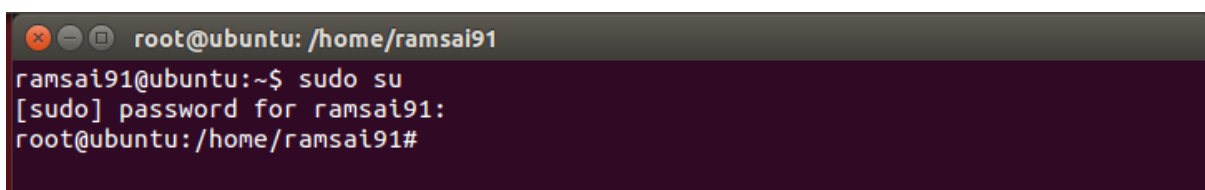


MANUAL DOCUMENT

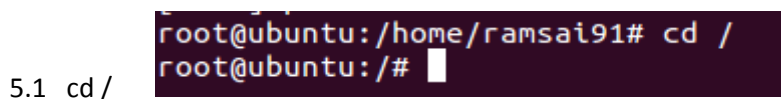
Constraints: Make sure, the ports which you give as command line arguments are free before running the program.

1. Download and Extract the Zip file "PROG3_KANNAN_SAIRAM.zip" file from the Blackboard.
2. You will have a folder Assignment 3 which contains the source code.
3. Open the terminal and Login as root user by typing the following command.
 - 3.1 `sudo su` (You will be asked to enter your superuser password. Type your superuser password.)
4. Once You enter your password the symbol will change from \$ to # , which means you are logged in as root.



```
root@ubuntu: /home/ramsa91
ramsa91@ubuntu:~$ sudo su
[sudo] password for ramsa91:
root@ubuntu: /home/ramsa91#
```

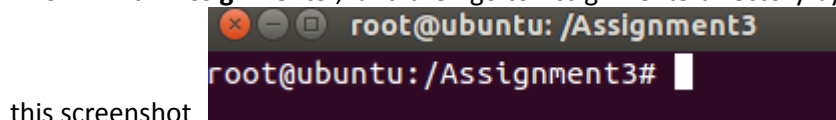
5. Go to the root directory by typing the following command



```
root@ubuntu: /home/ramsa91# cd /
root@ubuntu: /#
```

6. create a directory named "Assignment 3" inside root by typing the following command.

- 6.1 `mkdir Assignment3` , and then go to Assignment3 directory by typing `cd /Assignment 3` .You will get



```
root@ubuntu: /Assignment3
root@ubuntu: /Assignment3#
```

this screenshot

7. Move the files from the downloaded Assignment3 folder to the created Assignment 3 directory

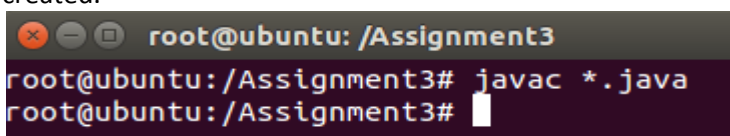
- 7.1 Syntax: `mv<source location> <target location>`

In this case, source location is where the file are downloaded and target location is the Assignment 2 directory. For Example [`mv /home/<username>/Downloads/Assignment3 /Assignment3`]

8. Once you execute the above command, the files inside Assignment3 folder will be moved to /Assignment2 directory.

9. Go to Assignment Directory (`cd /Assignment3`). You will see eight java files inside Assignment3 Directory along with two subfolders having peers containing different files.

10. Compile all the files once using the command, `javac *.java` . The class files for each java file will be created.



```
root@ubuntu: /Assignment3
root@ubuntu: /Assignment3# javac *.java
root@ubuntu: /Assignment3#
```

```

root@ubuntu:/Assignment3/peer1# javac *.java
Note: peerClient.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
root@ubuntu:/Assignment3/peer1#

```

```

root@ubuntu:/Assignment3/peer2# javac *.java
Note: peerClient.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
root@ubuntu:/Assignment3/peer2#

```

11. Run the peer.java file, with the parameters as mentioned in the design document.

For Example: **java peer -i 1 -h ubuntu -p 4000 -m 32 -f**

-i Stands for ID of the peer, -h stands for Host name, -p stands for Port number to be used, -m stands for Maximum ID value of the peer, -f to indicate it is the first host in the peer to peer network. Once you execute this command, you will get message like Server 1 started. (This is the first peer)

```

root@ubuntu:/Assignment3# java peer -i 1 -h ubuntu -p 4000 -m 32 -f
Server 1 Started!

