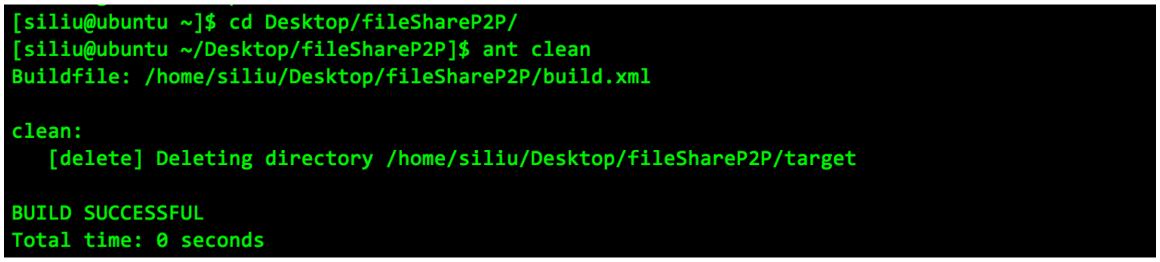
File Sharing System Manual

Si Liu
A20334820

This system is primarily developed and tested in Eclipse. It can be run in Eclipse normally.

As the assignment required, "Ant" is used to automate the compilation. This manual will list the ant compile instructions step by step, and interactions with the system through Linux terminal. (Ant build file: build.xml)

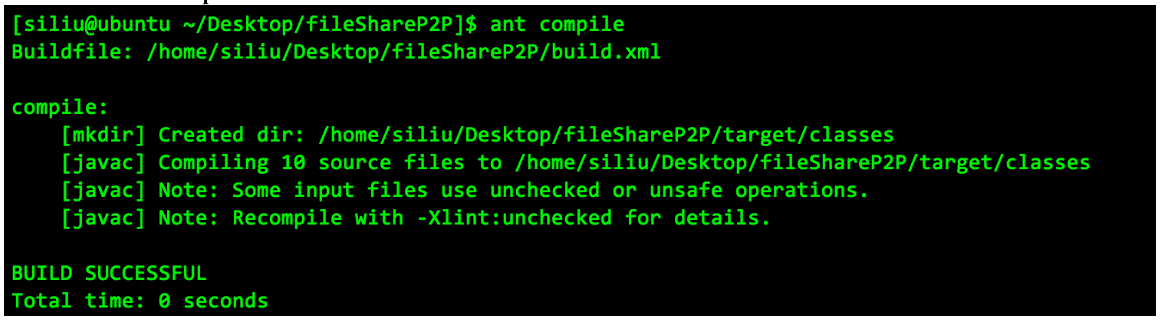
1. Ant compile:
   Get into the project folder (fileShareP2P) first before running any command:
   cd fileShareP2P
   a. ant clean



```
[siliu@ubuntu ~]$ cd Desktop/fileShareP2P/
[siliu@ubuntu ~/Desktop/fileShareP2P]$ ant clean
Buildfile: /home/siliu/Desktop/fileShareP2P/build.xml

clean:
   [delete] Deleting directory /home/siliu/Desktop/fileShareP2P/target

BUILD SUCCESSFUL
Total time: 0 seconds
```

Figure 1 The result of "ant clean"

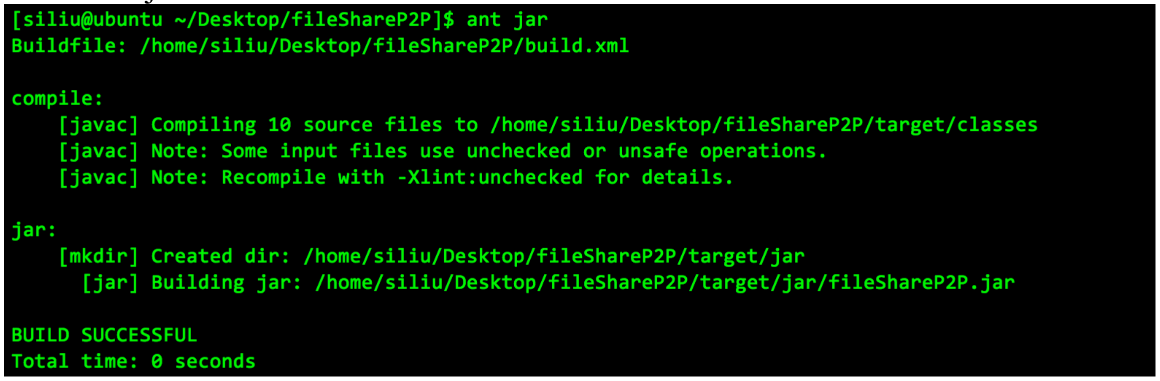
   b. ant compile



```
[siliu@ubuntu ~/Desktop/fileShareP2P]$ ant compile
Buildfile: /home/siliu/Desktop/fileShareP2P/build.xml

compile:
    [mkdir] Created dir: /home/siliu/Desktop/fileShareP2P/target/classes
    [javac] Compiling 10 source files to /home/siliu/Desktop/fileShareP2P/target/classes
    [javac] Note: Some input files use unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.

BUILD SUCCESSFUL
Total time: 0 seconds
```

