Manual Instruction

Linlin Chen

A20348195

Lchen96@hawk.iit.edu

Here I will introduce how to run my program step by step. Most of them have been introduced in output files, so if you directly read the output file, maybe you already know how to run my program.

1. Compiling

To compile my program, please follow the following instructions (I already compile it, but if you want feel free to re-compile, but since there are several packages, so please compile it following this instruction)
First, compile the Utilities package, which is all the utilities used in my program.
The compiling instruction is:

```
javac -cp commons-codec-1.10.jar:commons-io-2.5.jar:. Utilities/*.java
```

```
lchen@lchens-MacBook-Pro ~/Google Drive/Graduate_Class/550-AdvancedOperatingSystem/Project/P1/LinlinChe
n/Code javac -cp commons-codec-1.10.jar:commons-io-2.5.jar:. Utilities/*.java
lchen@lchens-MacBook-Pro ~/Google Drive/Graduate_Class/550-AdvancedOperatingSystem/Project/P1/LinlinChe
n/Code
```

Then compile the IndexServer package:

```
javac -cp commons-codec-1.10.jar:commons-io-2.5.jar:. IndexServer/IndexingServer.java
```

```
lchen@lchens-MacBook-Pro ~/Google Drive/Graduate_Class/550-AdvancedOperatingSystem/Project/P1/LinlinChe
n/Code javac -cp commons-codec-1.10.jar:commons-io-2.5.jar:. IndexServer/IndexingServer.java
lchen@lchens-MacBook-Pro ~/Google Drive/Graduate_Class/550-AdvancedOperatingSystem/Project/P1/LinlinChe
```

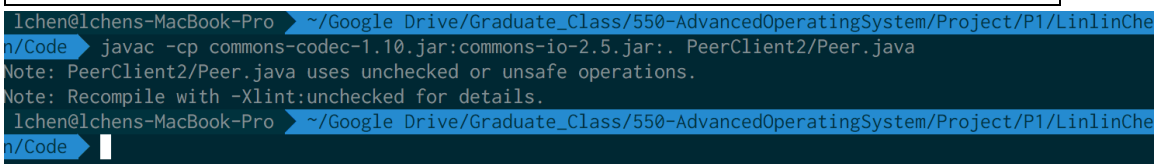
Then compile the PeerClient1 package:

```
javac -cp commons-codec-1.10.jar:commons-io-2.5.jar:. PeerClient1/Peer.java
```

```
lchen@lchens-MacBook-Pro ~/Google Drive/Graduate_Class/550-AdvancedOperatingSystem/Project/P1/LinlinChe
n/Code javac -cp commons-codec-1.10.jar:commons-io-2.5.jar:. PeerClient1/Peer.java
Note: PeerClient1/Peer.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
lchen@lchens-MacBook-Pro ~/Google Drive/Graduate_Class/550-AdvancedOperatingSystem/Project/P1/LinlinChe
n/Code
```

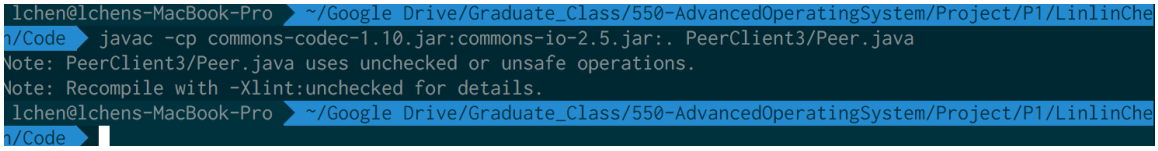
Then compile PeerClient2 package:

```
javac -cp commons-codec-1.10.jar:commons-io-2.5.jar:. PeerClient2/Peer.java
```



Lastly compile PeerClient3 package:

```
javac -cp commons-codec-1.10.jar:commons-io-2.5.jar:. PeerClient3/Peer.java
```



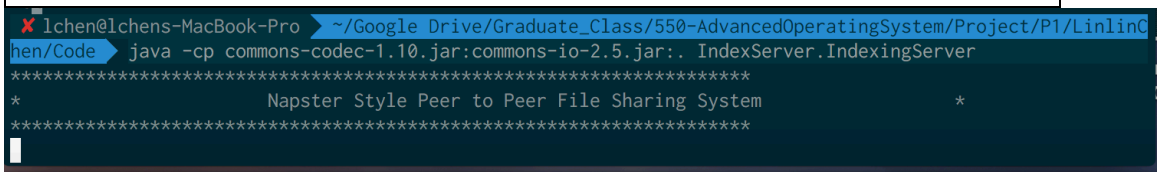
Now we finish all the compiling work.

