

1. Write a Program to design Chat Application using Client and Server Approach.

**Aim:** To perform chat application using client and server approach

**Program:**

**i.Client Code:**

```
import java.io.*;

import java.net.*;

try{

    String ip = IP.getText();

    Integer port = Integer.parseInt(Port.getText());

    Socket s = new Socket(ip,port);

    DataInputStream dis = new DataInputStream(s.getInputStream());

    DataOutputStream dos = new DataOutputStream(s.getOutputStream());

    String str = Cmessage.getText();

    dos.writeUTF(str);

    String newStr = dis.readUTF();

    Cmessage.append("\n"+newStr);

    s.close();

}

catch(Exception e){

    e.printStackTrace();

}
```

**ii.Server Code:**

```
import java.io.*;

import java.net.*;

public class Server extends javax.swing.JFrame implements Runnable {

public void run(){

    try{
```

```

Integer port = Integer.parseInt(SPort.getText());

ServerSocket ss = new ServerSocket(port);

while(true){

    Socket s = ss.accept();

    DataInputStream dis = new DataInputStream(s.getInputStream());

    DataOutputStream dos = new DataOutputStream(s.getOutputStream());

    String str = dis.readUTF();

    Smessage.append("\nMessage Received
from/" + s.getInetAddress().toString() + ":" + str + "\nSending response...\n");

    dos.writeUTF("\nServer Response: " + str);

    s.close();

}

}

catch(Exception e){

    e.printStackTrace();

}

}

private void ReceiveActionPerformed(java.awt.event.ActionEvent evt) {

    Thread t = new Thread(this, "t1");

    t.start();

    Smessage.append("Server listening...\n");

    Receive.setEnabled(false); // TODO add your handling code here:

}

```

## Output: Execution Steps

### a. Executing from same system:

The screenshot shows two windows side-by-side. The 'Client' window on the left has a 'Port' field with '5051', an 'IP Address' field with '10.2.4.19', and a 'Message' box containing 'This is first program' and 'Server Response: This is first program'. A 'Submit' button is at the bottom. The 'Server' window on the right has a 'Port' field with '5051' and a 'Message' box showing 'Server listening...', 'Message Received from//10.2.4.19:This is first program', and 'Sending response...'. A 'StartServer' button is at the bottom.

### b. Executing from remote system:

The screenshot shows two windows side-by-side. The 'CLIENT' window on the left has a 'PORT' field with '5051', an 'IP ADDRESS' field with '10.2.4.19', and a 'MESSAGE' box containing 'HELLO THIS IS A CHAT APPLICATION' and 'Server Response: HELLO THIS IS A CHAT APPLICATION'. A 'SUBMIT' button is at the bottom. The 'Server' window on the right has a 'Port' field with '5051' and a 'Message' box showing 'Server listening...', 'Message Received from//10.2.4.20:HELLO THIS IS A CHAT APPLICATION', and 'Sending response...'. A 'StartServer' button is at the bottom.

c. Executing from local machine(Single Framework):

The screenshot displays a 'Chat Application' window with two distinct sections. The top section, representing the server, includes a 'Port' input field with the value '5051' and a 'Start' button. A message box below shows the sequence: 'Server listening...', 'Message Received from//10.2.4.19:This is DS Lab', and 'Sending response...'. The bottom section, representing the client, features 'Port' (5051) and 'IP Address' (10.2.4.19) input fields, a 'CMessage' input field containing 'This is DS Lab', and a 'Send' button. A message box in this section shows 'This is DS Lab' and 'Server Response: This is DS Lab'.

Section	Field	Value
Server Interface	Port	5051
	Message	Server listening... Message Received from//10.2.4.19:This is DS Lab Sending response...
Client Interface	Port	5051
	IP Address	10.2.4.19
	CMessage	This is DS Lab

2. Write a Program to demonstrate Domain Name Server.

**Aim:** To demonstrate Name Server using Domain Name Server.

**Program:**

**i.Client Code:**

```
import java.io.*;

import java.net.*;

private void SubmitActionPerformed(java.awt.event.ActionEvent evt) {

    try{

        String ip =IP.getText();

        int port =Integer.parseInt(CPort.getText());

        Socket s=new Socket(ip,port);

        DataInputStream dis=new DataInputStream(s.getInputStream());

        DataOutputStream dos=new DataOutputStream(s.getOutputStream());

        String domain=DName.getText();

        dos.writeUTF(domain);

        String response=dis.readUTF();

        Message.append(" "+response+"\n");

        s.close();

    }

    catch(Exception e)

    {

        e.printStackTrace();

    }

}
```

