Cycle-1-Shell Scripting

0:UNIX commnds Date : 27/1/20

PROBLEM 0: UNIX COMMANDS

AIM

Write shell script to perform the following tasks.

Try out the following Unix commands(use manual and help features for support

1. echo, read
2. more, less
3. man
4. chmod, chown
5. cd, mkdir, pwd, ls, find
6. cat, mv, cp, rm
7. wc, cut, paste
8. head, tail, grep, expr
9. Redirections & Piping
10. useradd, usermod, userdel, passwd
11. tar

PROGRAM CODE:

echo , read	• Echo is to display line of text or strings that are passed as arguments. Syntax : Echo [option] [string] • Read is used to read from a file descriptor read. Syntax : Read text
more , less	• More is used to view the text file in command prompt displaying one screen at a time in case file is large. Syntax : more -d sample.txt • Less is used to read contents of text file in page per time. Syntax : sudo | less
man	• Man shows the manual of any command present in linux. Syntax : man chown

chmod, chown	• Chmod is to modify the access or permission of a user. Syntax : chmod 766 ex.txt

	• Chown is to change owner of a file. Syntax : sudo chown clave : mary example.txt
cd, mkdir, pwd, ls, find	• Cd changes current directory. Syntax : cd directory_name • Mkdir makes a directory if not already exist . Syntax : mkdir <directory name> • Pwd shows present directory. Syntax : pwd • Ls lists all file and folder in directory. Syntax : ls -l • Find tracks down file. Syntax : find . name * ones *
cat, mv, cp, rm	• Cat are generally used to concatenate files. Syntax : cat [OPTION]..[FILE] • Mv moves file or rename files. Syntax : mv [OPTION]source • Cp copies the files and directories from source to destination. Syntax : cp [OPTION] source • Rm removes files and directories. Syntax: rm [OPTION]…[FILE]
wc, cut, paste	• Wc is used to count the number of characters, words in a file. • Cut is used for cutting out sections for each line of files and writes result to standard output. • Paste is used to join files horizontally by outputting lines.
head, tail, grep, expr	• Head command prints lines from beginning of a file and tail prints lines from end of file. • Grep is used to search for the specified text in a file. • Expr evaluates a given expression and displays corresponding output.
Redirections, Piping	• Redirection is a feature when executes a command,we can change input or output devices. • Piping is a form of redirection ie used in linux to send output of one command for further processing.
useradd, usermod, userdel, passwd	• Useradd is used to create new accounts in linux. • Usermod used to modify existing accounts in linux. • Userdel is used to delete account in linux. • Passwd is used to assign password to local accounts of users.
tar	• Tar stands for tap to achieve which is used to tape drive back up command used by linux. Syntax : tar [OPTIONS] [ARCHIEVE-FILE] [FILE OD DIRECTORY TO BE ACHIEVED]

1:welcome message Date : 2/2/20

PROGRAM 1 :WELCOME MESSAGE

AIM :

Print a customized welcome message. Get the name of the user as input and attach the name to the welcome message.

PROGRAM CODE:

Welcome.sh	#!/bin/bash Read “Enter your name :” name Echo “Hello! $name. Welcome”

RESULT :

Output :-

Enter your name :

Gouri

Hello Gouri Welcome

2:greatest of 2 numbers Date : 6/2/20

PROGRAM 3 :GREATEST OF 2 NUMBERS

AIM:

Take 2 numbers as input and print the greater of the two.

PROGRAM CODE:

Greatest.sh	Echo “Enter first number” Read num1 Echo “Enter second number” Read num2 If [$num1 -gt $num2] Then Echo “Greatest number is” $num1 Else Echo “Greatest number is” $num2 fi

Result :

Enter first num:

20

Enter second num:

35

Greatest number is

35

3:Odd numbers Date : 9/1/20

PROGRAM 3 : ODD NUMBERS

AIM :

Print the first 20 odd numbers.