```

12. Open new terminal, go to /Assignment3 directory as root, and execute the peer.java file with the following parameters. (This is the second peer)

For example: **java peer -i 2 -h ubuntu -p 4001 -m 32 -r ubuntu -s 4000**

Here -r stands for host name of other peers, -s stands for portnumber of any existing peer connected in a peer to peer network.

You will get message like this: Server 2 started

```

root@ubuntu:/Assignment3# java peer -i 2 -h ubuntu -p 4001 -m 32 -r ubuntu -s 4000
00
Hostname: ubuntu Port: 4001 ID: 2 NextHostname: ubuntu NextPort: 4000 NextID: 1
MaxID: 32 isFirstPeer: false Hashtable: {}
Hostname: ubuntu Port: 4001 ID: 2 NextHostname: ubuntu NextPort: 4000 NextID: 1
MaxID: 32 isFirstPeer: false Hashtable: {}
Hostname: ubuntu Port: 4001 ID: 2 NextHostname: ubuntu NextPort: 4000 NextID: 1
MaxID: 32 isFirstPeer: false Hashtable: {}
Server 2 Started!

```

You can see peer2 connected to peer1

Meanwhile in peer1, you can see the message it is connected to peer 2

```

root@ubuntu:/Assignment3# java peer -i 1 -h ubuntu -p 4000 -m 32 -f
Server 1 Started!

Hostname: ubuntu Port: 4000 ID: 1 NextHostname: ubuntu NextPort: 4000 NextID: 1 M
axID: 32 isFirstPeer: true Hashtable: {}
Hostname: ubuntu Port: 4000 ID: 1 NextHostname: ubuntu NextPort: 4001 NextID: 2 M
axID: 32 isFirstPeer: true Hashtable: {}
Hostname: ubuntu Port: 4000 ID: 1 NextHostname: ubuntu NextPort: 4001 NextID: 2 M
axID: 32 isFirstPeer: true Hashtable: {}

```

13. Similarly open new terminal and connect peer3 to existing peers , using the command.

For example: **java peer -i 3 -h ubuntu -p 4002 -m 32 -r ubuntu -s 4000**

You can see the message like Server 3 started and its connected to peer1.

```
root@ubuntu:/Assignment3# java peer -i 3 -h ubuntu -p 4002 -m 32 -r ubuntu -s 4000
Hostname: ubuntu Port: 4002 ID: 3 NextHostname: ubuntu NextPort: 4000 NextID: 1
MaxID: 32 isFirstPeer: false Hashtable: {}
Hostname: ubuntu Port: 4002 ID: 3 NextHostname: ubuntu NextPort: 4000 NextID: 1
MaxID: 32 isFirstPeer: false Hashtable: {}
Hostname: ubuntu Port: 4002 ID: 3 NextHostname: ubuntu NextPort: 4000 NextID: 1
MaxID: 32 isFirstPeer: false Hashtable: {}
Hostname: ubuntu Port: 4002 ID: 3 NextHostname: ubuntu NextPort: 4000 NextID: 1
MaxID: 32 isFirstPeer: false Hashtable: {}
Hostname: ubuntu Port: 4002 ID: 3 NextHostname: ubuntu NextPort: 4000 NextID: 1
MaxID: 32 isFirstPeer: false Hashtable: {}
Server 3 Started!
```

14. Similarly you can connect many servers to peer to peer network, For our consideration let us take , three servers connected to peer to peer network. In order to speak to any one of the peer, Go to /cd/Assignment3 directory and enter the following command.

For example, if you want to speak with server 1, type the following command

Syntax: **telnet <host name> <port number of the required peer>**

For server1 , port number is 4000, Therefore the command is : **telnet ubuntu 4000** , you will get following message like this

```
root@ubuntu:/Assignment2# telnet ubuntu 4000
Trying 127.0.1.1...
Connected to ubuntu.
Escape character is '^]'.