**ii.Server Code:**

```
import java.io.*;

import java.net.*;

import java.util.*;
```

```

public class dnsserver extends javax.swing.JFrame implements Runnable {

    public void run()

    {

        try{

            int sPort=Integer.parseInt(SPort.getText());

            ServerSocket ss=new ServerSocket(sPort);

            //binded server socket - listens for connections

            while(true)

            {

                Socket s=ss.accept();

                //client's request has come; connection is established

                /* Getting I/O Streams */

                DataInputStream dis=new DataInputStream(s.getInputStream());

                DataOutputStream dos=new DataOutputStream(s.getOutputStream());

                //Get the request

                String req=dis.readUTF();

                Smessage.append(" " +s.getInetAddress().toString()+"/");

                //displaying from which client what domain request is coming

                /* READING FROM FILE */

                try{

                    BufferedReader br=new BufferedReader(new InputStreamReader(new
FileInputStream("DNS.txt")));

                    String flInput=br.readLine();

                    int flag=0;

                    while(flInput!=null)

                    {

                        StringTokenizer stk=new StringTokenizer(flInput); //tokens

                        String dname=stk.nextToken();

```

```

        String dIP=stk.nextToken();

        if(req.equals(dname))
        {
            dos.writeUTF(dname+" " +"IP is "+"/"+dIP+"/"+"\\n");

            flag=1;
        }

        flInput=br.readLine();
    }

    if(flag==0)

        dos.writeUTF(req+"/NOT FOUND");
    }

    catch(Exception e)
    {
        e.printStackTrace();
    }
}

catch(Exception e)
{
    e.printStackTrace();
}
}

private void StartActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    Thread t=new Thread(this,"ns");

    t.start();

    Start.setEnabled(false);
}

```

```
Smessage.append("Server is listening...\n");  
  
}
```

### Output:

#### a. DNS using Two Frameworks

The screenshot shows two side-by-side windows: SERVER and CLIENT.

**SERVER Window:**

- Port: 5051
- Message: Server is listening... /127.0.0.1/
- StartServer button

**CLIENT Window:**

- Port: 5051
- IP Address: localhost
- Domain Name: www.google.com
- Response: www.google.com IP is /100.2.3.4/
- Submit button

The screenshot shows two side-by-side windows: SERVER and CLIENT.

**SERVER Window:**

- Port: 5051
- Message: Server is listening... /127.0.0.1/ /127.0.0.1/
- StartServer button

**CLIENT Window:**

- Port: 5051
- IP Address: localhost
- Domain Name: www.osmania.com
- Response: www.google.com IP is /100.2.3.4/  
www.osmania.com/NOT FOUND
- Submit button



b. DNS using Single Framework

The screenshot shows a web-based DNS application interface. The window has a title bar with standard minimize, maximize, and close buttons. The main content area is divided into two sections. The top section is titled 'DNS' and contains a 'Port' input field with the value '5051', a 'StartServer' button, and a 'Message' box displaying 'Server is listening... /127.0.0.1/'. The bottom section contains a 'Port' input field with '5051', an 'IP' input field with 'localhost', a 'Domain' input field with 'www.amazon.com', a 'Submit' button, and a 'Response' box displaying 'The IP address of www.amazon.com is /200.5.8.4/'.

Input File:

The screenshot shows a Notepad window titled 'DNS - Notepad'. The window contains a text file with the following content:   
www.google.com 100.2.3.4  
www.amazon.com 200.5.8.4  
www.osmaniac.com 120.2.5.9  
The status bar at the bottom indicates 'Ln 4, Col 1', '100%', 'Windows (CRLF)', and 'UTF-8'.