PROGRAM CODE:

ODD	Echo “First 20 odd numbers are: “ For ((x=1; x<40; x+=2)) Do echo $x; done

RESULT :

1

3

5

7

9

11

13

15

17

19

21

23

25

27

29

31

33

35

37

39

PROGRAM 4 : SUM OF 20 NUMBERS

AIM:

Store 20 numbers in an array and print their sum.

PROGRAM CODE:

SUM	arr=() sum=0 for((i=0;i<20;i++)) do read n arr += ($n) done echo “${arr[@]}” for i in ${arr[@]} do sum=`expr $sum + $i` done echo “sum=”.$sum

Result :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Sum=210

5:Creating text file Date : 16/2/20

PROGRAM 5 : CREATING TEXT FILE

AIM:

Create a text file with 20 lines of text.

PROGRAM CODE :

Text1	cat > text1.txt My dream In the night of my dreams There comes the goddess of love
	With a pleasant and smiling face She taught me how to love Through the wonder of human mind Which knows how to tackle with, hard ones with love She led me to the place of peace and harmony Where people loved and served one another No complaints! No regrets! Everything comfortable there She taught me the “value of life” Our life to be served for someone We to be live for someone I realized the world How it is, how it could be All because of human mind That can even capture the world Then the goddess disappeared I wake up to a new world Knowing to love and serve others..

RESULT :

File created text1.txt

6:Replacing strings Date : 20/2/20

Program 6 : REPLACING STRING

AIM:

Open the file created in program 5 and replace any string with another without using stream editor.

PROGRAM CODE :

Text1	Open the file by, Cat > text1.txt ex -s -c '%s/My dream /Angel of my dream/g | x' text1.txt

RESULT :

Angel of my dream

In the night of my dreams
There comes the goddess of love
With a pleasant and smiling face

She taught me how to love
Through the wonder of human mind
Which knows how to tackle with, hard ones with love

She led me to the place of peace and harmony
Where people loved and served one another
No complaints! No regrets!
Everything comfortable there

She taught me the "value of life"
Our life to be served for someone
We to be live for someone

I realized the world
How it is, how it could be
All because of human mind

That can even capture the world
Then the goddess disappeared
I wake up to a new world
Knowing to love and serve others..

PROGRAM 7 : PROTOCOLS AND DESCRIPTION

AIM :

Open the /etc/protocols file and copy the protocol number of the following protocols intoanother file named “favorite protocols” and format it in the same way as the original /etc/protocol file.

1. udp
2. idrp
3. skip
4. ipip

PROGRAM CODE:

Protocols.txt	> grep udp /etc/protocols > grep udp /etc/protocols >> downloads/new.txt >grep idrp /etc/protocols >grep idrp /etc/protocols >>downloads/new.txt >grep skip /etc/protocols >grep skip /etc/protocols >>downloads/new.txt >grep ipip /etc/protocols >grep ipip /etc/protocols >>downloads/new.txt >Cntrl + D

RESULT :

1. udp 17 UDP #user Datagram protocol
2. Udplite 136 UDPLITE #UDP – lite[RFC3828]
3. Idrp 45 IDRP #inter – domain routing protocol
4. Skip 57 SKIP #SKIP
5. Ipip 94 IPIP #ip – within – ip encapsulation protocol

PROGRAM 8 : USING AT AND BATCH

AIM :

Use "at" and "batch" to schedule tasks

PROGRAM CODE:

At	at at > 23: 07 at > mkdir gouri at > ctrl+d Job 10 at mon 10:11:00 2020
Batch	>batch at>touch mca042/new.txt >echo "hello world" > mca042/new.txt At > EOT Job 20 at 29 23:12:00 2020

RESULT

At 10:11 a new directory gouri is created.

At 23:12 new text with content hello world is created.

9:CRON commands Date : 2/3/'20

PROGRAM 9 : CRON COMMAND

AIM :

Use cron to schedule task.

PROGRAM CODE

CRON	>crontab –e Choose : 1 36 22 29 1 1-5 touch mca042/gouri.txt && echo "this is an example" > mca042/gouri.txt

Result:

A new file gouri.txt is created with content "this is an example" at 22:36 on every Monday to Friday on 29 th January.

