## 

##### Write two programs (server/client) and establish a socket to communicate.

**Objectives:**

1. To learn about fundamentals of IPC through C socket programming.
2. Learn and understand the OS intraction with socket programming.
3. Use of system call and IPC mechanism to write effective application programs.
4. To know the port numbersing and process relation.
5. To knows the iterative and concurrent server concept.

**Theory:**

JAVA SOCKET PROGRAMMING

A very basic one-way Client and Server setup where a Client connects, sends messages to server and the server shows them using socket connection. Java API networking package (java.net) takes care of all of that, making network programming very easy for programmers.

CLIENT SIDE PROGRAMMING:

Establish a Socket Connection

To connect to other machine we need a socket connection. A socket connection means the two machines have information about each other’s network location (IP Address) and TCP port.The java.net.Socket class represents a Socket. To open a socket:

Socket socket = new Socket(“127.0.0.1”, 5000)

* + First argument – IP address of Server. (127.0.0.1 is the IP address of localhost, where code will run on single stand-alone machine).
  + Second argument – TCP Port. (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535)

To communicate over a socket connection, streams are used to both input and output the data. Closing the connection

The socket connection is closed explicitly once the message to server is sent.

SERVER SIDE PROGRAMMING:

Establish a Socket Connection

To write a server application two sockets are needed.

* A ServerSocket which waits for the client requests (when a client makes a new Socket())
* A plain old Socket socket to use for communication with the client. getOutputStream() method is used to send the output through the socket. Close the Connection

After finishing, it is important to close the connection by closing the socket as well as input/output streams.

#### Data Dictionary:

|  |  |  |  |
| --- | --- | --- | --- |
| Sr Number | Variable/Function | Datatype | Use |
| 1 | Ss | ServerSocket | Create a socket for server side communication. |
| 2 | S | Socket | Socket is created. |
| 3 | Dos | DataOutputStream | Output Stream. |
| 4 | Dis | DataInputStream | Input Stream. |
| 5 | Str | String | String to display message from clients. |

**Program:**

SERVER:

import java.net.\*; import java.io.\*; class Server1

{

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

{

ServerSocket ss=new ServerSocket(5050);//5050 is port no. System.out.println("Server is Waiting ");

Socket s=ss.accept();//waiting for client

DataOutputStream dos=new DataOutputStream(s.getOutputStream()); DataInputStream dis=new DataInputStream(s.getInputStream()); String str="Welcomes you are connected \n";

dos.writeUTF(str); //sends msg to client str=dis.readUTF(); //reads msg send by client System.out.println("From client"+" "+str); ss.close();

s.close();

dos.close();

dis.close();

}

}

CLIENT:

import java.net.\*; import java.io.\*; class Client1

{

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

{

Socket s=new Socket("localhost",5050);

DataOutputStream dos=new DataOutputStream(s.getOutputStream()); DataInputStream dis=new DataInputStream(s.getInputStream()); String str=dis.readUTF(); //receives msg send by server System.out.println("From server"+" "+str);

str="Thank u for connecting"; dos.writeUTF(str); //writes to server s.close();

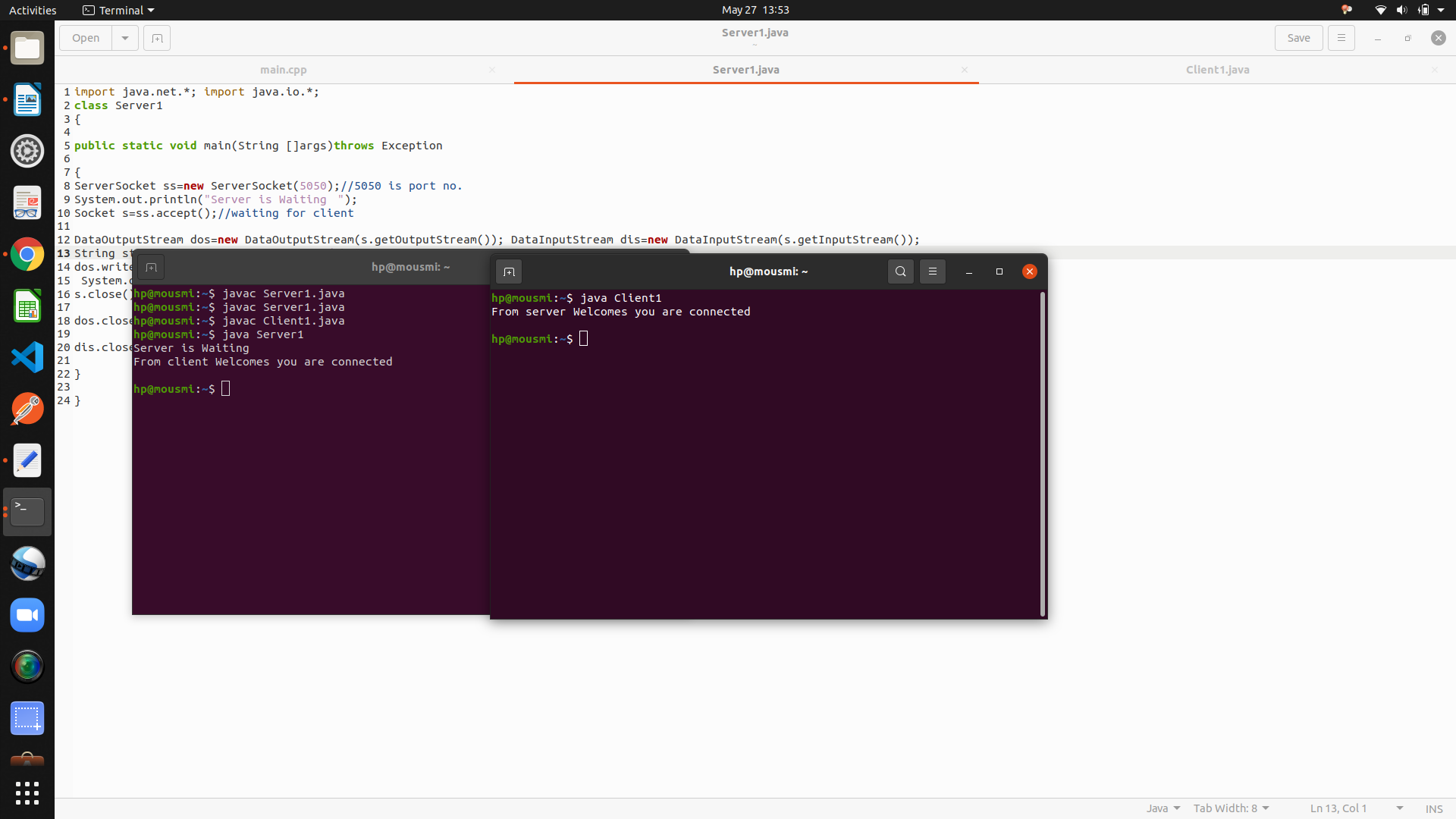
dos.close();

dis.close();

}

}

#### OUTPUT:

****

**Conclusion:**

Java can be used to establish communication between two programs on remote or same machine using sockets and system calls.

#### References:

<http://www.prasannatech.net/2008/07/socket-programming-tutorial.html>