3. Write a Program to demonstrate Chat Server bulletin.

**Aim: To perform Chat Server Bulletin**

**Program:**

**Client Code:**

```
private void SubmitActionPerformed(java.awt.event.ActionEvent evt) {  
  
    try{  
  
        String ip = IP.getText();  
  
        Integer port = Integer.parseInt(CPort.getText());  
  
        Socket s = new Socket(ip,port);  
  
        DataInputStream dis = new DataInputStream(s.getInputStream());  
  
        DataOutputStream dos = new DataOutputStream(s.getOutputStream());  
  
        String st = Username.getText();  
  
        dos.writeUTF(st);  
  
        String str = CMessage.getText();  
  
        dos.writeUTF(str);  
  
        String newStr = dis.readUTF();  
  
        CMessage.append("\n"+newStr);  
  
        s.close();  
  
    }  
  
    catch(Exception e){  
  
        e.printStackTrace();  
  
    }    // TODO add your handling code here:  
  
}
```

**Server Code:**

```
public class Server extends javax.swing.JFrame implements Runnable {  
  
    public void run(){  
  
        try{
```

```

Integer port = Integer.parseInt(SPort.getText());

ServerSocket ss = new ServerSocket(port);

while(true){

    Socket s = ss.accept();

    DataInputStream dis = new DataInputStream(s.getInputStream());

    DataOutputStream dos = new DataOutputStream(s.getOutputStream());

    String st = dis.readUTF();

    String str = dis.readUTF();

    SMessage.append("\nThe Username is : "+st+"\n");

    SMessage.append("\nMessage Received
from/"+s.getInetAddress().toString()+":\n"+str+"\nSending response...\n");

    dos.writeUTF("\nServer Response: "+str);

    s.close();

}

}

catch(Exception e){

    e.printStackTrace();

}

}

private void StartActionPerformed(java.awt.event.ActionEvent evt) {

    Thread t = new Thread(this,"t1");

    t.start();

    SMessage.append("Server listening...\n");

    Start.setEnabled(false);

    // TODO add your handling code here:

}

```

## **Output:**

The image shows two side-by-side application windows. The left window, titled 'Server', has a 'Port' field set to '5051' and a 'Message' box containing the text: 'Server listening...', 'The Username is : Student', 'Message Received from 127.0.0.1:', 'Hello', and 'Sending response...'. A 'Start' button is at the bottom. The right window, titled 'Client', has a 'Port' field set to '5051', an 'IP Address' field set to 'localhost', and a 'Username' field set to 'Student'. Its 'Message' box shows 'Hello' and 'Server Response: Hello'. A 'Submit' button is at the bottom.

Field	Server Value	Client Value
Port	5051	5051
IP Address		localhost
Username		Student

**Server Message Log:**

- Server listening...
- The Username is : Student
- Message Received from 127.0.0.1:
- Hello
- Sending response...

**Client Message Log:**

- Hello
- Server Response: Hello

#### 4. Write a Program to perform FTP Upload.

**Aim:** To perform FTP Upload.

**Program:**

**FtpUpload:**

```
import java.io.*;

import java.util.*;

import org.apache.commons.net.ftp.*;

import org.apache.commons.net.ftp.FTPClient;

import org.apache.commons.net.ftp.FTPReply;

import javax.swing.*; //will have to search all classes of swing package. takes time
import javax.swing.SwingUtilities; //saves search. immediate reference
import javax.swing.filechooser.*; //for file dialog for upload/download

public class ftpUpload extends JFrame {

    private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {

        try{

            String IP=txtIP.getText(); //GET IP

            FTPClient ftpc=new FTPClient(); //CREATE CLIENT

            txtAC.append("Establisg connection to the server ["+IP+"]...\n");

            ftpc.connect(IP); //CONNECT WITH SERVER

            int reply=ftpc.getReplyCode();

            //to check the status of the connection

            //    Positive Completion reply: the requested action has been successfully    completed.
            A new request may be initiated.

            if(FTPReply.isPositiveCompletion(reply))

                txtAC.append("Connection established with server ["+IP+"]\n"+" "+reply);//220
                Service ready for new user.

            else

                txtAC.append("Connection failed with the server ["+IP+"]\n");
```

```

String uname=txtUN.getText(); //GET USERNAME

String pwd=txtPW.getText(); //GET PASSWORD

txtAC.append("Logging into the server ["+IP+"]\n");

if(ftpc.login(uname,pwd))

    txtAC.append("Successfully logged into ["+uname+"] at ["+IP+"]\n");

else

    txtAC.append("Unable to login to ["+uname+"] at ["+IP+"]\n");

//txtAC.append("Disconnecting from the server...\n");

ftpc.disconnect();

//if client is idle for long then server disconnects and other operations fail. That is why disconnected
here.

    }

catch(Exception e)

{

    e.printStackTrace();

}

// TODO add your handling code here:

}

private void btnuploadActionPerformed(java.awt.event.ActionEvent evt) {

    String fileName = "";

    String fileAbsName = "";

    JFileChooser fc=new JFileChooser();//JFileChooser is a easy and an effective way to prompt the
user to choose a file or a directory

    int returnVal = fc.showOpenDialog(ftpUpload.this);

    if (returnVal == JFileChooser.APPROVE_OPTION) //Approve option: returns yes or ok

    {

        File file = fc.getSelectedFile();

        fileAbsName = file.getAbsolutePath();