Program 10: Unix Mail

Aim:

Set up UNIX mail and use mail to send and receive mails to and from users using shell script.

PROGRAM CODE:

Mail	mail -s "ADM" gourivijayan14@gmail.com cc : Good morning to all Welcome to ADM class Cntrl + D

RESULT:

The message with subject ADM with corresponding contents are received on the respective email.

Cycle-2-Version Control using git

1:Git repository Date : 9/3/20

AIM:

1. Create team account.
2. Create empty repository in any git remote repository service and add collaborators.
3. Leader must create the first commit.
4. All members must clone the remote repository.
5. Each member must create a feature branch each and add features to them(any mod)
6. Commit changes to branches.
7. Push the branches.
8. View Graph.
9. Leader must make changes to the master.
10. All member must rebase their branches to the position of latest commit in master.
11. Merge all branches to master.
12. Cherry pick commits from each branch created earlier.
13. View Status.
14. View History.
15. Delete all branches.

PROGRAM CODE :

Check the status of files in the index verses the working directory for the git repository by using the following command

> Git status

Use the below commands to set the the git configurations in terminal.

> git config --global user.name “maggammag”

> git config --global user.email xxxyyy999@gmail.com

Create a clone or copy of the target repository

> git clone https://github.com/maggammag/Project1.git.

In order to list all the files that are currently available in the repository , use the command

> ls

Pull changes made by others on remote repository to our local repository , use the command

> git pull –rebase origin master

Create a new file named example1.txt by,

> nano example1.txt

Now add this file to the staging area. For this use the command

> git add example1.txt

Commit these changes by using

> git commit –m “First commit” example1.txt

This will add the file and with a commit message first commit so that we can search it later.

Push these files to the repository using the command

> Git push –u origin master

Create a branch for adding our own features from the master branch use the command

> Git checkout –b gouri

> Git push –u origin gouri

Merge the feature branch to the master branch by

> Git checkout master

> Git merge gouri

In order to cherry-pick use the command as follows

> Git log –oneline

> Git cherry-pick ef7200

To delete a feature branch , use the command :

> git branch -d gouri /*deletes gouri branch*/

RESULT :

Output of branch gouri by the git commands.

