

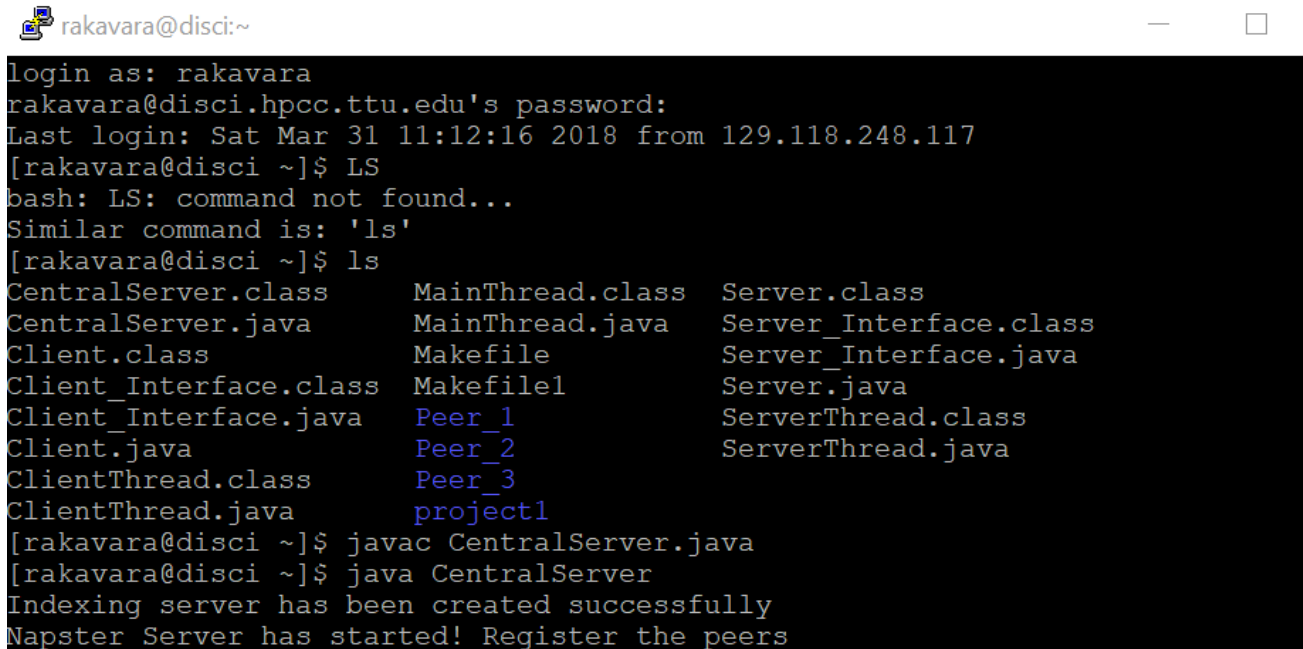
Advanced Operating Systems Test Documentation

Rishika Reddy Akavaram
R11523879

Case-1:

Running Central Index Server

CentralServer.java acts as a Napster central indexing server. It maintains the log of all the peers that register with it and their contents.

A terminal window titled 'rakavara@disci:~' showing the execution of a Java program. The user logs in as 'rakavara' and enters their password. The terminal shows the last login time and IP address. The user then runs 'LS' which fails, and 'ls' which lists the files in the current directory. The files listed are CentralServer.class, CentralServer.java, Client.class, Client_Interface.class, Client_Interface.java, Client.java, ClientThread.class, ClientThread.java, MainThread.class, MainThread.java, Makefile, Makefile1, Peer_1, Peer_2, Peer_3, project1, Server.class, Server_Interface.class, Server_Interface.java, Server.java, ServerThread.class, and ServerThread.java. The user then runs 'javac CentralServer.java' and 'java CentralServer'. The output shows 'Indexing server has been created successfully' and 'Napster Server has started! Register the peers'.

Central Indexing Server

This screenshot shows central Indexing server has been created successfully

Test Result: Pass

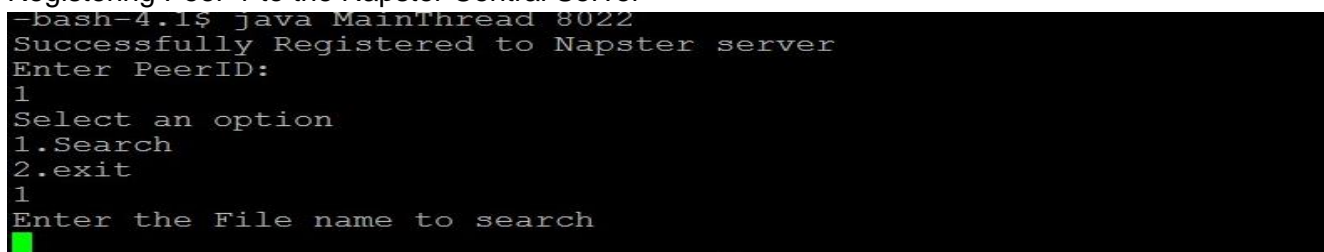
Case-2: Registering peers on Index server

ThreadMainClass.java acts as a client. When a client is connected to Central Indexing Server it will register all its files with the indexing server. The server then builds the index for the peer.