```

```

        fileName = file.getName();
    }

    try
    {
        String ServerIP = txtIP.getText();

        FTPClient f = new FTPClient();

        f.connect(ServerIP);

        String user = txtUN.getText();

        String passwd = txtPW.getText();

        f.login(user, passwd);

        File firstLocalFile = new File(fileAbsName);

        String firstRemoteFile = fileName;

        InputStream inputStream = new FileInputStream(firstLocalFile);

        boolean done = f.storeFile(firstRemoteFile, inputStream);

        inputStream.close();

        if (done)

            txtAC.append("File "+ fileName +" is uploaded successfully."+fileAbsName);

        else

            txtAC.append("File "+ fileName +" cannot be uploaded.");

        f.disconnect();
    }

    catch (Exception ex)

    {

        ex.printStackTrace();

    }

    // TODO add your handling code here:

}

```

```

public static void main(String args[]) {

    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {

            new ftpc1().setVisible(true);

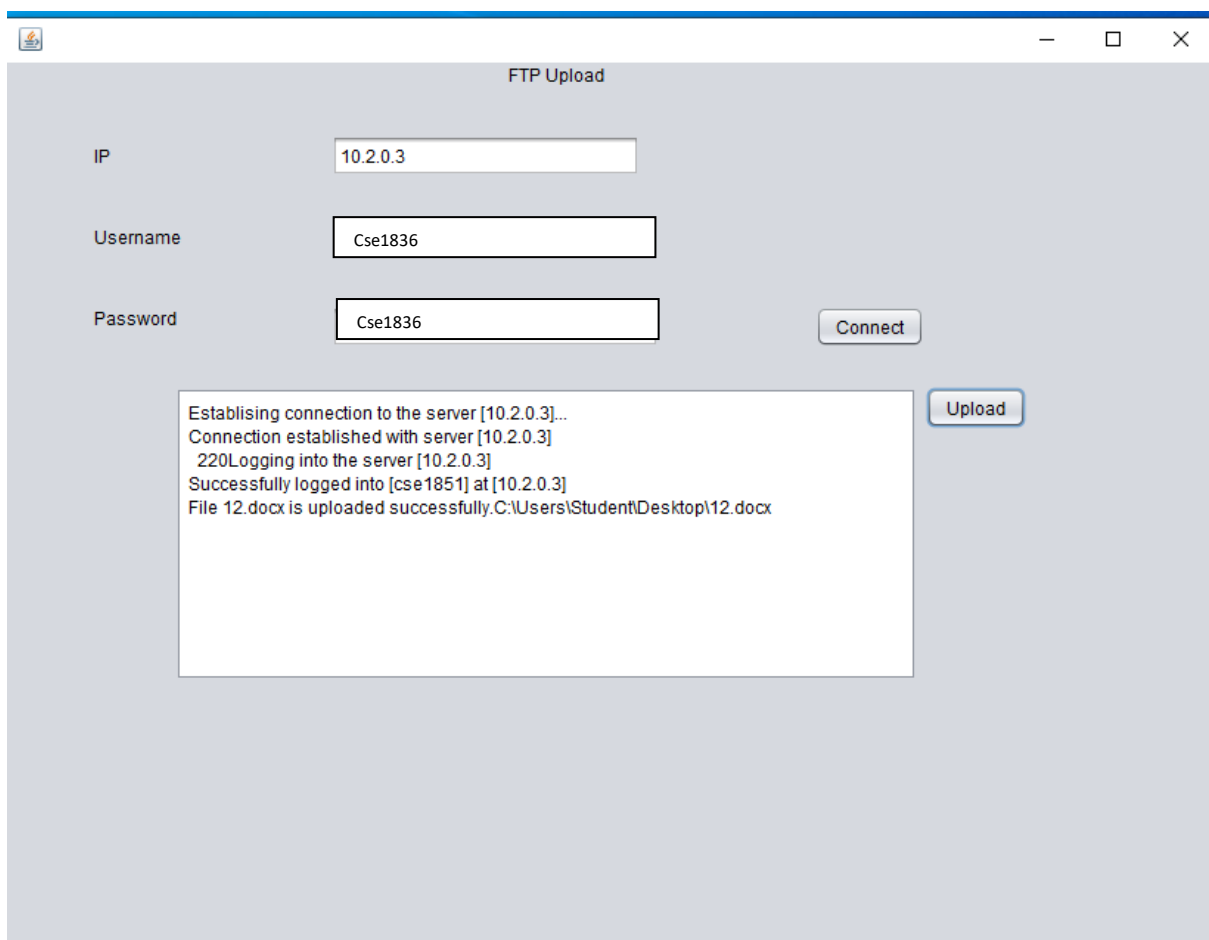
        }

    });

}