2. To Start-up

To run the index server and each client, please input the following instructions one by one.

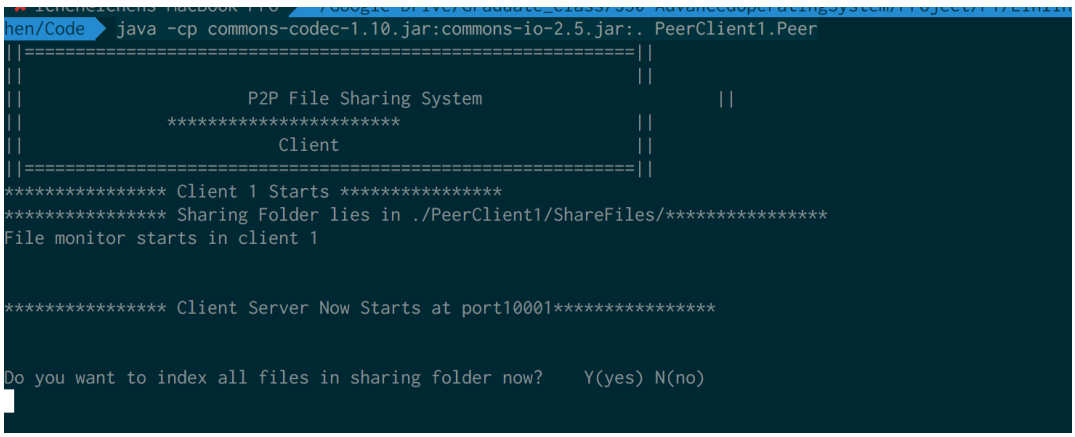
Start the index server:

```
java -cp commons-codec-1.10.jar:commons-io-2.5.jar:. IndexServer.IndexingServer
```



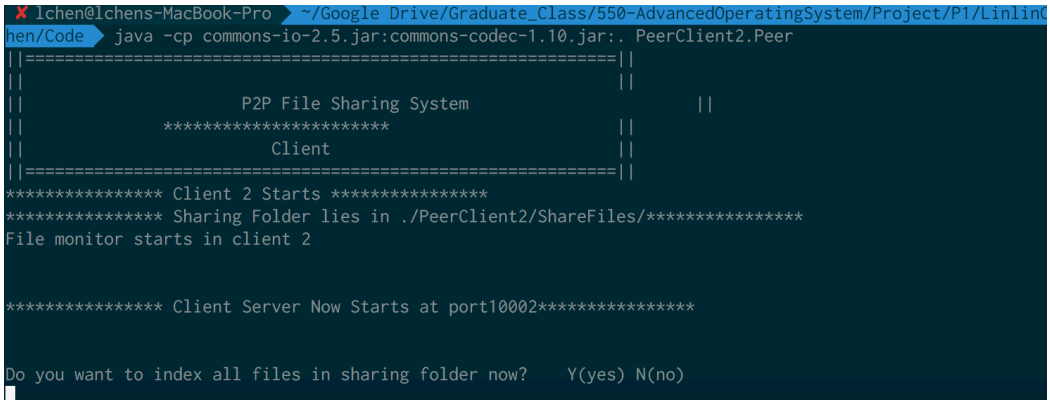
Start the client 1:

```
java -cp commons-codec-1.10.jar:commons-io-2.5.jar:. PeerClient1.Peer
```



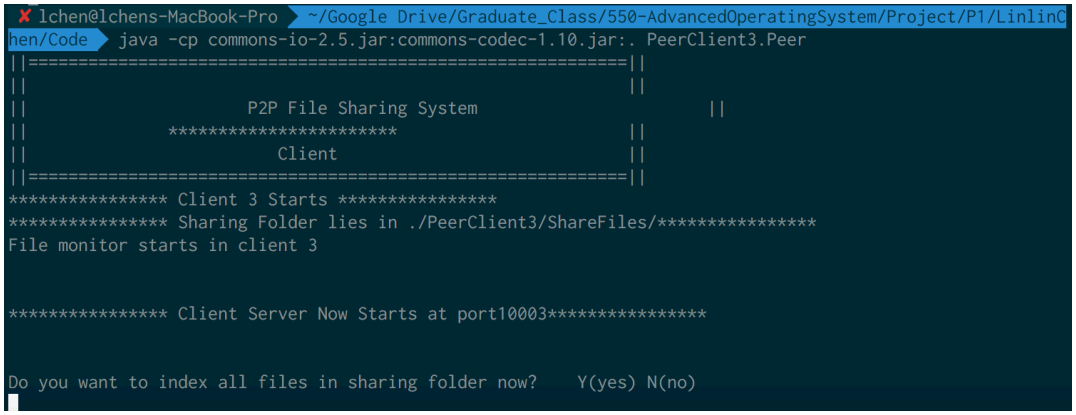
Start the client 2:

```
java -cp commons-codec-1.10.jar:commons-io-2.5.jar:. PeerClient2.Peer
```