```

To write (put) a file , enter the command ADD followed by the filename, **ADD a.txt**

```
root@ubuntu:/Assignment3# telnet ubuntu 4000
Trying 127.0.1.1...
Connected to ubuntu.
Escape character is '^]'.
ADD a.txt
3171_a3/1.0 ADD 0 400 NotResponsibleCRLFADD a.txtCRLF
```

For server2, port number is 4001, Therefore the command is : **telnet ubuntu 4001**, you will get following message like this

To write (put) a file, enter the command ADD followed by the filename, **ADD c.txt**

```

ramesai91@ubuntu:~$ telnet ubuntu 4001
Trying 127.0.1.1...
Connected to ubuntu.
Escape character is '^]'.
ADD c.txt
3171_a3/1.0 ADD 0 400 NotResponsibleCRLFADD c.txtCRLF

```

The filename is stored in the server based on the hash function. You will see in which server it is stored in the output file. Similarly you can add multiple ADD message (put commands). The message will be stored in an appropriate server based on the DHT.

15. Once the files are added, the files should be registered to the Indexing Server, such that peers can look-up to the files for downloading. Run the Indexing server using the command **java IndexServer** in the new terminal

```

root@ubuntu:/Assignment3# java IndexServer
ubuntu/127.0.1.1
ubuntu:4233

```

16. Open two new terminals, run the `peerClient.java` for peer1 and peer2, for registering its file to the server. For Example in peer1, register `a.txt` and for peer2, register `c.txt`

```

root@ubuntu:/Assignment3/peer1# java peerClient
Trying to connect with server. 127.0.1.1
[b.txt, d.txt, peerClient.java~, b.txt~, a.txt, peerClient.java, d.txt~, peerClient.class, peerClient.java~~]
Enter the filename to be registered from the above list
a.txt
Enter the filename to be searched

```

```

root@ubuntu:/Assignment3/peer2# java peerClient
Trying to connect with server. 127.0.1.1
[peerClient.java~, c.txt, f.txt, peerClient.java, e.txt, peerClient.class]
Enter the filename to be registered from the above list
c.txt

```

17. Now the files are registered, you will be asked to enter the filename to be looked up in other peers. . peer1 will look for "c.txt" from the index server which is mapped to distributed hash table. Once the "c.txt" file is found in peer2, It goes to peer2's Ip address and downloads the file "c.txt" automatically. Similarly peer2 looks for "a.txt" from index server which is mapped to distributed hash table . Once "a.txt" is found in peer1, it goes to peer1's Ip address and downloads "a.txt" automatically. Since Ip address is same, a dedicated port is given for sending and receiving of files.

```

Trying to connect with server. 127.0.1.1
[b.txt, d.txt, peerClient.java~, b.txt~, a.txt, peerClient.java, d.txt~, peerClient.class, peerClient.java~~]
Enter the filename to be registered from the above list
a.txt
Enter the filename to be searched
c.txt
Waiting for connection to be established with the peer

```

```

root@ubuntu:/Assignment3/peer2# java peerClient
Trying to connect with server. 127.0.1.1
[peerClient.java~, c.txt, f.txt, peerClient.java, e.txt, peerClient.class]
Enter the filename to be registered from the above list
c.txt
Enter the filename to be searched
a.txt

```

The required Downloaded file will be shown in the output file.

15. The Query command (get command) in step 14, will let us know whether the file is present in the given server.

```

root@ubuntu:/Assignment3# telnet ubuntu 4000
Trying 127.0.1.1...
Connected to ubuntu.
Escape character is '^]'.
ADD a.txt
3171_a3/1.0 ADD 0 400 NotResponsibleCRLFADD a.txtCRLF
QUERY c.txt
3171_a3/1.0 ADD 0 400 NotResponsibleCRLFQUERY c.txtCRLF

```

```

ramsaig91@ubuntu:~$ telnet ubuntu 4001
Trying 127.0.1.1...
Connected to ubuntu.
Escape character is '^]'.
ADD c.txt
3171_a3/1.0 ADD 0 400 NotResponsibleCRLFADD c.txtCRLF
QUERY a.txt
3171_a3/1.0 ADD 0 400 NotResponsibleCRLFQUERY a.txtCRLF

```

In this case, Not responsible means, this peer does not contain the given file. The code 400 stands that a peer in the peer to peer network is existing with the given file. Similarly you can give multiple get commands for each peers, to check which file is located in which server.

The output file shows the hash table and the log file for all peer to peer operations and their results.