```

**Output:**





## 5. Write a Program to perform FTP Download.

### **Aim: To perform FTP Download**

### **Program:**

```
import java.io.*;

import java.util.*;

import org.apache.commons.net.ftp.*;

import org.apache.commons.net.ftp.FTPClient;

import org.apache.commons.net.ftp.FTPReply;

import javax.swing.*; //will have to search all classes of swing package. takes time

import javax.swing.SwingUtilities; //saves search. immediate reference

import javax.swing.filechooser.*; //for file dialog for upload/download

public class ftpDownload extends javax.swing.JFrame {

    private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {

        try{

            String IP=txtIP.getText(); //GET IP

            FTPClient ftpc=new FTPClient(); //CREATE CLIENT

            txtAC.append("Establishing connection to the server ["+IP+"]...\n");

            ftpc.connect(IP); //CONNECT WITH SERVER

            int reply=ftpc.getReplyCode();

            //to check the status of the connection

            //      Positive Completion reply: the requested action has been successfully completed.
            //      A new request may be initiated.

            if(FTPReply.isPositiveCompletion(reply))

                txtAC.append("Connection established with server ["+IP+"]\n"+" "+reply);//220
                Service ready for new user.

            else

                txtAC.append("Connection failed with the server ["+IP+"]\n");

            String uname=txtUN.getText(); //GET USERNAME
```

```

String pwd=txtPW.getText(); //GET PASSWORD

txtAC.append("Logging into the server ["+IP+"]\n");

if(ftpc.login(uname,pwd))

    txtAC.append("Successfully logged into ["+uname+"] at ["+IP+"]\n");

else

    txtAC.append("Unable to login to ["+uname+"] at ["+IP+"]\n");

txtAC.append("Disconnecting from the server...\n");

ftpc.disconnect();

```

//if client is idle for long then server disconnects and other operations fail. That is why disconnected here.

```

    }

    catch(Exception e)

    {

        e.printStackTrace();

    }

    // TODO add your handling code here:

}

private void btndownloadActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    try{

        String IP=txtIP.getText();

        String uname=txtUN.getText();

        String pwd=txtPW.getText();

        FTPClient ftpc=new FTPClient();

        ftpc.connect(IP);

        int reply=ftpc.getReplyCode();

        if(FTPReply.isPositiveCompletion(reply))

```

```

        txtAC.append("Connection established with server ["+IP+"]\n");
    else
        txtAC.append("Connection failed with server ["+IP+"]\n");
    if(ftpc.login(uname, pwd))
    {
        txtAC.append("Successfully logged into ["+uname+"] at ["+IP+"]\n");
        txtIP.enableInputMethods(false);
        /*Since we have successfully logged in, disable input on these fields*/
        txtUN.enableInputMethods(false);
        txtPW.enableInputMethods(false);
        String selectedFile=cb1.getSelectedItem().toString();
        //get name of the file chosen for download
        File dFileName=new File(selectedFile);
        //to get the file object for reading and writing
        OutputStream os=new BufferedOutputStream(new FileOutputStream(dFileName)); //since
        reading is done in parts :.BufferedOutputStream
        boolean success=ftpc.retrieveFile(selectedFile,os);
        //storeFile() for upload
        os.close();
        if(success)
            txtAC.append("Successfully downloaded file "+selectedFile+"\n"+dFileName.getAbsolutePath());
        else
            txtAC.append("Could not download file "+selectedFile+"\n");
    }
    else
        txtAC.append("Unable to log into ["+uname+"] at ["+IP+"]\n");
    ftpc.disconnect();
}