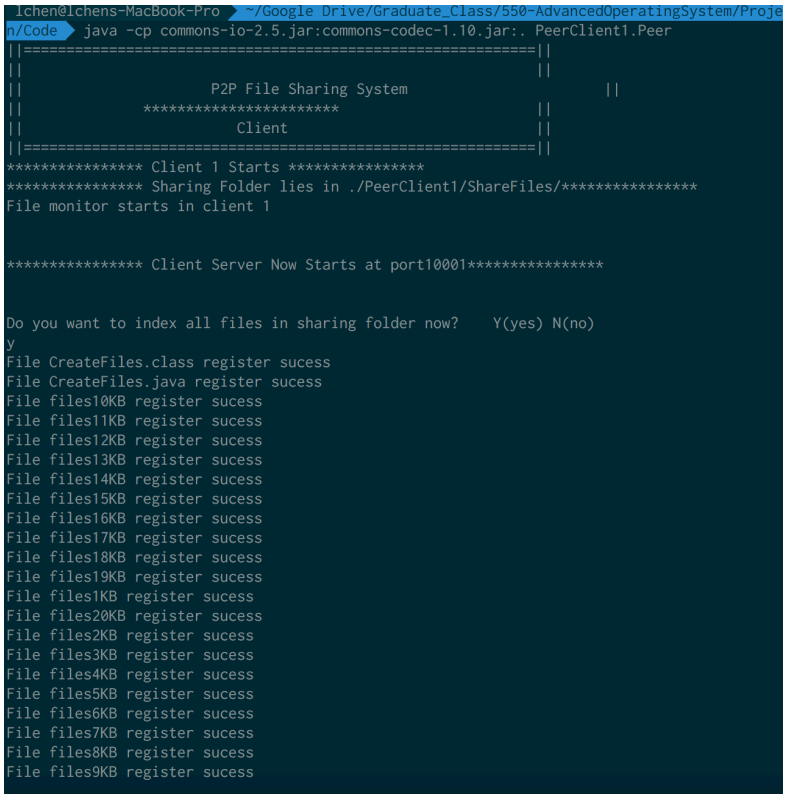
Start the client 3:

```
java -cp commons-codec-1.10.jar:commons-io-2.5.jar:. PeerClient3.Peer
```



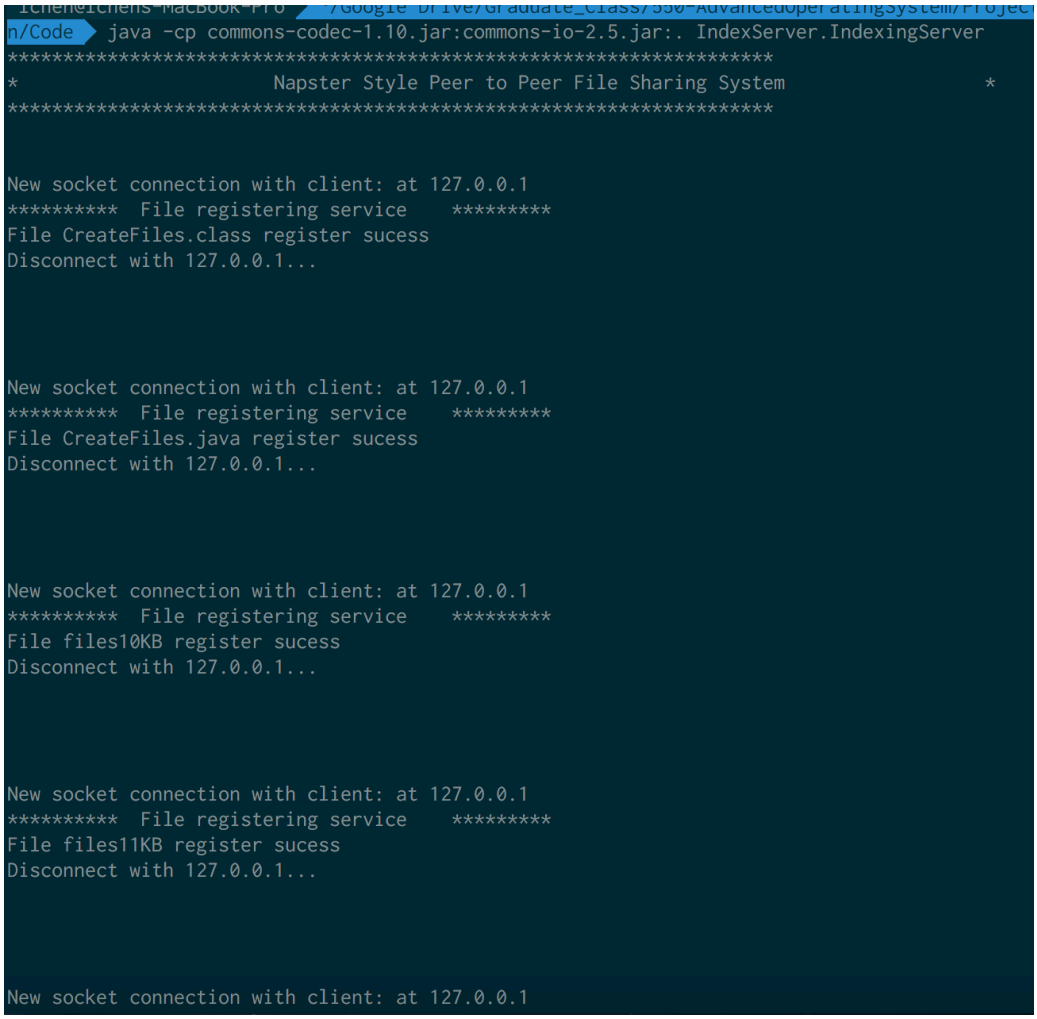
When the program starts, for the client side, there would be a question asking whether you would like to register all files in the current sharing folder. If you choose "Y", all files in sharing folder will be registered to the server.