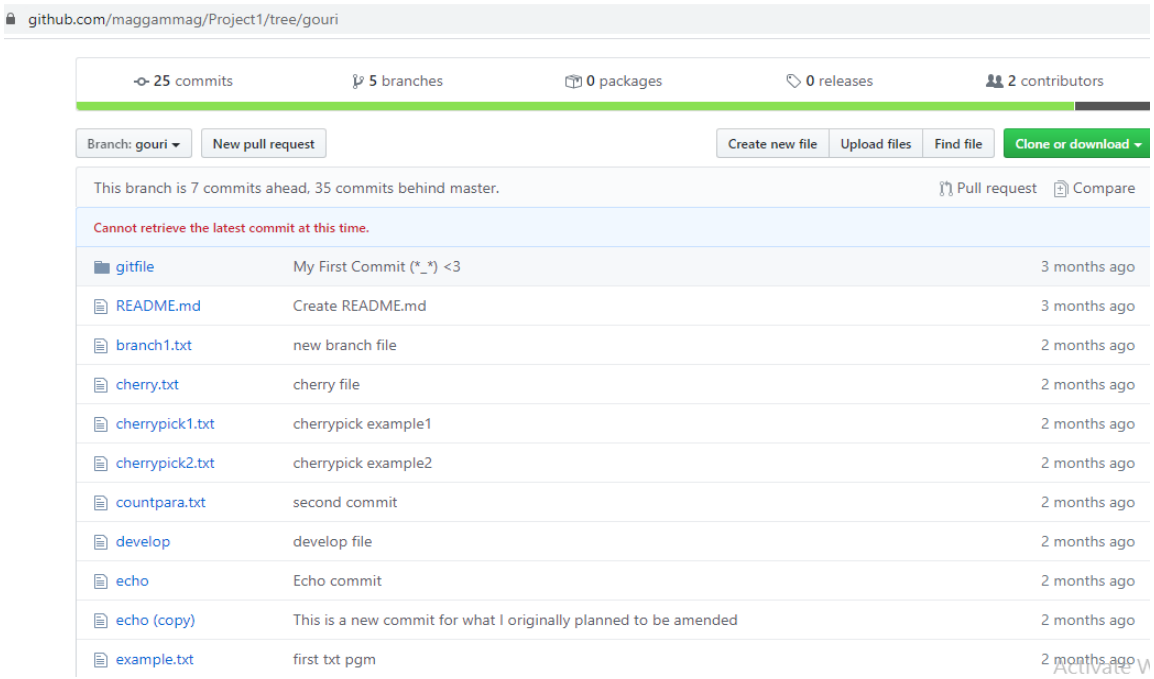

Figure 1 :branch gouri with files pushed.

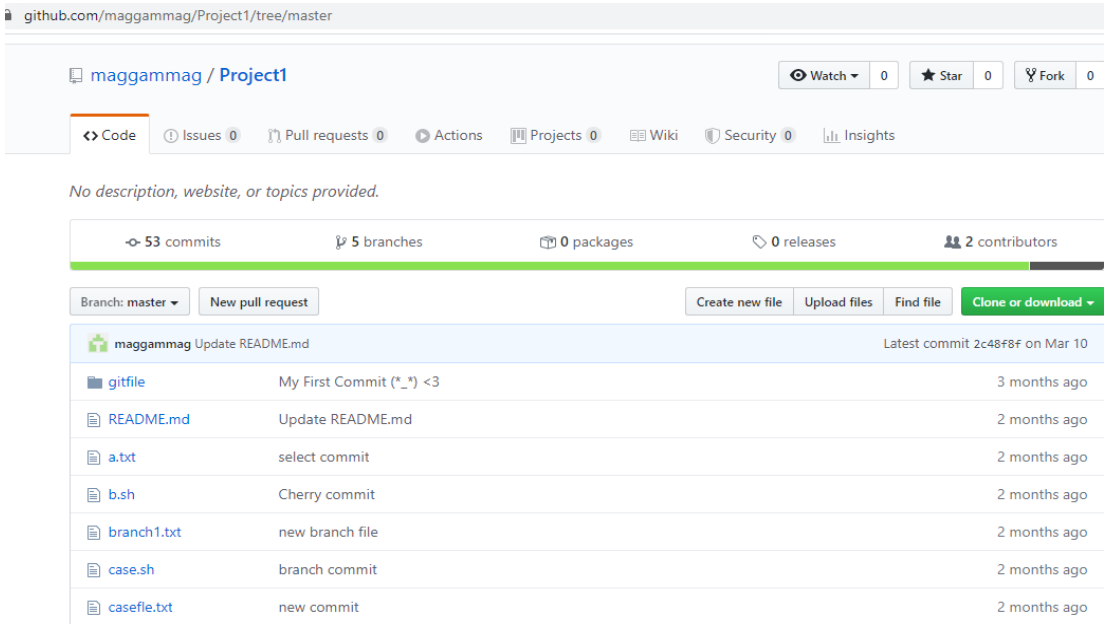

Figure2: Master branch

Cycle-3-Network Programming In Java

1:TCP client-server Date : 12/3/'20

Program 1 :TCP Client Server Communication

AIM:

Implement Bidirectional Client-Server communication using TCP.

PROGRAM CODE:

Server.java

```
// Server2 class that
// receives data and sends data

import java.io.*;
import java.net.*;

class Server2 {

	public static void main(String args[])
		throws Exception
	{

		// Create server Socket
		ServerSocket ss = new ServerSocket(888);

		// connect it to client socket
		Socket s = ss.accept();
		System.out.println("Connection established");

		// to send data to the client
		PrintStream ps
			= new PrintStream(s.getOutputStream());

		// to read data coming from the client
		BufferedReader br
			= new BufferedReader(
				new InputStreamReader(
					s.getInputStream()));

		// to read data from the keyboard
		BufferedReader kb
			= new BufferedReader(
```

	new InputStreamReader(System.in));
	
	// server executes continuously
	while (true) {
	
	String str, str1;
	
