Output after running multiple Clients.

- **Building Project using ant.**
  Execute command / DistributedPeerToPeerFileTransfer $ ant

```
aditya@ubuntu: ~/workspace/AOS/DistributedPeerToPeerFileTransfer
aditya@ubuntu:~/workspace/AOS/DistributedPeerToPeerFileTransfer$ ant
Buildfile: /home/aditya/workspace/AOS/DistributedPeerToPeerFileTransfer/build.xml

clean:
   [delete] Deleting directory /home/aditya/workspace/AOS/DistributedPeerToPeerFileTransfer/bin
   [delete] Deleting directory /home/aditya/workspace/AOS/DistributedPeerToPeerFileTransfer/dist

init:
    [mkdir] Created dir: /home/aditya/workspace/AOS/DistributedPeerToPeerFileTransfer/bin

compile:
    [javac] Compiling 6 source files to /home/aditya/workspace/AOS/DistributedPeerToPeerFileTransfer/b
in
    [javac] Note: /home/aditya/workspace/AOS/DistributedPeerToPeerFileTransfer/src/edu/distributedFile
System/ClientImpl.java uses unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.

dist:
    [mkdir] Created dir: /home/aditya/workspace/AOS/DistributedPeerToPeerFileTransfer/dist
      [jar] Building jar: /home/aditya/workspace/AOS/DistributedPeerToPeerFileTransfer/dist/Distribute
dP2PFileTransfer.jar

main:

BUILD SUCCESSFUL
Total time: 3 seconds
aditya@ubuntu:~/workspace/AOS/DistributedPeerToPeerFileTransfer$
```

- **Running a Client**
  - Execute **command** / **DistributedPeerToPeerFileTransfer** $ **./runClient.sh**
    - Clients gets Start and ready to accept request from another clients.