When client gets the peerId then all the file will automatically register using register method. The register method consists of two parameters. One is peerid and other is filename.

Peer-1:

Registering Peer-1 to the Napster Central Server

A terminal window showing the execution of a Java program. The user runs '-bash-4.1\$ java MainThread 8022'. The output shows 'Successfully Registered to Napster server'. The user then enters 'Enter PeerID:' and '1'. The program then prompts 'Select an option' and the user enters '1'. The program then prompts 'Enter the File name to search' and the user enters a file name (partially visible as '1').

Peer-2:

Registering peer-2 to the Napster Central Server

```
-bash-4.1$ java MainThread 8033
Successfully Registered to Napster server
Enter PeerID:
2
Select an option
1.Search
2.exit
1
Enter the File name to search
```

Peer-3:


Registering peer-3 to the Napster Central Server

```
-bash-4.1$ java MainThread 8333
Successfully Registered to Napster server
Enter PeerID:
3
Select an option
1.Search
2.exit
1
Enter the File name to search
file11.txt
File:file11.txt is found on:
1
2
Enter the peer you want to connect:
```

Case-3: Search for File

Each peer can search files when connected to index server and returns all the matching peers to the requestor.

i. List of files that are assigned to each peer before transfer(before execution of obtain method):

 rakavara@disci:~/Peer_3

```
login as: rakavara
rakavara@disci.hpcc.ttu.edu's password:
Last login: Sat Mar 31 14:20:11 2018 from 129.118.248.119
[rakavara@disci ~]$ LS
bash: LS: command not found...
Similar command is: 'ls'
[rakavara@disci ~]$ ls
CentralServer.class      MainThread.class      Server.class
CentralServer.java      MainThread.java      Server_Interface.class
Client.class             Makefile              Server_Interface.java
Client_Interface.class   Makefile1             Server.java
Client_Interface.java    Peer_1                ServerThread.class
Client.java              Peer_2                ServerThread.java
ClientThread.class       Peer_3
ClientThread.java        project1
[rakavara@disci ~]$ cd Peer_1
[rakavara@disci Peer_1]$ ls
file11.txt  file15.txt  file1.txt  file6.txt  file8.txt
file14.txt  file16.txt  file3.txt  file7.txt
[rakavara@disci Peer_1]$ cd ..
[rakavara@disci ~]$ cd Peer_2
[rakavara@disci Peer_2]$ ls
file11.txt  file14.txt  file18.txt  file20.txt  file2.txt
file13.txt  file17.txt  file19.txt  file21.txt  file3.txt
[rakavara@disci Peer_2]$ cd ..
[rakavara@disci ~]$ cd Peer_3
[rakavara@disci Peer_3]$ ls
file10.txt  file14.txt  file18.txt  file2.txt  file6.txt
file13.txt  file16.txt  file20.txt  file5.txt  file9.txt
```

ii. Searching for a particular file from one of the peer :

- Here we are searching for file11.txt from peer-3

- As we have already connected to peer-3 as shown above and selected the option to search. Now we have to enter the file name that we want to search.
- In the figure below we are searching file11.txt is present in both the peers
- It returns peer-1 and peer-2

```

Enter PeerID:
3
Select an option
1.Search
2.exit
1
Enter the File name to search
file11.txt
File:file11.txt is found on:
1
2
Enter the peer you want to connect:

```

Test Result: pass

Case-4: - Obtaining(Downloading) Files

i. Selecting a Peer from List of returned Peers

- Now peer-3 selects 1 in order to retrieve file from peer-1
- File is then successfully transferred to peer-3

```

Enter PeerID:
3
Select an option
1.Search
2.exit
1
Enter the File name to search
file11.txt
File:file11.txt is found on:
1
2
Enter the peer you want to connect:
1
Successfully Connected
8022
File Transfer Sucessfull

```

ii. List of files in the peer after downloading

Peer-3 now contains File11.txt in its list of files as shown in the figure below

```

Client.class      Client.java      MainThread.class  Makefile1      Peer_3  Server_Interface.class  ServerThread.class
[rakavara@disci ~]$ cd Peer_3
[rakavara@disci Peer_3]$ ls
file10.txt file11.txt file13.txt file14.txt file16.txt file18.txt file20.txt file2.txt file5.txt file6.txt file9.txt

```

Case-5:- Calculating Average Response time