	// repeat as long as the client
	// does not send a null string
	
	// read from client
	while ((str = br.readLine()) != null) {
	System.out.println(str);
	str1 = kb.readLine();
	
	// send to client
	ps.println(str1);
	}
	
	// close connection
	ps.close();
	br.close();
	kb.close();
	ss.close();
	s.close();
	
	// terminate application
	System.exit(0);
	
	} // end of while
	}
	}

Client.java	// Client2 class that
	// sends data and receives also
	
	import java.io.*;
	import java.net.*;
	
	class Client2 {
	
	public static void main(String args[])
	throws Exception
	{
	
	// Create client socket
	Socket s = new Socket("localhost", 888);

```
                // to send data to the server
                DataOutputStream dos
                        = new DataOutputStream(
                                s.getOutputStream());

                // to read data coming from the server
                BufferedReader br
                        = new BufferedReader(
                                new InputStreamReader(
                                        s.getInputStream()));

                // to read data from the keyboard
                BufferedReader kb
                        = new BufferedReader(
                                new InputStreamReader(System.in));
                String str, str1;

                // repeat as long as exit
                // is not typed at client
                while (!(str = kb.readLine()).equals("exit")) {

                        // send to the server
                        dos.writeBytes(str + "\n");

                        // receive from the server
                        str1 = br.readLine();

                        System.out.println(str1);
                }

                // close connection.
                dos.close();
                br.close();
                kb.close();
                s.close();
        }
}
```

RESULT:

TCP Client and Server communication.

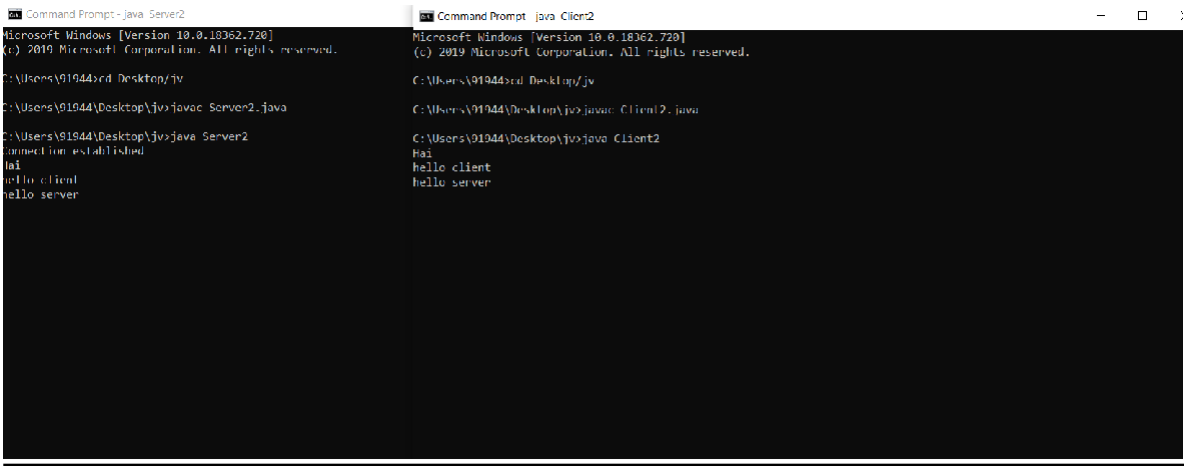

Figure 1: Communication between local client and server

2:TCP echo server Date : 16/3/'20

PROGRAM 2 : Echo server with TCP

AIM:

Implement Echo Server using TCP

PROGRAM CODE:

Echoserver.java	import java.io.*; import java.net.*; public class EchoServer { public static void main(String args[]) throws Exception { try { int Port; BufferedReader Buf =new BufferedReader(new InputStreamReader(System.in));