Here is a start up interface.

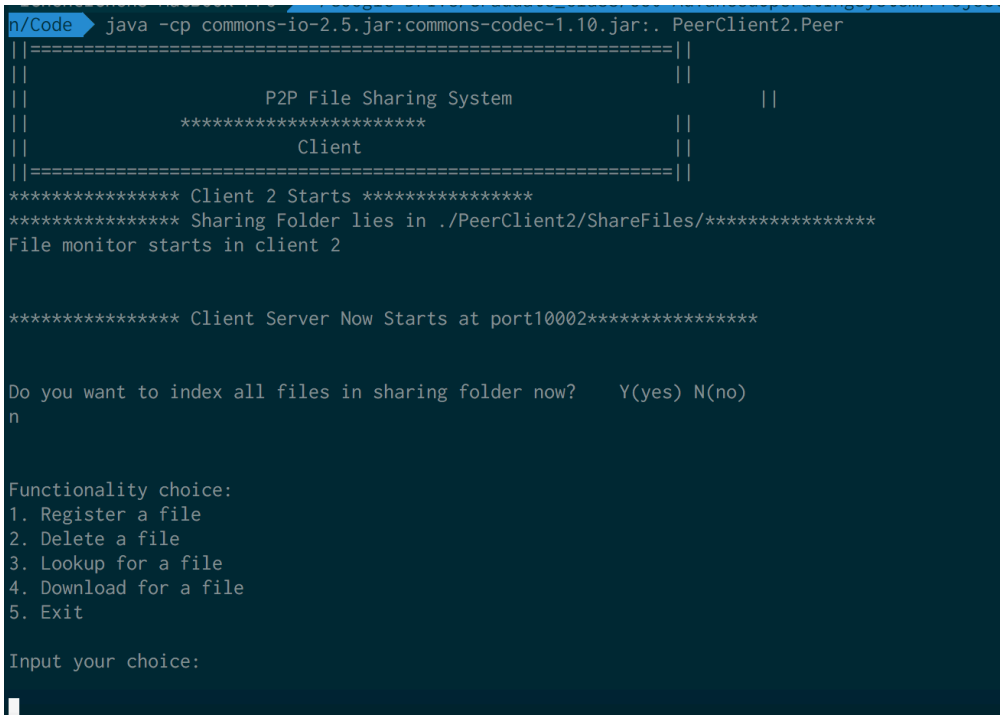


This shows when client 1 starts and the user choose to index all files in sharing folder.

Correspondingly, the server side will show all the incoming connections and the service requested. Here is a picture.



