

COMMUNICATION NETWORKS

Peer to Peer File Sharing

In this project, we aim to create a file sharing protocol similar to Dropbox with support for download and upload for files and indexed searching . For this we set up server and clients on the system to enable File Sharing. It is essential that Server script has to be run prior to the execution of the client python script or the client script will fail.

Server.py

The host system which is willing to share folders on it , has this program being executed on it. The number corresponding to port over which connection is to be made , is taken as input. Socket is created and instantiated after getting imported from the Socket library. The socket must be bound to an address and listening for connections. For this we use , `socket.bind()` and `socket.accept()` , the return value of which is a pair (`conn`, `address`) where *conn* is a *new* socket object usable to send and receive data on the connection, and *address* is the address bound to the socket on the other end of the connection. This newly created socket is non inheritable. The arguments of the commands passed by client , are understood and parsed , functions are called depending on them.

It listens upto 5 clients queuing their requests here , in our code.

Client.py :

This is a program that interacts with the server

- **Socket:**

Initially socket library is imported and a socket is created & instantiated with 2 parameters , **socket.AF_INET** and **socket.SOCK_STREAM**. **AF_INET** refers to the address family ipv4 and The **SOCK_STREAM** means connection oriented TCP protocol that will be used to transport our messages in the network.

- Client takes the inputs i.e details about the host ip, port number across which connection should be made and the absolute path of shared folder located on the host , from the terminal.
- Later the connection between Host and Client over a port is made using **s.connect((host_ip, port))** and the access for the shared folder is checked
 - If the server is not available on the given host ip address , the socket is closed , connection is quit.
- Then the client prompts for the Query to be served ,
 - If no command is entered , or `close()` is requested , the socket connection will be closed using **s.close()** and **exit(0)** in the function `close_server()`. Else ,
 - According to the command entered , depending on if it's
 - i. IndexGet
 - ii. FileHash
 - iii. FileDownload

If it belongs to case i or ii the arguments are directly sent to the server and the received data is printed until a 'done' is received and a 'received' signal is sent. In case the data reception stops from server to client or received signal is not sent from client to server, a message saying "Connection error" is displayed.

In case of iii the filename and flag indicating TCP/UDP connection is sent along with arguments.

Specifications: This system incorporates the following set of commands:

1.IndexGet flag (args):

When the flag is **shortlist**,

Objective: Returns the 'name', 'size', 'timestamp' and 'type' of the files between the given start and end timestamps.

At the server,

1. Taking the input arguments of the command, it checks if the command is valid, else it displays an error message and sends **done**.
2. Uses the command, `os.popen` with giving argument **find**, option **-newermt** to find all files in the shared directory which are modified after timestamp1 and before timestamp2.
3. If there are no files modified during the time interval given, it displays "No Files Found" and waits for further input commands.
4. Else, for all the files in the list, using **stat** command with options, %n, %s, %F, %z for name, size in bytes, file type, last modified time stamp respectively are obtained and sent. If an interruption occurs or if it doesn't receive received for any data sent, it displays an error message saying it's a bad connection, otherwise a **done** is sent.

At the client,

1. It sends the input commands to server, keeps receiving data and keeps sending **received** signal till a **done** is received.
2. If the data reception is interrupted, an error message is displayed saying "Error in Connection" and the connection to the server is closed.
3. Once **done** is received, it stops.

```

Terminal
sathya@sathya-Inspiron-5559:~$ sudo -s python2 c.py
[sudo] password for sathya:
Host ip: 127.0.0.1
PORT: 1008
Path of Shared Folder: /home/sathya/Desktop/sr
Connection Established
Enter Command: IndexGet longlist
name: ./n2.py Size: 1168 bytes Type: regular file Timestamp:2019-03-19 19:53:41.772286187 +0530
name: ./n3.py Size: 489 bytes Type: regular file Timestamp:2019-03-23 10:23:00.012142671 +0530
name: ./n1.py Size: 971 bytes Type: regular file Timestamp:2019-03-19 19:53:48.084286488 +0530
name: ./n4.py Size: 519 bytes Type: regular file Timestamp:2019-03-23 10:23:52.512145174 +0530
Enter Command: IndexGet shortlist 2018-03-19 18:00:00 2019-03-23 10:00:00
name: ./n2.py Size: 1168 bytes Type: regular file Timestamp:2019-03-19 19:53:41.772286187 +0530
name: ./n1.py Size: 971 bytes Type: regular file Timestamp:2019-03-19 19:53:48.084286488 +0530
Enter Command: FileHash verify n1.py
file: n1.py
last modified: 2019-03-19 19:53:48.084286488 +0530
checksum: 3743154873
Enter Command: FileHash checkall
file: ./n2.py
last modified: 2019-03-19 19:53:41.772286187 +0530
checksum: 1473391975
file: ./n3.py
last modified: 2019-03-23 10:23:00.012142671 +0530
checksum: 454260300
file: ./n1.py
last modified: 2019-03-19 19:53:48.084286488 +0530
checksum: 3743154873
file: ./n4.py

```

When the flag is **longlist** ,

Objective: Returns the 'name' , 'size' , 'timestamp'(last modified) and 'type' of all the files in the shared folder.