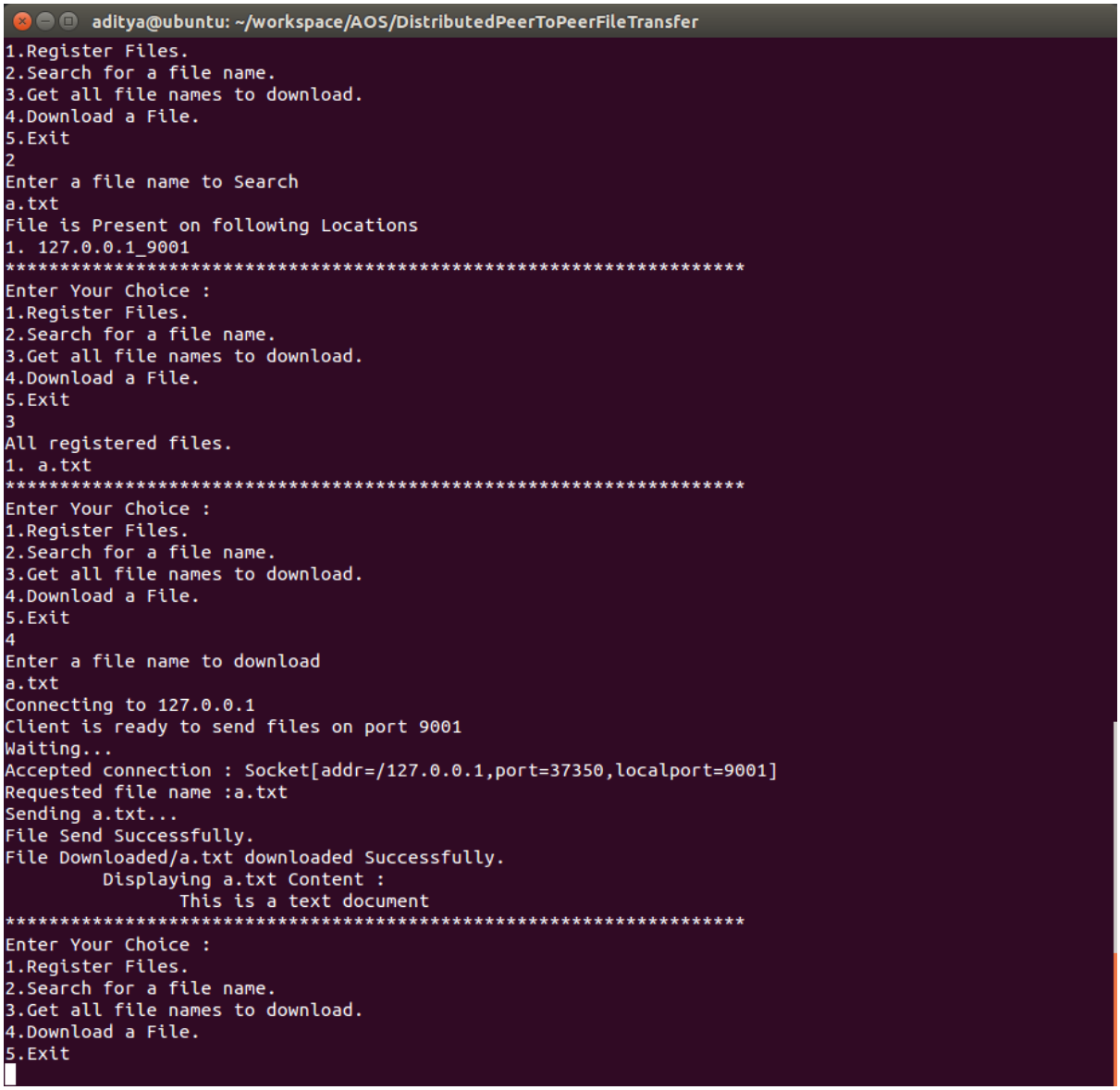
  - Now you can give input from client.
    1. **Register Files**→ It will register all files of “ClientData” folder and will store their locations in a distributed hash table of some machine
    2. **Search for a file name**→ It will take file name as an input from user and will search that file in distributed hash table. And will display on which server that value is present.
    3. **Get all file names to download** →It will display all file names which user can download.
    4. **Download a file** → It will take file name as an input from user and will search that file in distributed hash table and will download that file in “Downloaded” folder if found on any server.
    5. **Exit** → Will terminate the program.
    6. Also can Use **ctrl + C** to terminate the program.
  - Below screenshot shows the output after all the options are selected.

Register Files and Search

```
aditya@ubuntu: ~/workspace/AOS/DistributedPeerToPeerFileTransfer

aditya@ubuntu:~/workspace/AOS/DistributedPeerToPeerFileTransfer$ ./runClient.sh
Client is ready to send files on port 9001
Client 9991 is ready to put elements in DHT.
Waiting...
Waiting...
**********************************************************************
Enter Your Choice :
1.Register Files.
2.Search for a file name.
3.Get all file names to download.
4.Download a File.
5.Exit
1
Registering files on decentralized indexing server
**********************************************************************
Enter Your Choice :
1.Register Files.
2.Search for a file name.
3.Get all file names to download.
4.Download a File.
5.Exit
2
Enter a file name to Search
a.txt
File is Present on following Locations
1. 127.0.0.1_9001
**********************************************************************
Enter Your Choice :
1.Register Files.
2.Search for a file name.
3.Get all file names to download.
4.Download a File.
5.Exit
3
All registered files.
1. a.txt
**********************************************************************
Enter Your Choice :
1.Register Files.
2.Search for a file name.
3.Get all file names to download.
4.Download a File.
5.Exit
4
Enter a file name to download
a.txt
Connecting to 127.0.0.1
```

Download a File



Cases where this program will not work:

1. If all serverIp is not configure properly in config.properties. Client will not start.
2. If the port is not available, or already in use by some other applications. Then you need to change the port number.
3. Before starting all the Clients if user tries to do any operation this program will break. As system won't be able to create all sockets and initialize streams.