Figure 2 The result of "ant compile"

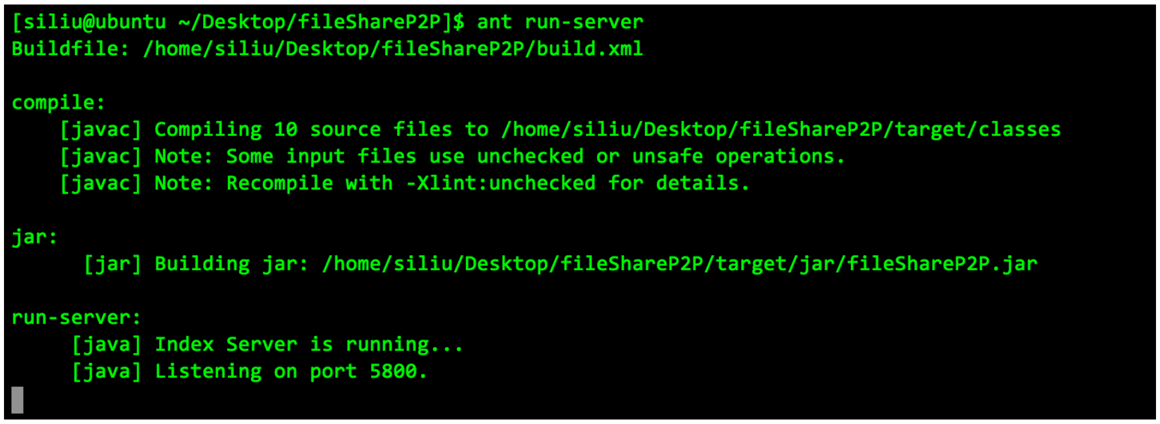
   c. ant jar



```
[siliu@ubuntu ~/Desktop/fileShareP2P]$ ant jar
Buildfile: /home/siliu/Desktop/fileShareP2P/build.xml

compile:
    [javac] Compiling 10 source files to /home/siliu/Desktop/fileShareP2P/target/classes
    [javac] Note: Some input files use unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.

jar:
    [mkdir] Created dir: /home/siliu/Desktop/fileShareP2P/target/jar
      [jar] Building jar: /home/siliu/Desktop/fileShareP2P/target/jar/fileShareP2P.jar

BUILD SUCCESSFUL
Total time: 0 seconds
```

Figure 3 The result of "ant jar"