Without any constraints on the modified times unlike to that in shortlist , here stat command is used to obtain and print name , size in bytes , file type , last modified time stamps of all the files in the shared directory. Similar to the actions in shortlist , till a **done** is received from server , the clients keeps receiving , sending **received** signal.

```

Terminal
sathya@sathya-Inspiron-5559:~$ sudo -s python2 c.py
[sudo] password for sathya:
Host ip: 127.0.0.1
PORT: 1008
Path of Shared Folder: /home/sathya/Desktop/sr
Connection Established
Enter Command: IndexGet longlist
name: ./n2.py Size: 1168 bytes Type: regular file Timestamp:2019-03-19 19:53:41.772286187 +0530
name: ./n3.py Size: 489 bytes Type: regular file Timestamp:2019-03-23 10:23:00.012142671 +0530
name: ./n1.py Size: 971 bytes Type: regular file Timestamp:2019-03-19 19:53:48.084286488 +0530
name: ./n4.py Size: 519 bytes Type: regular file Timestamp:2019-03-23 10:23:52.512145174 +0530
Enter Command: IndexGet shortlist 2018-03-19 18:00:00 2019-03-23 10:00:00
name: ./n2.py Size: 1168 bytes Type: regular file Timestamp:2019-03-19 19:53:41.772286187 +0530
name: ./n1.py Size: 971 bytes Type: regular file Timestamp:2019-03-19 19:53:48.084286488 +0530
Enter Command: FileHash verify n1.py
file: n1.py
last modified: 2019-03-19 19:53:48.084286488 +0530
checksum: 3743154873
Enter Command: FileHash checkall
file: ./n2.py
last modified: 2019-03-19 19:53:41.772286187 +0530
checksum: 1473391975
2 FileHash flag(args):
file: ./n3.py
last modified: 2019-03-23 10:23:00.012142671 +0530
checksum: 454260300
file: ./n1.py
last modified: 2019-03-19 19:53:48.084286488 +0530
checksum: 3743154873
file: ./n4.py

```

2.FileHash flag(args):

When the flag is **Verify** ,

Objective: Checks for the specific file name provided as command line argument and return its 'checksum' and 'last modified' timestamp.

Output: checksum and last modified timestamp of the input file.

At the server ,

1. Taking the input arguments of the command , it checks if the command is valid ,else it displays an error message and sends **done**.
2. Using the new socket object conn(in code , cs) and the file name in the parsed arguments,
3. **cksum** and **stat** commands for obtaining checksum and last modified time of the files are obtained [cksum [Option]... [File]... , stat [OPTION]... FILE...]
4. Once the filename , checksum and last modified time are sent through conn(cs) (i.e s in verify function) , if it doesn't receive **received**, an error message saying "Bad Connection Error" is displayed . Else, the server sends **done** .

At the Client ,

1. The client sends the input command arguments to server
2. While the server sends data, client receives data and sends **received** , prints data received.

- When it receives **done** , and if there is no input anymore , the connection to client closes, else it waits for the next input command.

```
Terminal
Connection Established
Enter Command: IndexGet longlist
name: ./n2.py Size: 1168 bytes Type: regular file Timestamp:2019-03-19 19:53:41.772286187 +0530
name: ./n3.py Size: 489 bytes Type: regular file Timestamp:2019-03-23 10:23:00.012142671 +0530
name: ./n1.py Size: 971 bytes Type: regular file Timestamp:2019-03-19 19:53:48.084286488 +0530
name: ./n4.py Size: 519 bytes Type: regular file Timestamp:2019-03-23 10:23:52.512145174 +0530
Enter Command: IndexGet shortlist 2018-03-19 18:00:00 2019-03-23 10:00:00
name: ./n2.py Size: 1168 bytes Type: regular file Timestamp:2019-03-19 19:53:41.772286187 +0530
name: ./n1.py Size: 971 bytes Type: regular file Timestamp:2019-03-19 19:53:48.084286488 +0530
Enter Command: FileHash verify n1.py
file: n1.py
last modified: 2019-03-19 19:53:48.084286488 +0530
checksum: 3743154873
Enter Command: FileHash checkall
file: ./n2.py
last modified: 2019-03-19 19:53:41.772286187 +0530
checksum: 1473391975
file: ./n3.py
last modified: 2019-03-23 10:23:00.012142671 +0530
checksum: 454260300
file: ./n1.py
last modified: 2019-03-19 19:53:48.084286488 +0530
checksum: 3743154873
file: ./n4.py
last modified: 2019-03-23 10:23:52.512145174 +0530
checksum: 2635826307
Enter Command:
Bye
sathya@sathya-Inspiron-5559:~$
```

When the flag is **Checkall** ,

Objective: Check perform what 'verify' does for all the files in the shared folder.

Output : Filename , checksum and last modified timestamp of all the files in the shared directory

For **checkall**, the procedure followed for **verify** is repeated for all the files in the shared folder.


```
Terminal
Connection Established
Enter Command: IndexGet longlist
name: ./n2.py Size: 1168 bytes Type: regular file Timestamp:2019-03-19 19:53:41.772286187 +0530
name: ./n3.py Size: 489 bytes Type: regular file Timestamp:2019-03-23 10:23:00.012142671 +0530
name: ./n1.py Size: 971 bytes Type: regular file Timestamp:2019-03-19 19:53:48.084286488 +0530
name: ./n4.py Size: 519 bytes Type: regular file Timestamp:2019-03-23 10:23:52.512145174 +0530
Enter Command: IndexGet shortlist 2018-03-19 18:00:00 2019-03-23 10:00:00
name: ./n2.py Size: 1168 bytes Type: regular file Timestamp:2019-03-19 19:53:41.772286187 +0530
name: ./n1.py Size: 971 bytes Type: regular file Timestamp:2019-03-19 19:53:48.084286488 +0530
Enter Command: FileHash verify n1.py
file: n1.py
last modified: 2019-03-19 19:53:48.084286488 +0530
checksum: 3743154873
Enter Command: FileHash checkall
file: ./n2.py
last modified: 2019-03-19 19:53:41.772286187 +0530
checksum: 1473391975
file: ./n3.py
last modified: 2019-03-23 10:23:00.012142671 +0530
checksum: 454260300
file: ./n1.py
last modified: 2019-03-19 19:53:48.084286488 +0530
checksum: 3743154873
file: ./n4.py
last modified: 2019-03-23 10:23:52.512145174 +0530
checksum: 2635826307
Enter Command:
Bye
sathya@sathya-Inspiron-5559:~$
```

3. FileDownload flag (args):

Objective: This command is used to download files from the shared folder of connected user to our shared folder. FileDownload can be done over TCP or UDP depending on users request.

Output: Contains the filename, filesize, lastmodified timestamp & the MD5hash of the requested file.

3.1 File download using TCP :

Transmission Control Protocol is a standard that defines how to establish and maintain a network conversation via which application programs can exchange data. First we create the Socket is created using socket.socket() and the socket type is specified as socket.SOCK_STREAM. When you do that, the default protocol that's used is the TCP. The file is directly sent from server to client without going through any intermediate server.