```

```

        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}

//listing

private void btnlistfilesActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    try{

        String IP=txtIP.getText();

        FTPClient ftpc=new FTPClient();

        txtAC.append("Establisng connection to the server ["+IP+"]...\n");

        ftpc.connect(IP);

        int reply=ftpc.getReplyCode();//to check the status of the connection

        if(FTPReply.isPositiveCompletion(reply))

            txtAC.append("Connection established with server ["+IP+"]\n"+" "+reply);

        else

            txtAC.append("Connection failed with the server ["+IP+"]\n");

        String uname=txtUN.getText();

        String pwd=txtPW.getText();

        txtAC.append("Logging into the server ["+IP+"]\n");

        if(ftpc.login(uname,pwd))

        {

            txtAC.append("Successfully logged into ["+uname+"] at ["+IP+"]\n");

            String pdir=ftpc.printWorkingDirectory();

            //changeWorkingDirectory() to change current directory

            txtAC.append("Working directory is "+pdir+"\n");
        }
    }
}

```

```

FTPFile ftpf[]=ftpc.listFiles(); //takes names of all files in current directory pdir
//ComboBox.removeAllItems(); //to remove the default 5 items

for(int i=0;i<ftpf.length;i++)
{
    cb1.addItem(ftpf[i].getName());
}
}
else
    txtAC.append("Unable to login to ["+uname+"] at ["+IP+"]\n");
txtAC.append("Disconnecting from the server...\n");
ftpc.disconnect();
}
catch(Exception e)
{
    e.printStackTrace();
}
}

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new ftpc1().setVisible(true);
        }
    });
}
}

```

## Output:

The screenshot shows a window titled "FTP Download" with a standard Windows title bar (minimize, maximize, close buttons). The interface includes input fields for "IP" (10.2.0.3), "Username" (Cse1836), and "Password" (Cse1836), along with a "Connect" button. Below these is a dropdown menu showing "12.docx" and a "List Files" button. A large text area displays the following log messages:

```
Establishing connection to the server [10.2.0.3]...  
Connection established with server [10.2.0.3]  
220Logging into the server [10.2.0.3]  
Successfully logged into [cse1851] at [10.2.0.3]  
Establishing connection to the server [10.2.0.3]...  
Connection established with server [10.2.0.3]  
220Logging into the server [10.2.0.3]  
Successfully logged into [cse1851] at [10.2.0.3]  
Working directory is /home/cse18/cse1851  
Connection established with server [10.2.0.3]  
Successfully logged into [cse1851] at [10.2.0.3]  
Successfully downloaded file 12.docx  
C:\Users\Student\Documents\NetBeansProjects\FTP51\12.docx
```

To the right of the log area is a "Download" button.

6. Write a Program to demonstrate Client Server using RMI.

**Aim:** To Perform Client Server using RMI.

**Program:**

**i. AddServerIntf**

```
import java.rmi.*;
public interface AddServerIntf extends Remote
{
    double add(double d1,double d2) throws RemoteException;
}
```

**ii. AddServerImpl**

```
import java.rmi.*;
import java.rmi.server.*;
public class AddServerImpl extends UnicastRemoteObject implements AddServerIntf
{
    public AddServerImpl() throws RemoteException
    {
    }
    public double add(double d1,double d2) throws RemoteException
    {
        return d1+d2;
    }
}
```

**iii. AddClient**

```
import java.rmi.*;
public class AddClient
{
    public static void main(String args[])
    {
        try
        {
            String addServerURL="rmi://" +args[0]+"/AddServer";
            AddServerIntf addServerIntf=
            (AddServerIntf)Naming.lookup(addServerURL);
            System.out.println("First no. is"+args[1]);
            double d1=Double.valueOf(args[1]).doubleValue();
            System.out.println("Second no. is"+args[2]);
            double d2=Double.valueOf(args[2]).doubleValue();
            System.out.println("the sum is "+addServerIntf.add(d1,d2));
        }
    }
}
```

```

        catch(Exception e)
        {
            System.out.println("Exception:" +e);
        }
    }
}