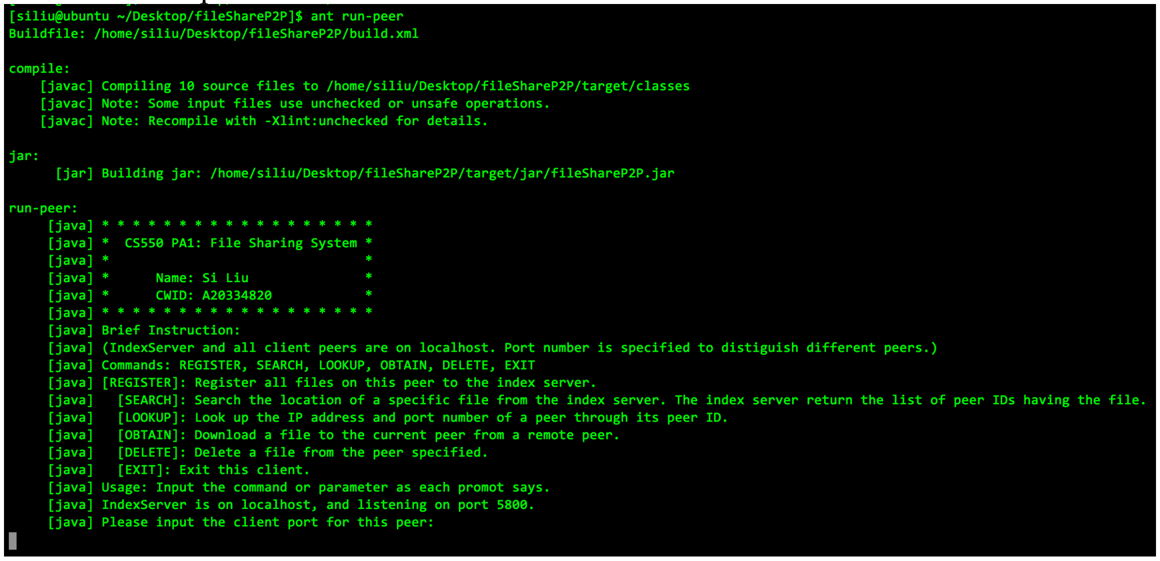
2. Start Index Server
    a. ant run-server



```
[siliu@ubuntu ~/Desktop/fileShareP2P]$ ant run-server
Buildfile: /home/siliu/Desktop/fileShareP2P/build.xml

compile:
    [javac] Compiling 10 source files to /home/siliu/Desktop/fileShareP2P/target/classes
    [javac] Note: Some input files use unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.

jar:
      [jar] Building jar: /home/siliu/Desktop/fileShareP2P/target/jar/fileShareP2P.jar

run-server:
     [java] Index Server is running...
     [java] Listening on port 5800.
```

Figure 4 Index server start display

3. Start Peer
    a. ant run-peer



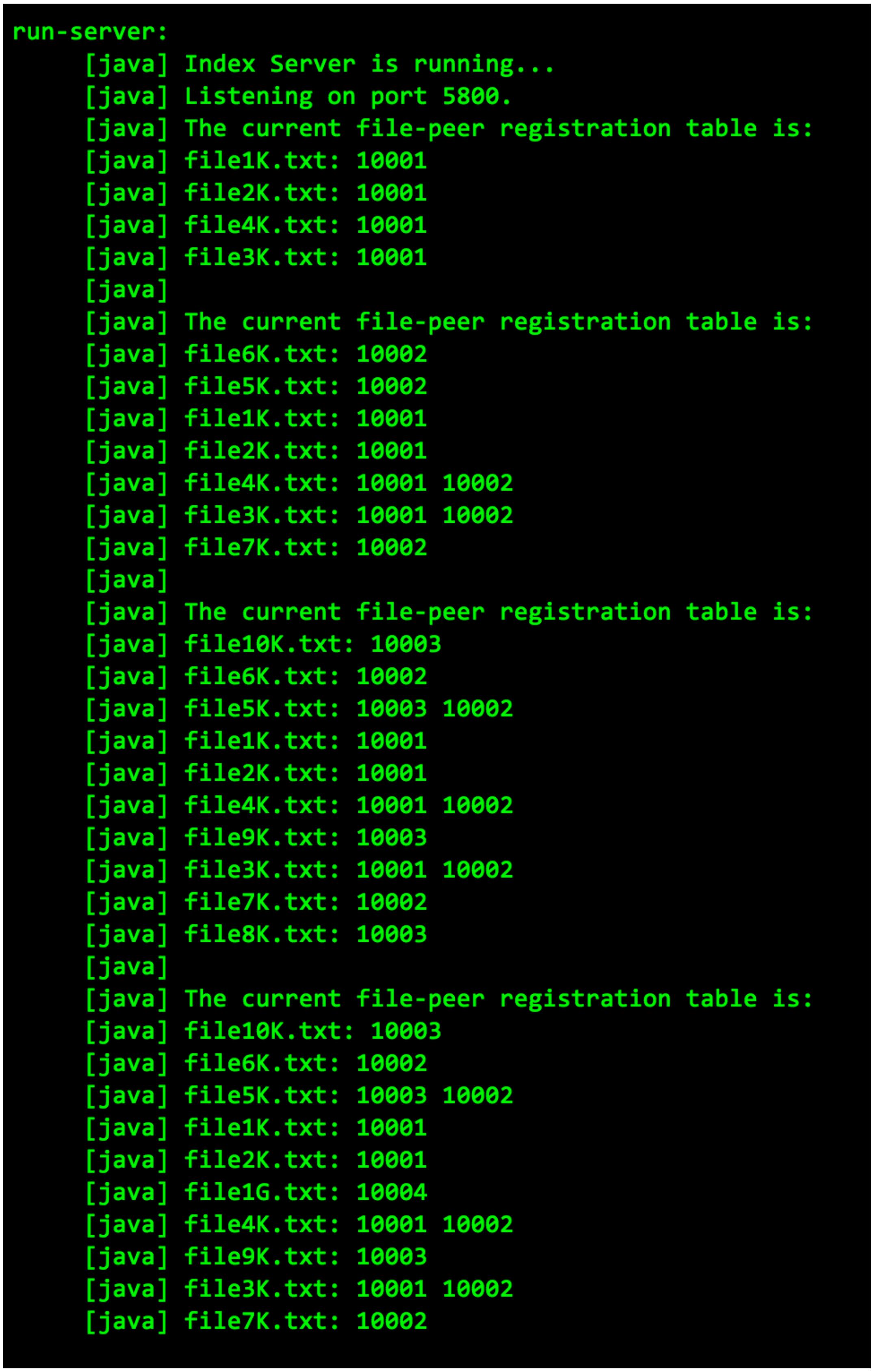
```
[siliu@ubuntu ~/Desktop/fileShareP2P]$ ant run-peer
Buildfile: /home/siliu/Desktop/fileShareP2P/build.xml

compile:
    [javac] Compiling 10 source files to /home/siliu/Desktop/fileShareP2P/target/classes
    [javac] Note: Some input files use unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.

jar:
      [jar] Building jar: /home/siliu/Desktop/fileShareP2P/target/jar/fileShareP2P.jar

run-peer:
     [java] * * * * * * * * * * * * * * * * * *
     [java] *  CS550 PA1: File Sharing System *
     [java] *                                 *
     [java] *      Name: Si Liu               *
     [java] *      CWID: A20334820            *
     [java] * * * * * * * * * * * * * * * * * *
     [java] Brief Instruction:
     [java] (IndexServer and all client peers are on localhost. Port number is specified to distiguish different peers.)
     [java] Commands: REGISTER, SEARCH, LOOKUP, OBTAIN, DELETE, EXIT
     [java] [REGISTER]: Register all files on this peer to the index server.
     [java]   [SEARCH]: Search the location of a specific file from the index server. The index server return the list of peer IDs having the file.
     [java]   [LOOKUP]: Look up the IP address and port number of a peer through its peer ID.
     [java]   [OBTAIN]: Download a file to the current peer from a remote peer.
     [java]   [DELETE]: Delete a file from the peer specified.
     [java]   [EXIT]: Exit this client.
     [java] Usage: Input the command or parameter as each promot says.
     [java] IndexServer is on localhost, and listening on port 5800.
     [java] Please input the client port for this peer:
```

