibakBOXServer.c', and 'client'. The 'client' file is highlighted. The Output view at the bottom shows the following log:

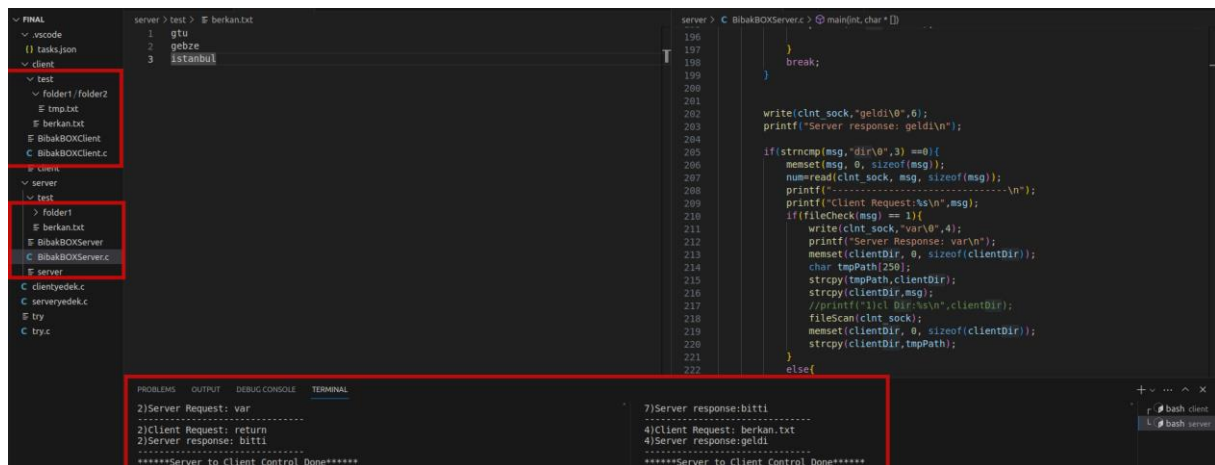
```
2)Client Request: tmp.txt
2)Server response: geldi
tmp.txt: File exists.
2)Server Request: var
.....
2)Client Request: return
2)Server response: bitt1
.....
*****Server to Client Control Done*****
.....
*****Server to Client Control Done*****
2)Client Request: return
2)Server response: bitt1
.....
*****Server to Client Control Done*****
2)Client Request: berkan.txt
2)Server response: geldi
berkan.txt: File exists.
2)Server Request: var
.....
```

The files that were deleted from the server's test directory are being placed back in the same location.

First Situation



Last Situation



4) Multiple programs running at the same time.

```
berkan@beko: ~/Desktop/client2
2)Client Request: return
2)Server response: bittl
*****Server to Client Control Done*****
-----Client to Server Control-----
Client Request: ahmet
Server response:var
Client Request: yasin.txt
Server response:geldl

berkan@beko: ~/Desktop...
*****Server to Client Control Done*****
2)Client Request: return
2)Server response: bittl
*****Server to Client Control Done*****
2)Client Request: return
2)Server response: bittl
*****Server to Client Control Done*****
strcm

berkan@beko: ~/Desktop/client2
printf(*****Server to Client Control Done*****
2)Client Request: return
2)Server response: bittl
*****Server to Client Control Done*****
listfil2)Client Request: berkan.txt
write(s2)Server response: geldl
memset(berkan.txt: File exists.
read(so2)Server Request: var
//print
//print 2)Client Request: return
//print 2)Server response: bittl
se {
//char *****Server to Client Control Done*****
write(sock, &EXIT_FAILURE, sizeof(EXIT_FAILURE));
memset(message, 0, sizeof(message));
read(sock, message, sizeof(message));
printf("-----\n");
printf("Client Request: %s\n", extract_filename(newPath));
printf("Server response:%s\n", message);
if(strlen(message, "geldl")>0){
memset(message, 0, sizeof(message));
read(sock, message, sizeof(message));
}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2)Client Request: berkan.txt
2)Server response: geldl
berkan.txt: File exists.
2)Server Request: var
-----
2)Client Request: return
2)Server response: bittl
*****Server to Client Control Done*****
4)Client Request: berkan.txt
4)Server response:geldl
*****Server to Client Control Done*****
7)Client Request: return
7)Server response:bittl
4)Client Request: bittl
4)Server response:geldl
Ln 423, Col 6 Spaces: 4 UTF-8 LF ( ) C Linux JP
```