In TCP it is a 3-way Handshake Protocol :

- Server is waiting for a connection
- Client sends the arguments
- Server sends the data
- Client replies with a **received** signal.

Advantages of TCP:

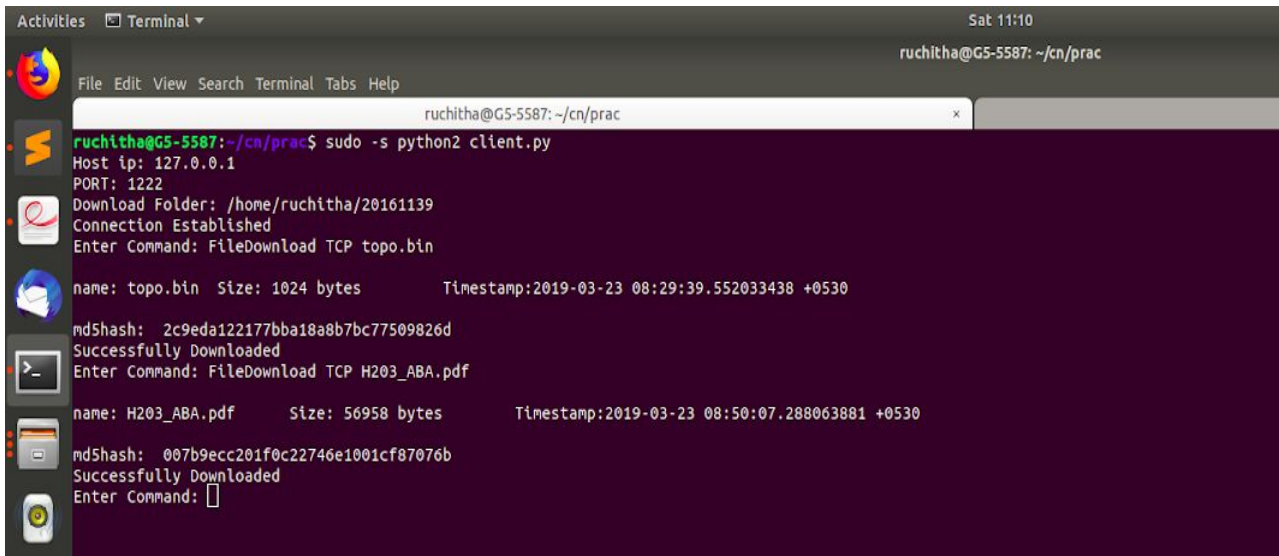
- TCP is reliable as the packets dropped in the network are detected and retransmitted by the sender.
- It provides in-order data delivery as data is read by your application in the order it was written by the sender.

Server :

1. Server establishes a connection to the obtained client's address over TCP using a constant port.
2. The server obtains the arguments from the client which includes the filename to be downloaded. Here server opens the file if it has necessary **permissions**.
3. And then the server reads the file information and sends it over the socket (max of 1024 bytes at once). The client replies with a "received" signal for each transmission.
4. At the EOF the server sends a "done" signal to indicate the end of transmission.
5. MD5 hash of the original file is calculated and sent to the client.

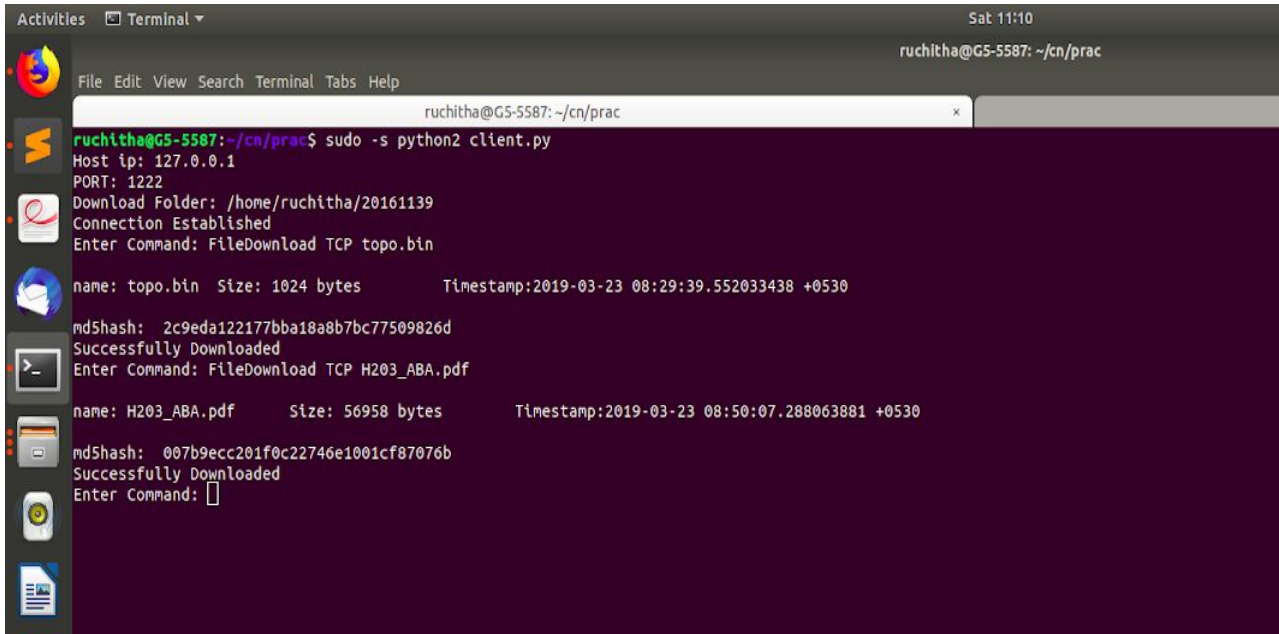
Client:

1. Client establishes connection with the server when server attempts to do so.
2. The receiving client listens to the TCP port for incoming connection.
3. Client checks for the proper syntax of the arguments and sends them to the server.
4. Client then reads the information sent by the server and replies with a "received" signal until it receives a "done" signal at the EOF .
5. At last MD5 hash of the downloaded file is calculated and compared with the MD5 hash of original file to see if the file is successfully downloaded.



```
Activities Terminal Sat 11:10
ruchitha@GS-5587: ~/cn/prac
File Edit View Search Terminal Tabs Help
ruchitha@GS-5587: ~/cn/prac
ruchitha@GS-5587:~/cn/prac$ sudo -s python2 client.py
Host ip: 127.0.0.1
PORT: 1222
Download Folder: /home/ruchitha/20161139
Connection Established
Enter Command: FileDownload TCP topo.bin
name: topo.bin Size: 1024 bytes Timestamp:2019-03-23 08:29:39.552033438 +0530
md5hash: 2c9eda122177bba18a8b7bc77509826d
Successfully Downloaded
Enter Command: FileDownload TCP H203_ABA.pdf
name: H203_ABA.pdf Size: 56958 bytes Timestamp:2019-03-23 08:50:07.288063881 +0530
md5hash: 007b9ecc201f0c22746e1001cf87076b
Successfully Downloaded
Enter Command: 
```

Fig 1 :Downloading binary and pdf files using TCP connection



```
Activities Terminal Sat 11:10
ruchitha@GS-5587: ~/cn/prac
File Edit View Search Terminal Tabs Help
ruchitha@GS-5587: ~/cn/prac
ruchitha@GS-5587:~/cn/prac$ sudo -s python2 client.py
Host ip: 127.0.0.1
PORT: 1222
Download Folder: /home/ruchitha/20161139
Connection Established
Enter Command: FileDownload TCP topo.bin

name: topo.bin Size: 1024 bytes Timestamp:2019-03-23 08:29:39.552033438 +0530
md5hash: 2c9eda122177bba18a8b7bc77509826d
Successfully Downloaded
Enter Command: FileDownload TCP H203_ABA.pdf

name: H203_ABA.pdf Size: 56958 bytes Timestamp:2019-03-23 08:50:07.288063881 +0530
md5hash: 007b9ecc201f0c22746e1001cf87076b
Successfully Downloaded
Enter Command: 
```

Fig 2 :Downloading csv and py files using TCP connection

3.2 File download using UDP :

User Datagram Protocol is a connectionless protocol which is useful for communication which requires efficient communication and doesn't have a problem with packet loss. The Socket is created using `socket.socket()` and the socket type is specified as `socket.SOCK_DGRAM`. UDP sends messages called datagrams and is considered best-effort mode of communication.

Advantages of UDP:

- UDP is efficient in communication as it does not require connection setup.
i.e no handshaking dialogues.
- There are no handshaking dialogues which makes it suitable for time-sensitive applications.

Disadvantages of UDP:

- UDP isn't reliable as the datagrams dropped in the network may not be detected.
- The datagrams received and read by the reader may be out of order from the writes.

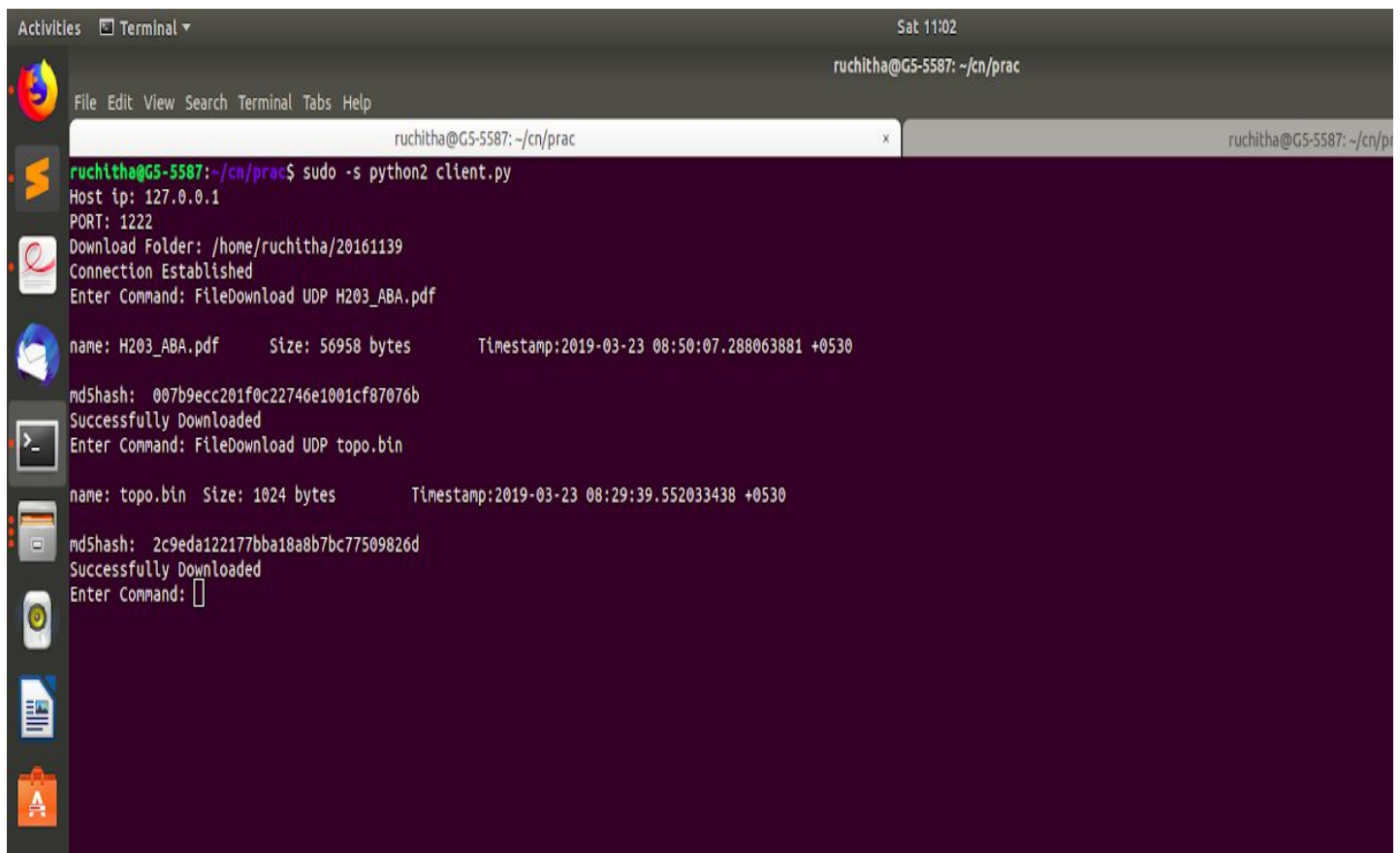
Server :