```

#### iv. AddServer

```

import java.rmi.*;
import java.rmi.server.*;
public class AddServer
{
    public static void main(String args[])
    {
        try
        {
            AddServerImpl addServerImpl=new AddServerImpl();

            Naming.rebind("AddServer",addServerImpl);
        }
        catch(Exception e)
        {
            System.out.println("Exception:" +e);
        }
    }
}

```

#### Exectuion Steps

**C: javac \*.java**

**C: rmic AddServerImpl**

**C: start rmiregistry**

**C: java AddServer**

**C: java AddClient localhost 30 23**

#### Output:

The image shows two side-by-side Windows command prompt windows. The left window shows the execution of the following commands: `javac *.java`, `rmic AddServerImpl` (with a warning about deprecated skeletons), `start rmiregistry`, and `java AddServer`. The right window shows the execution of `java AddClient localhost 30 23`, which produces the output: `First no. is 30`, `Second no. is 23`, and `the sum is 53.0`.



7. Write a Program to perform Simple Calculator using RMI  
(Addition,Subtraction.Multiplication,Division)

**Aim:** To perform simple calculator using RMI.

**Program:**

**i.One**

```
import java.rmi.*;

interface one extends Remote
{
    public int add(int a, int b) throws RemoteException;
    public int sub(int a, int b) throws RemoteException;
    public int mul(int a, int b) throws RemoteException;
    public int div(int a, int b) throws RemoteException;
}
```

**ii.Two**

```
import java.rmi.*;
import java.rmi.server.*;

public class two extends UnicastRemoteObject implements one
{
    public two() throws RemoteException { }

    public int add(int a, int b) throws RemoteException
    {
        System.out.println("Hello");
        return (a + b);
    }

    public int sub(int a, int b) throws RemoteException
    {
        System.out.println("Hello");
        return (a - b);
    }
}
```

```

}

public int mul(int a, int b) throws RemoteException
{
    System.out.println("Hello");

    return (a * b);
}

public int div(int a, int b) throws RemoteException
{
    System.out.println("Hello");

    return (a / b);
}
}

```

### **iii.RmiClient**

```

import java.io.*;

import java.rmi.*;

import java.net.*;

public class rmiclient
{
    public static void main(String args[]) throws Exception
    {
        try
        {
            System.out.println("The Two numbers are 40 and 5");

            String s1 = "rmi://localhost/add";

            one onex = (one)Naming.lookup(s1);

            int m = onex.add(40,5);

            System.out.println("Addition : " + m);

```

```

String s2 = "rmi://localhost/sub";

one oney = (one)Naming.lookup(s2);

int n = oney.sub(40, 5);

System.out.println("Subtraction : " + n);

String s3 = "rmi://localhost/mul";

one onez = (one)Naming.lookup(s3);

int o = onez.mul(40, 5);

System.out.println("Multiplication : " + o);

String s4 = "rmi://localhost/div";

one onew = (one)Naming.lookup(s4);

int p = onew.div(40, 5);

System.out.println("Division : " + p);

}

catch (Exception e)

{

System.out.println("Exception" + e);

}

}

}

```

#### **iv.RmiServer**

```

import java.io.*;

import java.rmi.*;

import java.net.*;

public class rmiserver

{

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

{

```

```

try
{
two twox = new two();

Naming.bind("add", twox);

System.out.println("Object registered");

two twoy = new two();

Naming.bind("sub", twoy);

System.out.println("Object registered");

two twoz = new two();

Naming.bind("mul", twoz);

System.out.println("Object registered");

two twow = new two();

Naming.bind("div", twow);

System.out.println("Object registered");

}

catch(Exception e)

{

System.out.println("Exception" + e);

}

}

}