5) Signal Handling

```
berkan.txt: File exists.
2)Server Request: var
-----
2)Client Request: return
2)Server response: bittl
*****Server to Client Control Done*****
^CTake CTRL+C Signal
berkan@beko:~/Desktop/Final/clients$ ^C
berkan@beko:~/Desktop/Final/clients$

4)Client Request: berkan.txt
4)Server response:geldl
*****Server to Client Control Done*****
^CTake CTRL+C Signal
berkan@beko:~/Desktop/Final/servers$ ^C
berkan@beko:~/Desktop/Final/servers$ ^C
berkan@beko:~/Desktop/Final/servers$ ^C
berkan@beko:~/Desktop/Final/servers$
Ln 423, Col 6 Spaces: 4 UTF-8 LF ( ) C Linux JP
```

Code Snippet

```
void signal_handler(int signal_number) {
    if (signal_number == SIGINT) {
        printf("Take CTRL+C Signal \n");
        exit(0);
    }
}

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


6) Running 10, 20 and 50 programs simultaneously.

10, 20 and 50 programs were run at the same time and the server was able to respond to this client.

```
client > $ script.sh
1  #!/bin/bash
2
3  for ((i=0; i<10; i++))
4  do
5      ./BibakBOXClient /home/berkan/Desktop/Final/client/test 4547 &
6  done
7
8  wait
9
```

```
nt > $ script.sh
1  #!/bin/bash
2
3  for ((i=0; i<20; i++))
4  do
5      ./BibakBOXClient /home/berkan/Desktop/Final/client/test 4547 &
6  done
7
8  wait
9
```

```
ent > $ script.sh
1  #!/bin/bash
2
3  for ((i=0; i<50; i++))
4  do
5      ./BibakBOXClient /home/berkan/Desktop/Final/client/test 4547 &
6  done
7
8  wait
9
```

It is seen that it works normally in 50 users.

```
client > $ script.sh
1 #!/bin/bash
2
3 for ((i=0; i<50; i++))
4 do
5     ./BibakBOXClient /home/berkan/Desktop/Final/client/test 4547 &
6 done
7
8 wait
9
```

```
client > C BibakBOXClient.c >
193 printf("Client Request: %s\n",extract_filename(newPath));
194 printf("Server response:%s\n",message);
195
196
197
198
199
200
201
202 listFilesAndDirectories(newPath,sock);
203 write(sock,"return0",7);
204 memset(message, 0, sizeof(message));
205 read(sock,message,sizeof(message));
206 //printf(".....\n");
207 //printf("7)Client Request: return\n");
208 //printf("7)Server response:%s\n",message);
209 } else {
210 //char * tmpStr = ;
211 write(sock,extract_filename(newPath),strlen(extract_filename(newPath))+1);
212
213 memset(message, 0, sizeof(message));
214 read(sock,message,sizeof(message));
215 printf(".....\n");
216 printf("Client Request: %s\n",extract_filename(newPath));
217 printf("Server response:%s\n",message);
218 if(strncmp(message,"geldi\0",5)==0){
219     memset(message, 0, sizeof(message));
220     read(sock,message,sizeof(message));
221     //printf("Server response:%s\n",message);
222 }
```

```
2)Client Request: berkan.txt
2)Server response: geldi
berkan.txt: File exists.
2)Server Request: var
.....
2)Client Request: return
2)Server response: bitti
.....
*****Server to Client Control Done*****
```

```
*****Server to Client Control Done*****
7)Client Request: return
7)Server response:bitti
.....
4)Client Request: berkan.txt
4)Server response:geldi
.....
*****Server to Client Control Done*****
```

Note

- update and delete operations are not done in the program