Figure 5 Peer start display

In this experiment, we start 4 peers. This table displays the peer port and id mapping relation.

Peer Port	Peer Id
5700	10001
5701	10002
5702	10003
5703	10004

As long as peer port is input, automatic registration starts. The index server side will display the result (full file-peer registration table). The following figure displays the changes in the index server side for automatic registration of 4 peers startup.



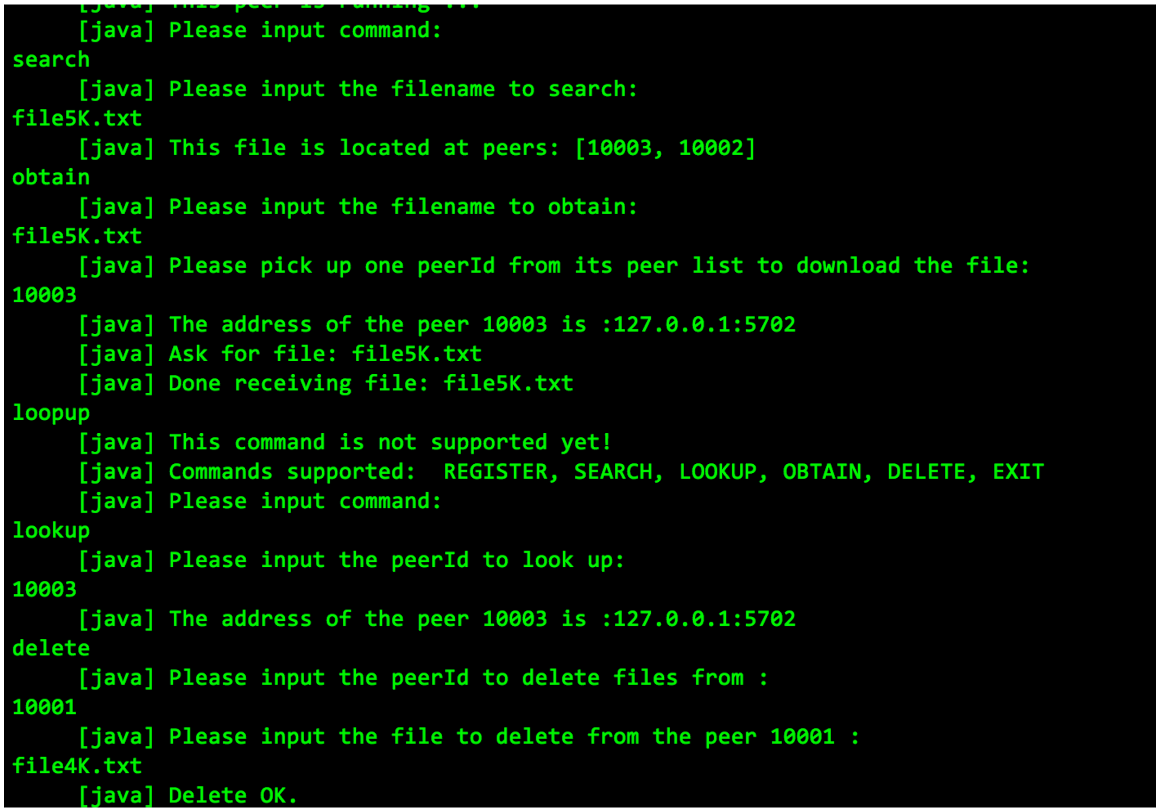
```
run-server:
     [java] Index Server is running...
     [java] Listening on port 5800.
     [java] The current file-peer registration table is:
     [java] file1K.txt: 10001
     [java] file2K.txt: 10001
     [java] file4K.txt: 10001
     [java] file3K.txt: 10001
     [java]
     [java] The current file-peer registration table is:
     [java] file6K.txt: 10002
     [java] file5K.txt: 10002
     [java] file1K.txt: 10001
     [java] file2K.txt: 10001
     [java] file4K.txt: 10001 10002
     [java] file3K.txt: 10001 10002
     [java] file7K.txt: 10002
     [java]
     [java] The current file-peer registration table is:
     [java] file10K.txt: 10003
     [java] file6K.txt: 10002
     [java] file5K.txt: 10003 10002
     [java] file1K.txt: 10001
     [java] file2K.txt: 10001
     [java] file4K.txt: 10001 10002
     [java] file9K.txt: 10003
     [java] file3K.txt: 10001 10002
     [java] file7K.txt: 10002
     [java] file8K.txt: 10003
     [java]
     [java] The current file-peer registration table is:
     [java] file10K.txt: 10003
     [java] file6K.txt: 10002
     [java] file5K.txt: 10003 10002
     [java] file1K.txt: 10001
     [java] file2K.txt: 10001
     [java] file1G.txt: 10004
     [java] file4K.txt: 10001 10002
     [java] file9K.txt: 10003
     [java] file3K.txt: 10001 10002
     [java] file7K.txt: 10002
```

Figure 6 Peers registration on startup

b. Commands:

Type commands in the console of any peer. As it is displayed in this figure, you can input the command as promoted.

i. Register: register
ii. Search file: (1) search; (2) filename
iii. Loop up peerId: (1) loopup; (2) peerId (get from the result of search file)
iv. Obtain file: (1) obtain; (2) filename; (3) peerId (get from the result of search file)
v. Delete file: (1) delete; (2) peerId; (3) filename



```
     [java] This peer is running ...
     [java] Please input command:
search
     [java] Please input the filename to search:
file5K.txt
     [java] This file is located at peers: [10003, 10002]
obtain
     [java] Please input the filename to obtain:
file5K.txt
     [java] Please pick up one peerId from its peer list to download the file:
10003
     [java] The address of the peer 10003 is :127.0.0.1:5702
     [java] Ask for file: file5K.txt
     [java] Done receiving file: file5K.txt
loopup
     [java] This command is not supported yet!
     [java] Commands supported:  REGISTER, SEARCH, LOOKUP, OBTAIN, DELETE, EXIT
     [java] Please input command:
lookup
     [java] Please input the peerId to look up:
10003
     [java] The address of the peer 10003 is :127.0.0.1:5702
delete
     [java] Please input the peerId to delete files from :
10001
     [java] Please input the file to delete from the peer 10001 :
file4K.txt
     [java] Delete OK.
```

Figure 7 Supported commands display