1. Server establishes a connection to the obtained client's address over TCP using a constant port.
2. The server obtains the arguments from the client which includes the filename to be downloaded.
3. We need to create a socket through which we will be receiving UDP datagrams from the client. Server processes the datagram for client's address and sends data.
4. And then the server reads the file information and sends it over the socket (max of 1024 bytes at once). The client replies with a "received" signal for each transmission.

5. At the EOF the server sends a “done” to indicate the end of transmission.
6. MD5 hash of the original hash is calculated and sent to the client.

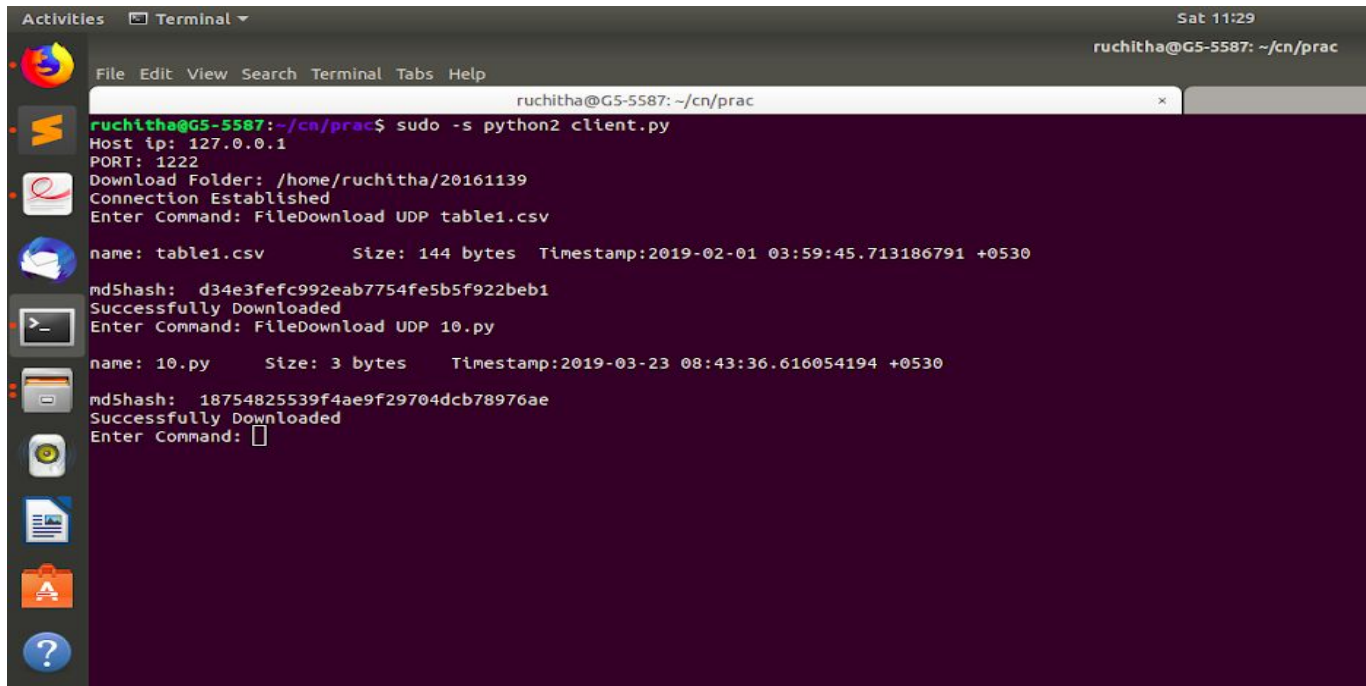
Client:

1. We need to declare the host ip address and port number over which we will be sending messages.
2. The client first sends a datagram with it's address and listens to the port for incoming replies.
3. Client checks for the proper syntax of the arguments and sends them to the server.
4. Client creates a socket which looks exactly like the one created in the server.
5. Client then reads the information sent by the server and replies with a “received” signal 201 until it receives a “done” signal at the EOF .
6. At last MD5 hash of the downloaded file is calculated and compared with the MD5 hash of original file to see if the file is successfully downloaded.



```
Activities Terminal Sat 11:02
ruchitha@G5-5587: ~/cn/prac
File Edit View Search Terminal Tabs Help
ruchitha@G5-5587: ~/cn/prac
ruchitha@G5-5587:~/cn/prac$ sudo -s python2 client.py
Host ip: 127.0.0.1
PORT: 1222
Download Folder: /home/ruchitha/20161139
Connection Established
Enter Command: FileDownload UDP H203_ABA.pdf
name: H203_ABA.pdf      Size: 56958 bytes      Timestamp:2019-03-23 08:50:07.288063881 +0530
md5hash: 007b9ecc201f0c22746e1001cf87076b
Successfully Downloaded
Enter Command: FileDownload UDP topo.bin
name: topo.bin  Size: 1024 bytes      Timestamp:2019-03-23 08:29:39.552033438 +0530
md5hash: 2c9eda122177bba18a8b7bc77509826d
Successfully Downloaded
Enter Command: 
```