While in client 2, it chooses not to index the files in sharing folder now, it will be look like:

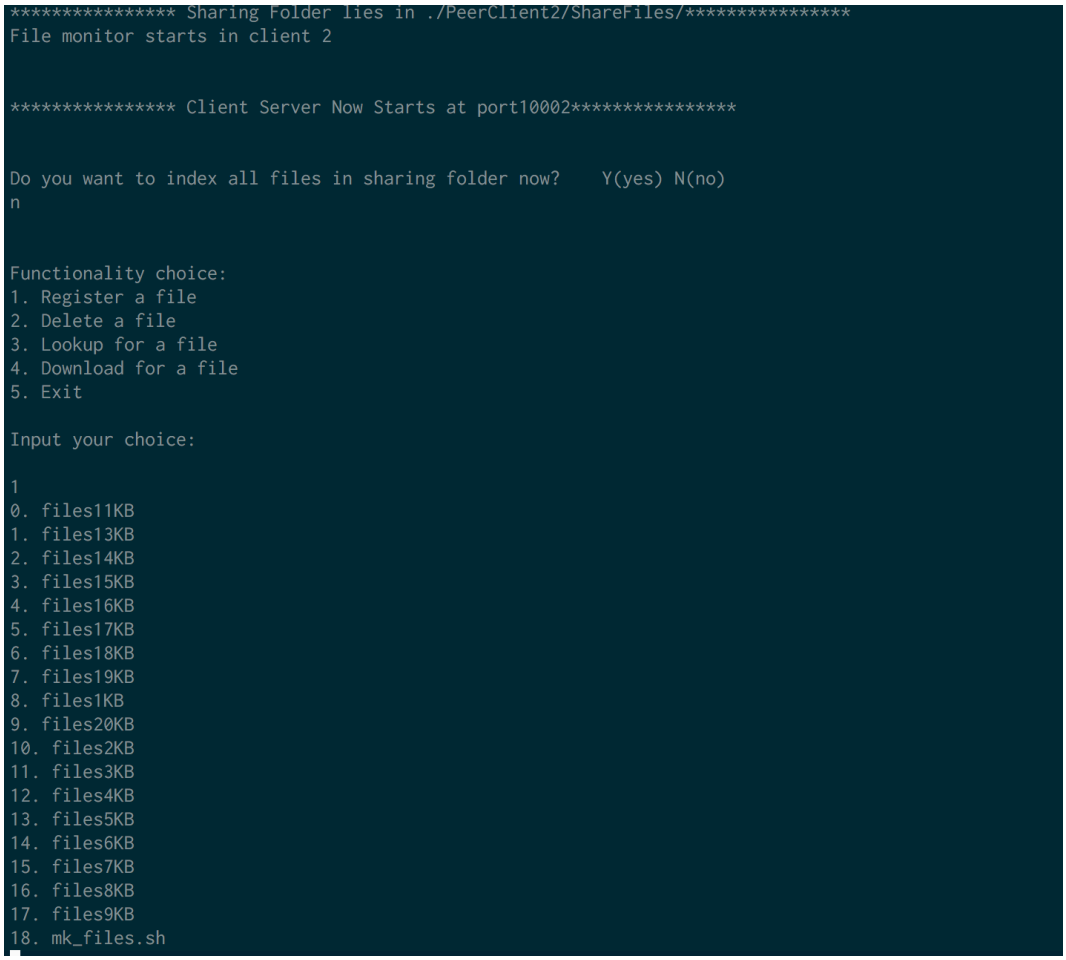


3. Register a file

As mentioned in output file, there are two ways to register a file, directly register file in terminal or create a new file in sharing directory and it will automatically register it.
Here I show how a file can be registered in two different ways.

1) Register a file in terminal.

Take client 2 as an example:



When we choose option 1. Register a file, it will list all files in the current sharing folder. You can directly input the file order number to choose which file you want to register. Here I input 2.

```
Input your choice:

1
0. CreateFiles.class
1. CreateFiles.java
2. files10KB
3. files11KB
4. files12KB
5. files13KB
6. files14KB
7. files15KB
8. files16KB
9. files17KB
10. files18KB
11. files19KB
12. files1KB
13. files20KB
14. files2KB
15. files3KB
16. files4KB
17. files5KB
18. files6KB
19. files7KB
20. files8KB
21. files9KB
2
File files10KB register sucess


Functionality choice:
1. Register a file
2. Delete a file
3. Lookup for a file
4. Download for a file
5. Exit

Input your choice:
```

The server will also give a notify to show whether registration succeed.

```
New socket connection with client: at 127.0.0.1
**********  File registering service    *********
File files10KB register sucess
Disconnect with 127.0.0.1...
```

Please notice that here, the files10KB is exactly the same with files10KB in client 1. Later we will show this to prove our system can distinguish two files relying on the MD5 value instead of file name.

2) Create a new file in folder.