	System.out.print(" Enter the Port Address : ");
	Port=Integer.parseInt(Buf.readLine());
	ServerSocket sok =new ServerSocket(Port);
	System.out.println(" Server is Ready To Receive a Message. ");
	System.out.println(" Waiting ..... ");
	Socket so=sok.accept();
	if(so.isConnected()==true)
	System.out.println(" Client Socket is Connected Succecfully. ");
	InputStream in=so.getInputStream();
	OutputStream ou=so.getOutputStream();
	PrintWriter pr=new PrintWriter(ou);
	BufferedReader buf=new BufferedReader(new InputStreamReader(in));
	String str=buf.readLine();
	System.out.println(" Message Received From Client : " + str);
	System.out.println(" This Message is Forwarded To Client. ");
	pr.println(str);
	pr.flush();
	}
	catch(Exception e)
	{
	System.out.println(" Error : " + e.getMessage());
	}
	}
	}

Echoclient.java	import java.io.*;
	import java.net.*;
	
	public class EchoClient
	{
	public static void main(String args[]) throws Exception
	{
	try {
	int Port;
	BufferedReader Buf =new BufferedReader(new InputStreamReader(System.in));
	System.out.print(" Enter the Port Address : ");
	Port=Integer.parseInt(Buf.readLine());
	Socket sok=new Socket("localhost",Port);
	if(sok.isConnected()==true)
	System.out.println(" Server Socket is Connected Succecfully. ");
	InputStream in=sok.getInputStream();
	OutputStream ou=sok.getOutputStream();
	PrintWriter pr=new PrintWriter(ou);

```
BufferedReader buf1=new BufferedReader(new
InputStreamReader(System.in));
BufferedReader buf2=new BufferedReader(new
InputStreamReader(in));
String str1,str2;
System.out.print(" Enter the Message : ");
str1=buf1.readLine();
pr.println(str1);
pr.flush();
System.out.println(" Message Send Successfully. ");
str2=buf2.readLine();
System.out.println(" Message From Server : " + str2);
    }
  catch(Exception e)
  {
   System.out.println(" Error : " + e.getMessage());
  }
}
}
```

RESULT:

Communication between echo server and client via TCP

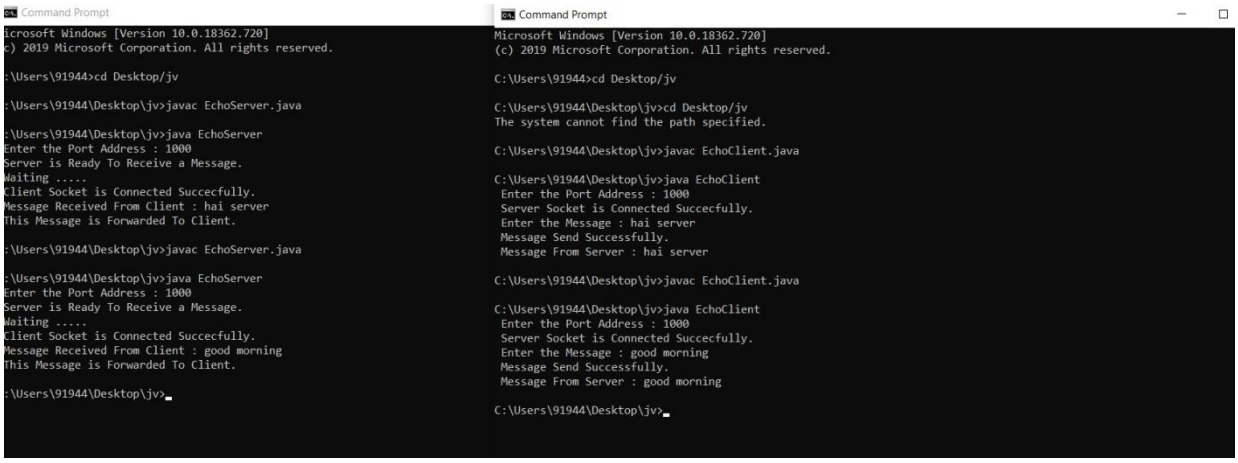

Figure 1 : TCP echo server and client

PROGRAM 3 : UDP Chat Server

AIM:

Implement Chat Server using UDP

PROGRAM CODE:

UDPClent.java

```
import java.io.*;
import java.net.*;
class UDPClient
{
public static DatagramSocket clientsocket;
public static DatagramPacket dp;
public static BufferedReader dis;
public static InetAddress ia;
public static byte buf[] = new byte[1024];
public static int cport = 789, sport = 790;
public static void main(String[] a) throws IOException
{
clientsocket = new DatagramSocket(cport);
dp = new DatagramPacket(buf, buf.length);
dis = new BufferedReader(new
InputStreamReader(System.in));
ia = InetAddress.getLocalHost();
System.out.println("Client is Running... Type 'STOP'to Quit");
while(true)
{
String str = new String(dis.readLine());
buf = str.getBytes();
if(str.equals("STOP"))
{
System.out.println("Terminated...");
clientsocket.send(new
DatagramPacket(buf,str.length(), ia,
sport));
break;
}
clientsocket.send(new DatagramPacket(buf,
str.length(), ia, sport));
clientsocket.receive(dp);
String str2 = new String(dp.getData(), 0,
dp.getLength());
System.out.println("Server: " + str2);
```

	} } }

UDPServer.java	import java.io.*;
	import java.net.*;
	class UDPServer
	{
	public static DatagramSocket serversocket;
	public static DatagramPacket dp;
	public static BufferedReader dis;
	public static InetAddress ia;
	public static byte buf[] = new byte[1024];
	public static int cport = 789,sport=790;
	public static void main(String[] a) throws IOException
	{
	serversocket = new DatagramSocket(sport);
	dp = new DatagramPacket(buf,buf.length);
	dis = new BufferedReader
	(new InputStreamReader(System.in));
	ia = InetAddress.getLocalHost();
	System.out.println("Server is Running...");
	while(true)
	{
	serversocket.receive(dp);
	String str = new String(dp.getData(), 0,
	dp.getLength());
	if(str.equals("STOP"))
	{
	System.out.println("Terminated...");
	break;
	}
	System.out.println("Client: " + str);
	String str1 = new String(dis.readLine());
	buf = str1.getBytes();
	serversocket.send(new
	DatagramPacket(buf,str1.length(), ia, cport));
	}
	}
	}

RESULT :

Figure explains chat communication between UDP client and server.

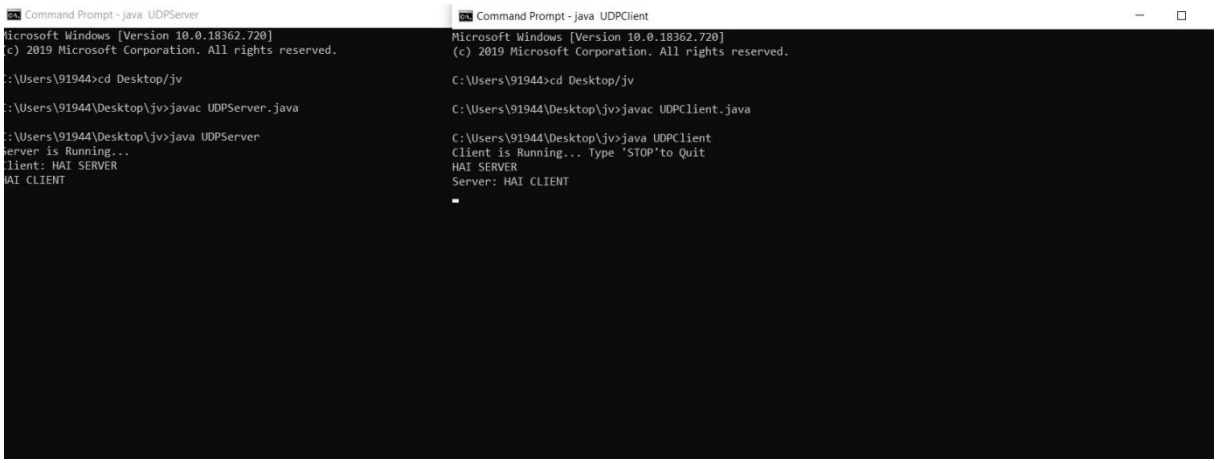

Figure 1 : UDP server and client communication