Fig 3 :Downloading binary and pdf files using UDP connection

A screenshot of a Linux terminal window. The window title is "Terminal" and it shows the user "ruchitha@G5-5587" in the directory "~/cn/prac". The terminal output shows the execution of a Python script "client.py" which establishes a UDP connection to a host at IP 127.0.0.1 on port 1222. The script successfully downloads two files: "table1.csv" (144 bytes, timestamp 2019-02-01 03:59:45.713186791 +0530) and "10.py" (3 bytes, timestamp 2019-03-23 08:43:36.616054194 +0530). The terminal also shows the MD5 hashes for both files and the prompt "Enter Command:" after each download.

```
ruchitha@G5-5587:~/cn/prac$ sudo -s python2 client.py
Host ip: 127.0.0.1
PORT: 1222
Download Folder: /home/ruchitha/20161139
Connection Established
Enter Command: FileDownload UDP table1.csv

name: table1.csv      Size: 144 bytes   Timestamp:2019-02-01 03:59:45.713186791 +0530
md5hash: d34e3fefc992eab7754fe5b5f922beb1
Successfully Downloaded
Enter Command: FileDownload UDP 10.py

name: 10.py          Size: 3 bytes    Timestamp:2019-03-23 08:43:36.616054194 +0530
md5hash: 18754825539f4ae9f29704dcb78976ae
Successfully Downloaded
Enter Command: 
```

Fig 4 :Downloading csv and py files using UDP connection

Bonus:

Shortlist:

First we need to import re and use re.search to find filenames with '.pdf' and '.txt' from the files selected within the given timestamps.

In **Fig5** , in the shared db folder all the highlighted files were modified files between given timestamps , with .pdf or .txt extensions.

In **Fig6**, in the shared db folder resume.pdf was added and modified at 17:20:00 which means it is not modified within the given timestamp and shortlist should not display it.