We can also create a file in the sharing folder and the client will automatically notify the server to register this file. I randomly copy a file into client 2's sharing folder. The file name is PA1(1).pdf. I did nothing in the terminal, and just copy a file into the sharing folder. Lets' see what happened in the terminal.

For client 2, it will give a message showing PA1(1).pdf is registered successfully. (I give no instructions in terminal)

```
Functionality choice:
1. Register a file
2. Delete a file
3. Lookup for a file
4. Download for a file
5. Exit

Input your choice:

File PA1(1).pdfregister sucess
```

In the server side, it will also show the successful message.

```
New socket connection with client: at 127.0.0.1
**********  File registering service    *********
File files11KB register sucess
Disconnect with 127.0.0.1...




New socket connection with client: at 127.0.0.1
**********  File registering service    *********
File PA1(1).pdf register sucess
Disconnect with 127.0.0.1...
```

4. Modify a file

For my project, I realize the MD5 calculation to distinguish two files, not simply relying on the filename. So once a file is modified, it will firstly notify the server to unregister the old one, and register the new file with unique MD5 value.

If we modify the file files10KB in client 2's sharing folder, as we just registered it in the server. I opened it and add some new contents in the file. Now let's see what happened in client 2.

```
Functionality choice:
1. Register a file
2. Delete a file
3. Lookup for a file
4. Download for a file
5. Exit

Input your choice:

File PA1(1).pdf register sucess
File files10KB unregister sucess
File files10KB register sucess
```

Also for the server, it will show an unregister request and register request sent from client 2.

```
New socket connection with client: at 127.0.0.1
**********  File registering service    *********
File PA1(1).pdf register sucess
Disconnect with 127.0.0.1...



New socket connection with client: at 127.0.0.1
**********  File unregistering service  *********
File files10KB unregister sucess
Disconnect with 127.0.0.1...



New socket connection with client: at 127.0.0.1
**********  File registering service    *********
File files10KB register sucess
Disconnect with 127.0.0.1...
```

In the look up stage, we will show that server indexed these two files with same file name files10KB, but it can distinguish them based on the MD5 value.