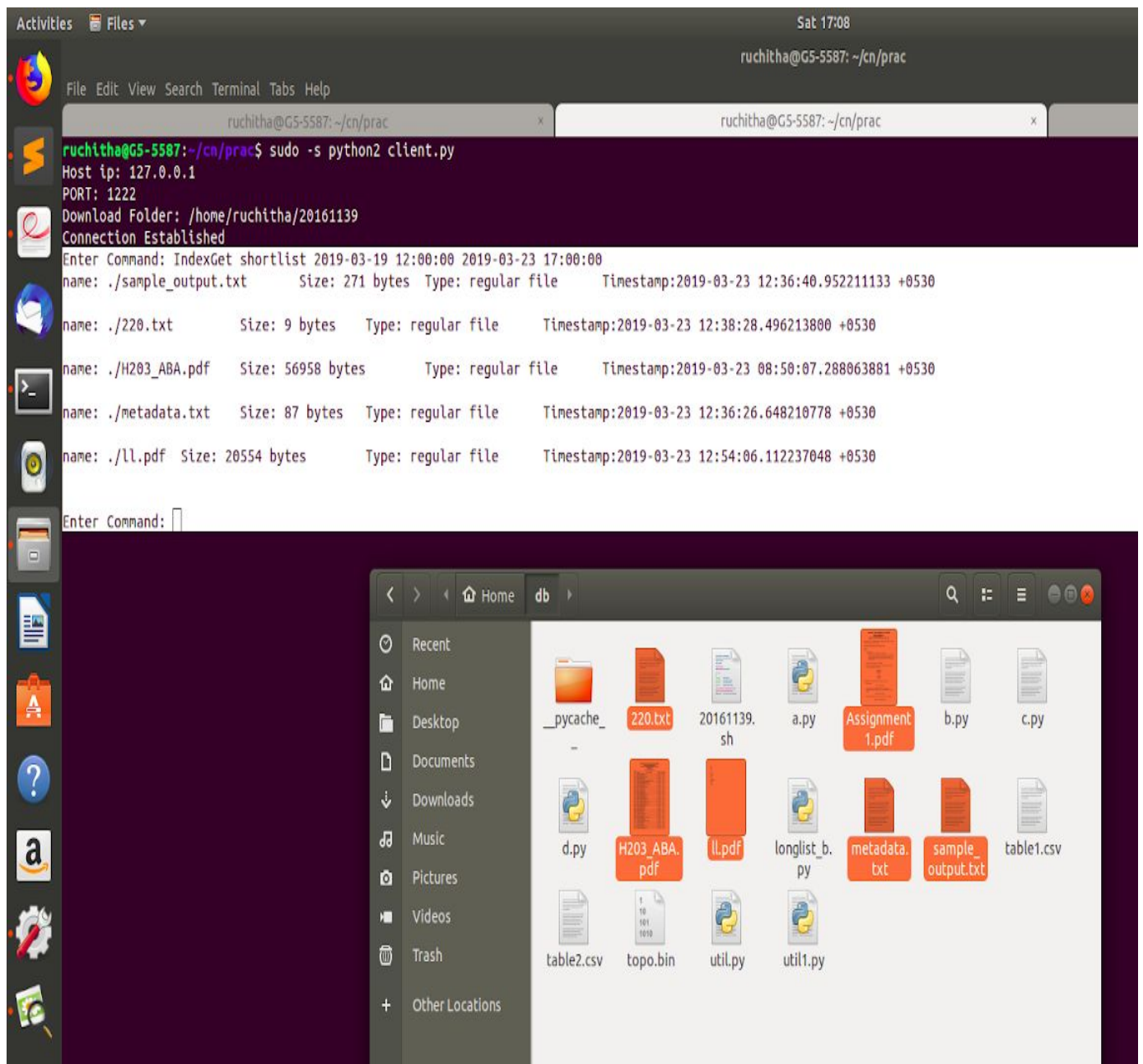


Fig5: Shortlist listing the .pdf and .txt files modified b/w given timestamps

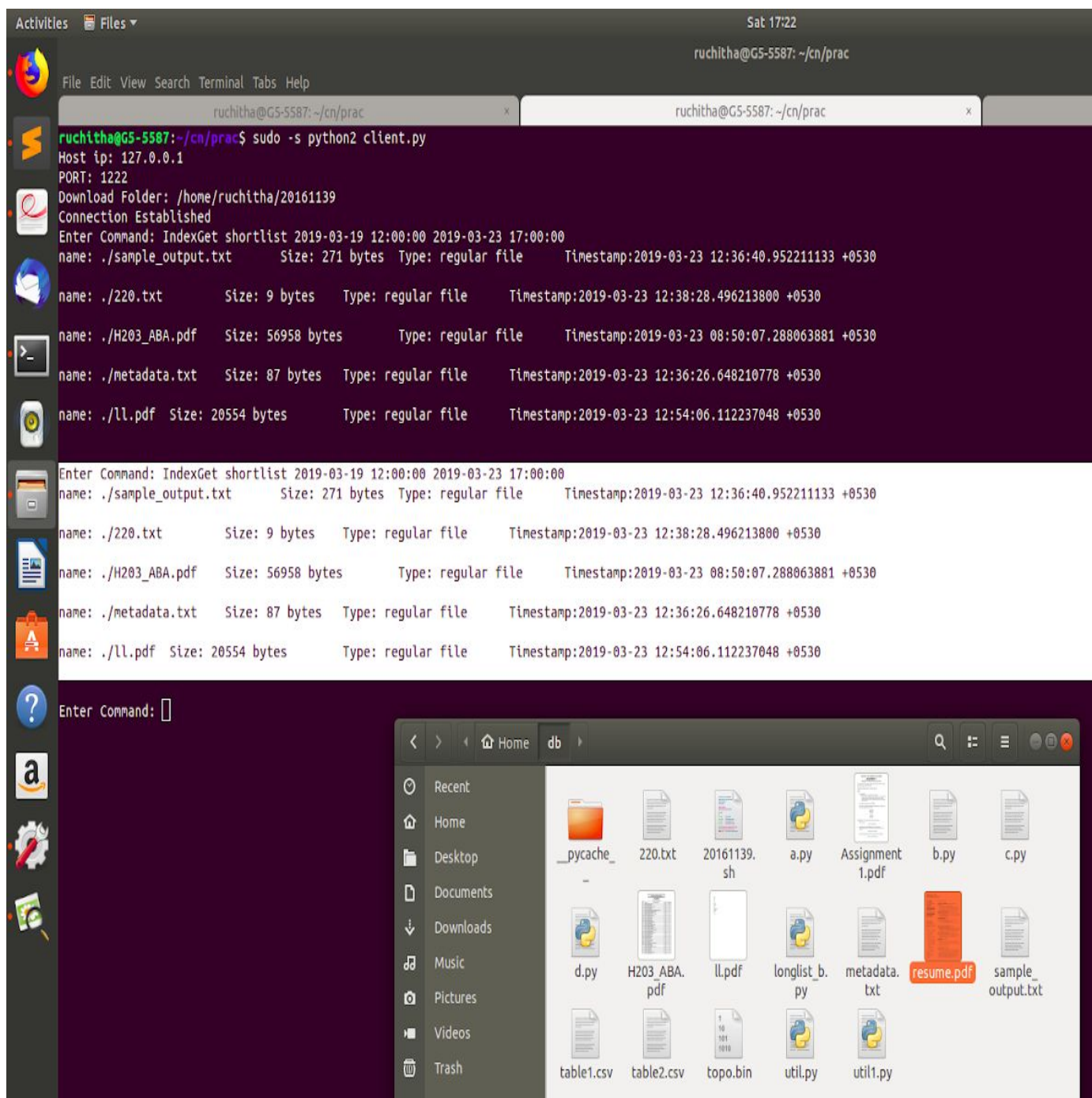


Fig6: Shortlist not listing the .pdf and .txt files modified before/after given timestamps

Bonus:

Longlist:

First we need to import re, dataminer and StringIO. With the help of data miner in the function pdf_with_pro() we can extract text from pdf and then using re.search() we can check if the text has the word “**Programmer**” in it. Even though there were many .pdf files in the shared folder the given CN_Assignment1 file has the word “Programmer” and ll.pdf shown in the figure has the word in the 10th line and hence only these files were listed by longlist.