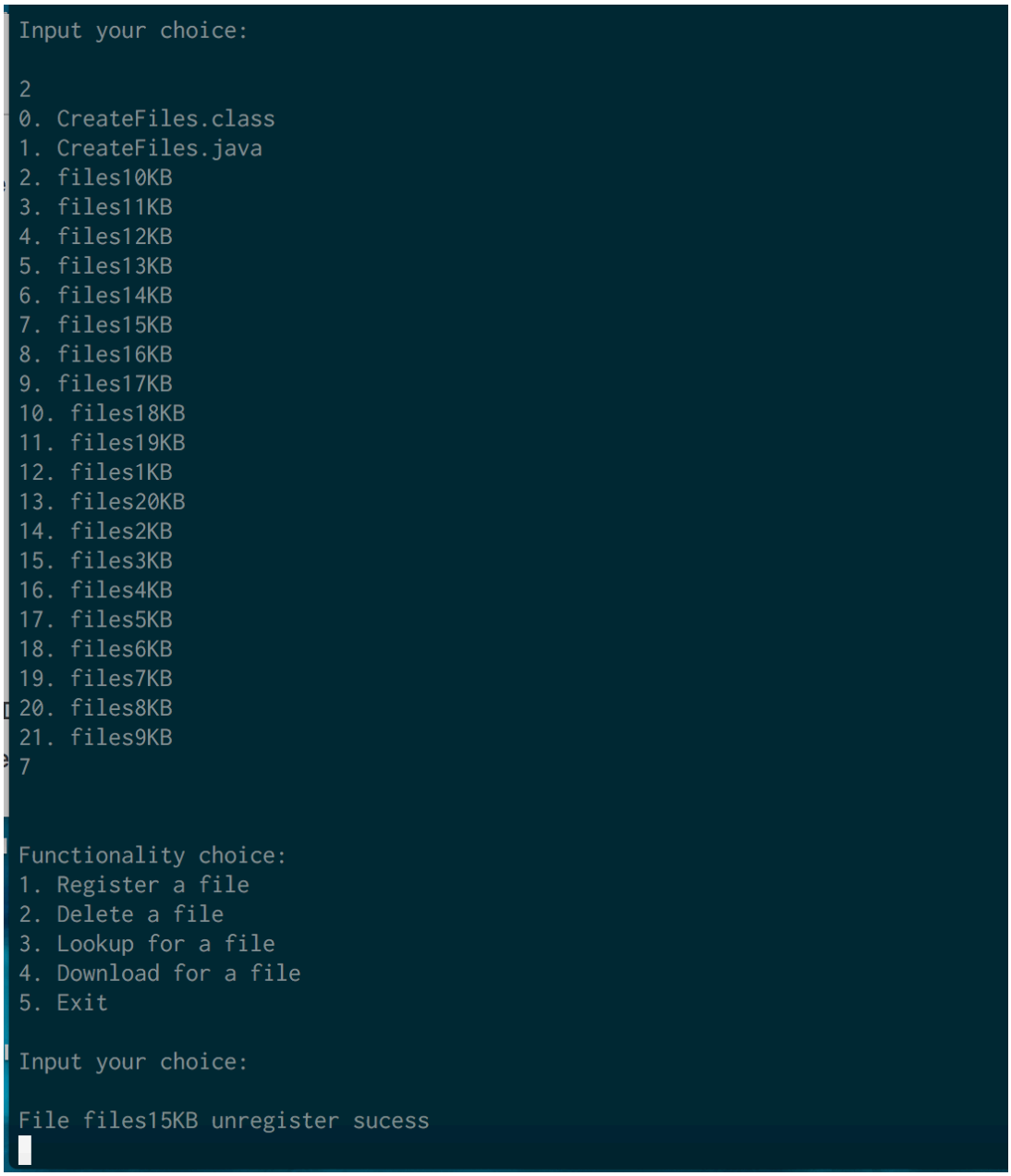
5. Delete a file

Again, to delete a file, we can either delete it in terminal or directly in the folder.

1) Delete a file from terminal.

Take client 1 as an example, as all the files stored in sharing folder has

been indexed in server, when I delete one file, it should automatically unregister the file in server. Here I choose 7, to delete files15KB.

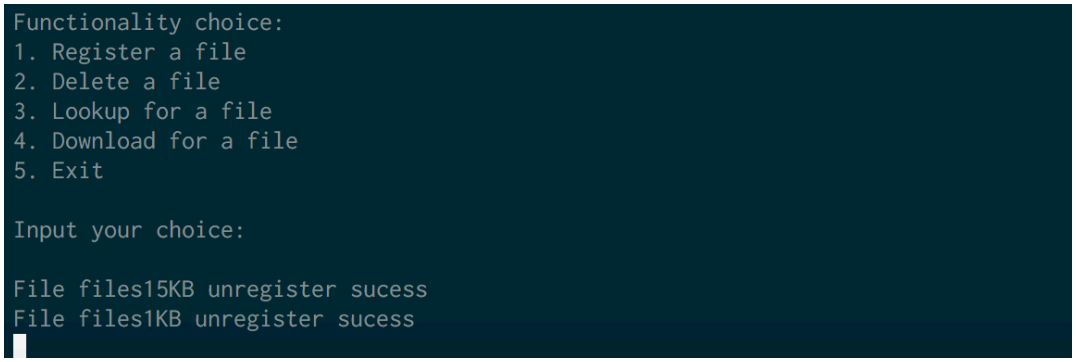


Because I set a file monitor to watch all the operations happened in the folder, so once a file is deleted, it will automatically unregister it from server side. You can see the unregister service starts a little later, because I set 5 seconds for the watcher to monitor the folder. So generally speaking, if the file monitor is on, to delete a file, I just need to delete it from the sharing folder and do nothing else. If the file monitor is off, the

programmer will automatically notify the server to unregister it. My programmer can work in both scenarios.

2) Directly delete a file in folder.

Deleting a file in folder has no difference with deleting from terminal. I directly delete a file in client 1's sharing folder, named files1KB.



We can see the client 1 will automatically unregister files1KB from server side. Again I give no instructions in terminal.

6. Look up a file

To look up a file, the server will firstly determine whether the given filename has different kinds of files (based on the MD5 value), if so, it will list all the files' size with this filename to the user, to choose one from it. And receiving the user's choice, then server will return all peers having this kind of file. If the server finds out that there is only one type of file exist, it will directly return all peers owning this file.
I will test all these two scenarios later.

1) The filename linking to only one type of file.

Client2 have the same file with client1 (I directly copy it from client1's folder), so let's lookup one file both indexed by client1 and client2, from client3 side. Here we take files8KB (which has been indexed by client1 and client2) as an example.

```
Input your choice:

3
0. files5KB
1. files4KB
2. files7KB
3. files6KB
4. files1KB
5. CreateFiles.java
6. files3KB
7. files2KB
8. files10KB
9. files12KB
10. files20KB
11. files11KB
12. files9KB
13. CreateFiles.class
14. files8KB
15. files18KB
16. files17KB
17. files19KB
18. files14KB
19. files13KB
20. files16KB
21. files15KB
22. PA1(1).pdf
23. Input another filename
14
Following peers have the file you are looking for:
PeerID:         IPAddress:              Bandwidth:
0. 1            127.0.0.1               10
1. 2            127.0.0.1               100
Do you want to download this file?       Y(yes)          N(no)
n
```

First when I choose function 3, it will firstly list all the indexed files in server side for user to choose, and also provide the option for user to input the filename. Here I directly choose 14, Files8KB. Then the result shows two peers, client1 and client2 have this file. Then the program also provides the download service for user to choose.

2) The filename has different kinds of files

Then we test if there exist two different files with the same file name indexed in the server, what would happen if we request from the server. As we mentioned, I modify the files10KB in client2, thus the server

should have two different files with name “files10KB” indexed. We now look up file name “files10KB”.

```
3
0. files5KB
1. files4KB
2. files7KB
3. files6KB
4. files1KB
5. CreateFiles.java
6. files3KB
7. files2KB
8. files10KB
9. files12KB
10. files20KB
11. files11KB
12. files9KB
13. CreateFiles.class
14. files8KB
15. files18KB
16. files17KB
17. files19KB
18. files14KB
19. files13KB
20. files16KB
21. files15KB
22. PA1(1).pdf
23. Input another filename
8
There exist different files with filename: files10KB
0. 280.0 KB
1. 560.01953125 KB
Choose which size you want to retrive
0
010
1
127.0.0.1
Following peers have the file you are looking for:
PeerID:         IPAddress:              Bandwidth:
0. 1            127.0.0.1               10
Do you want to download this file?       Y(yes)          N(no)
n
```

We can see the programmer will first show that there exist two different files with filename “files10KB” and provide each file size for user to choose. Then I choose the first one with 280KB size, then it will return only peer 1 has this file, which is exactly the same with the situation.

7. Download a file

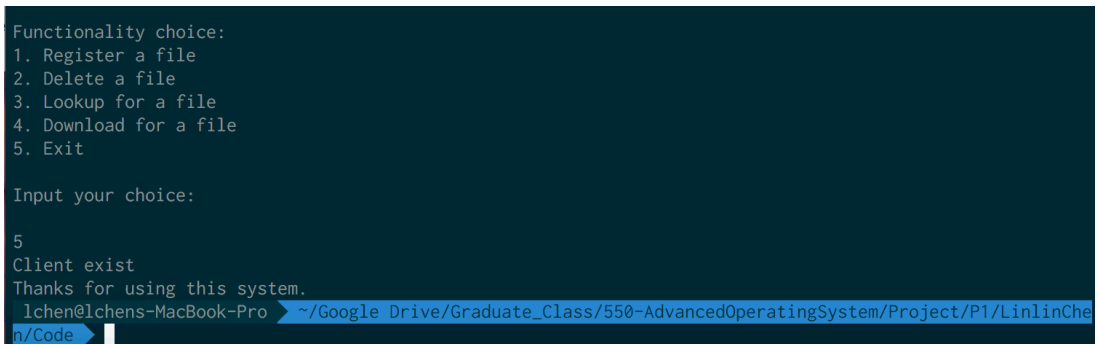
Now we test the function of downloading a file from another client.

```
7. files2KB
8. files20KB
9. files9KB
10. files8KB
11. Peer.java
12. files10KB
13. files12KB
14. files11KB
15. CreateFiles.class
16. files18KB
17. files17KB
18. files19KB
19. files14KB
20. files13KB
21. files16KB
22. files15KB
23. PA1(1).pdf
24. Input another filename
11
Following peers have the file you are looking for:
PeerID:         IPAddress:              Bandwidth:
0. 1            127.0.0.1               10
Input which peer you want to download file from
0
Connected with peer: 1...
Downloading file...
MD5 mathches the requested file
File Peer.java downloaded from peer 1 finished


Functionality choice:
1. Register a file
2. Delete a file
3. Lookup for a file
4. Download for a file
5. Exit

Input your choice:

File Peer.java register sucess
```

Here when I choose function 4 to download a file, it will firstly return a list of all indexed files for user to choose. The user could also input another name. Here I choose 11. “Peer.java” to download. It will firstly show only peer 1 has this file. Then I choose download the file from peer 1, and finally the file was successfully downloaded with MD5 value checking matching the original one. Also, after downloading the Peer.java, the program will automatically register this file in the server.

8. Exit the program

Simply input 5, the client will exit.



Above are the instructions for how to compile my code, how to start up the machine, and how to use each functionality.