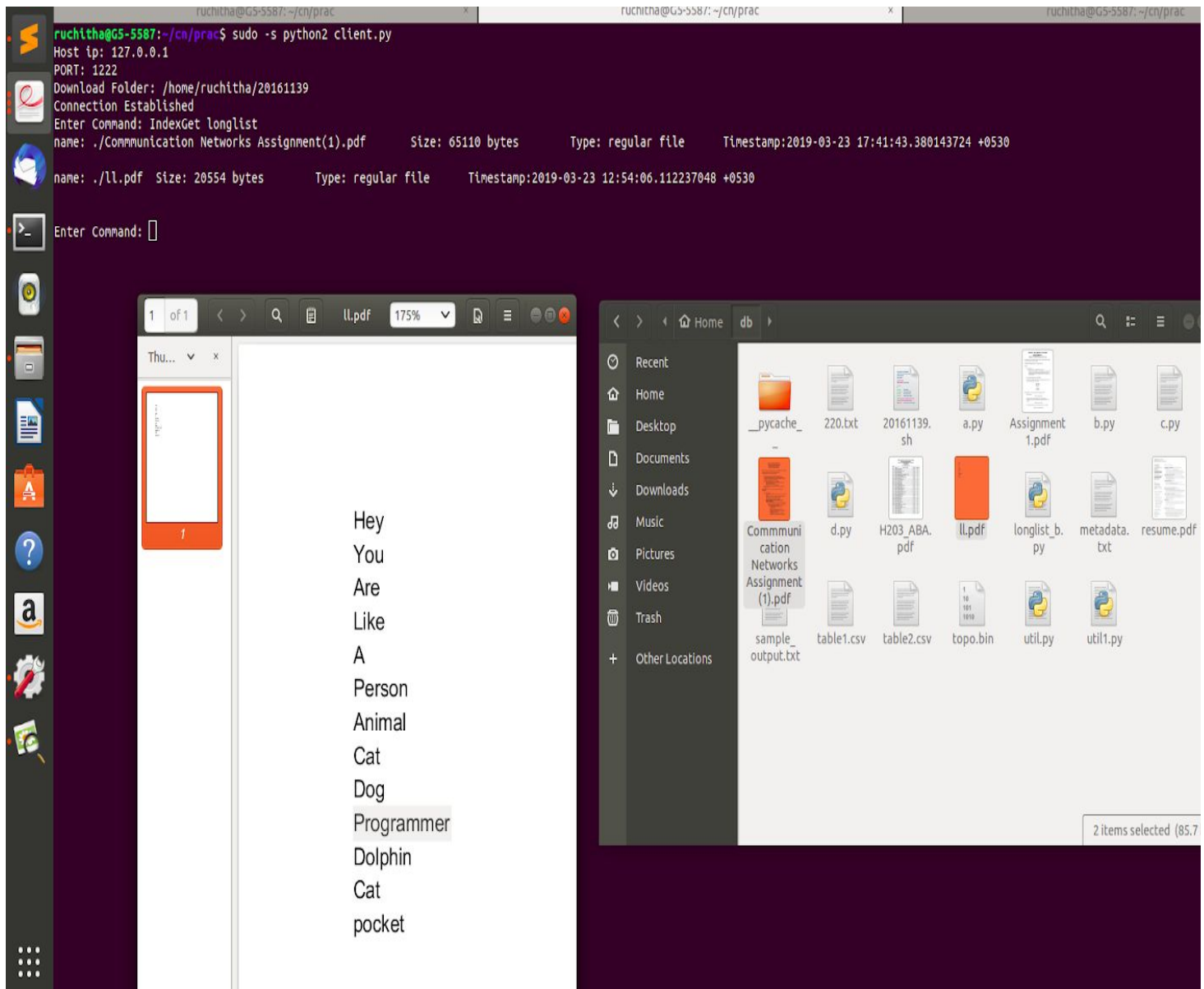


Fig7: Longlist listing the .pdf with the word “Programmer” in it.

Longlist bonus for .txt:

First we need to read the txt file and search for the word “Programmer” in every line and output only those with the given word. It is clear from the given figure that only ll.txt and 220.txt have the given word which are listed out by longlist.

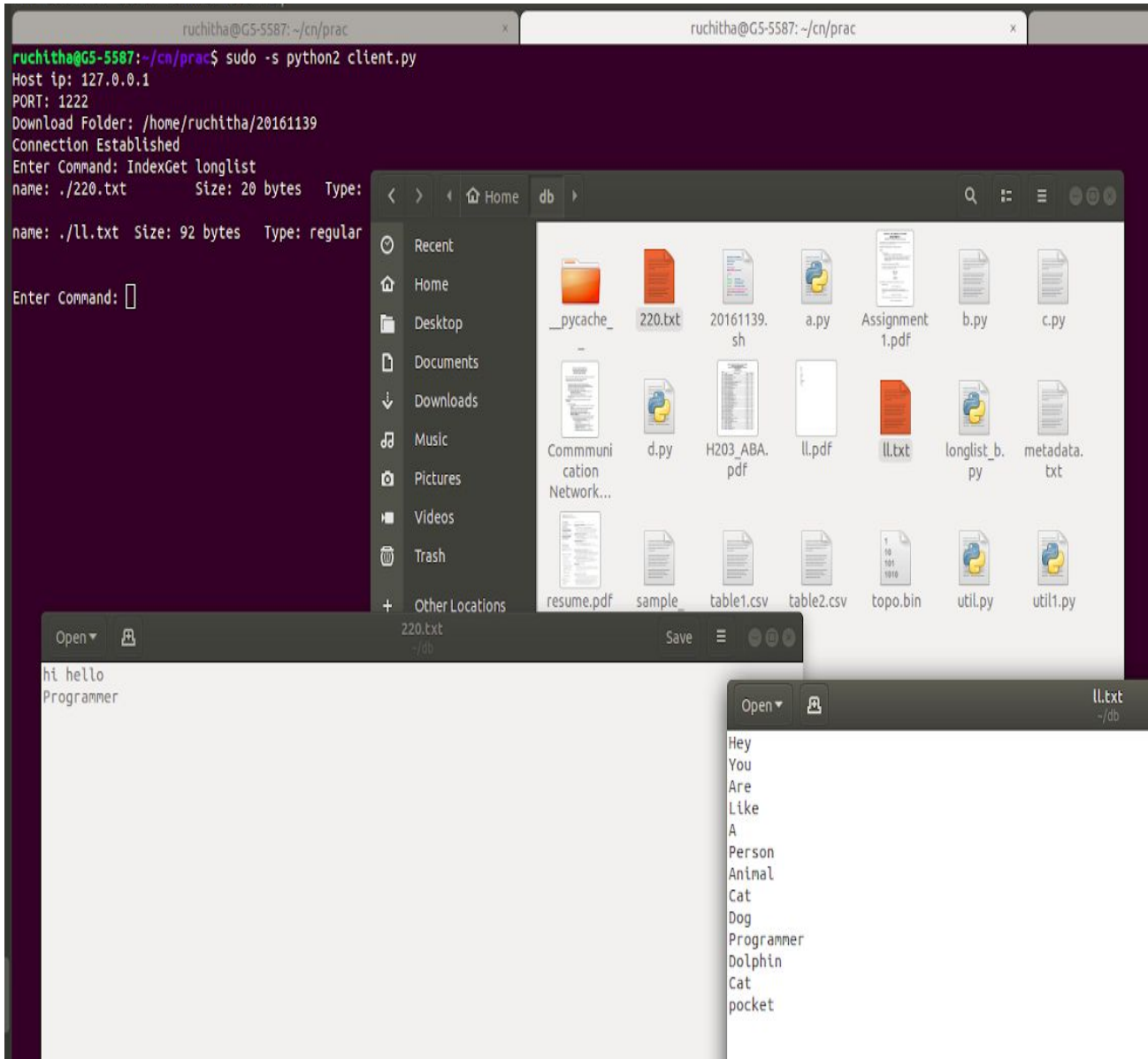


Fig8: Longlist listing the .txt with the word “Programmer” in it.

References

<https://docs.python.org/3/library/socket.html>

<https://stackoverflow.com/questions/5377139/how-to-find-all-files-ending-with-rb-with-linux>