```

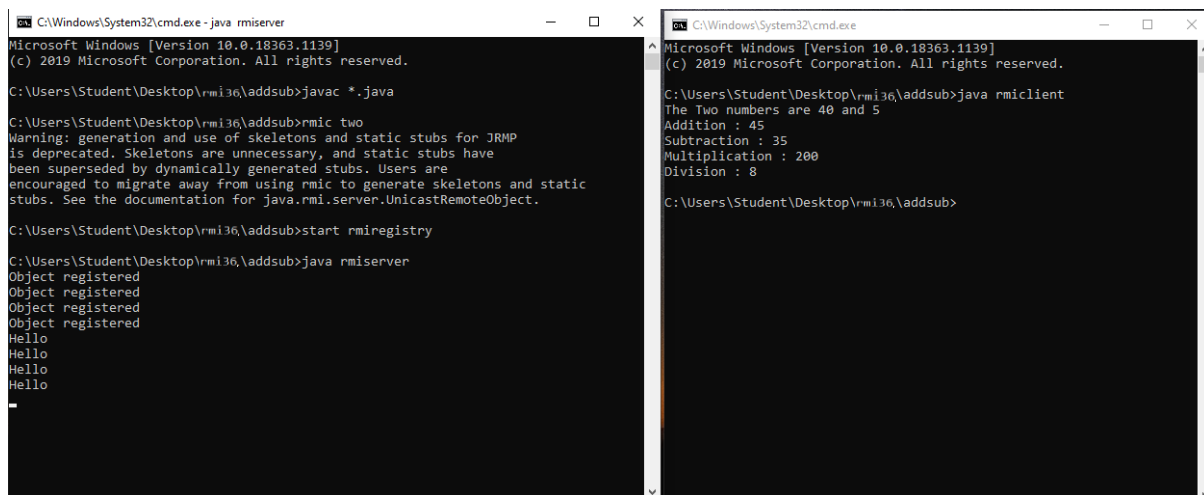
### **Exectuion Steps**

```

C: javac *.java
C: rmic two
C: start rmiregistry
C: java rmiserver
C: java rmiclient

```

## Output:



The image shows two side-by-side Windows command prompt windows. The left window is titled 'C:\Windows\System32\cmd.exe - java rmiserver' and the right window is titled 'C:\Windows\System32\cmd.exe'. Both windows show the output of Java RMI commands.

```
C:\Windows\System32\cmd.exe - java rmiserver
Microsoft Windows [Version 10.0.18363.1139]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Student\Desktop\rmi36\addsub>javac *.java

C:\Users\Student\Desktop\rmi36\addsub>rmic two
Warning: generation and use of skeletons and static stubs for JRMPI
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

C:\Users\Student\Desktop\rmi36\addsub>start rmiregistry

C:\Users\Student\Desktop\rmi36\addsub>java rmiserver
Object registered
Object registered
Object registered
Object registered
Hello
Hello
Hello
Hello
-
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.1139]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Student\Desktop\rmi36\addsub>java rmiclient
The Two numbers are 40 and 5
Addition : 45
Subtraction : 35
Multiplication : 200
Division : 8

C:\Users\Student\Desktop\rmi36\addsub>
```

8. Write a Program to demonstrate Domain Name Server using RMI.

**Aim:** To perform DNS using RMI.

**Program:**

**i.AddDNSServerIntf**

```
import java.rmi.*;

public interface DNSServerIntf extends Remote
{
    String DNS(String s1) throws RemoteException;
}
```

**ii.DNSServerImpl**

```
import java.rmi.*;
import java.rmi.server.*;

public class DNSServerImpl extends UnicastRemoteObject implements DNSServerIntf
{
    public DNSServerImpl() throws RemoteException
    {
    }

    public String DNS(String s1) throws RemoteException
    {
        if(s1.equals("www.osmania.ac.in"))
            return "50.32.24.29";

        if(s1.equals("www.mjcollege.ac.in"))
            return "90.82.44.89";

        if(s1.equals("www.jntu.ac.in"))
            return "150.32.64.20";

        if(s1.equals("www.yahoo.com"))
            return "88.39.124.129";

        else return "No Info about this Address";
    }
}
```

```
}  
}
```

### **iii.DNSClient**

```
import java.rmi.*;  
  
public class DNSClient {  
  
    public static void main(String args[])throws Exception  
  
    { try {  
  
        String dnsServerURL="rmi://" +args[0]+"/DNSServer";  
  
        DNSServerIntf dnsServerIntf= (DNSServerIntf)Naming.lookup(dnsServerURL);  
  
        System.out.println("The website name is "+args[1]);  
  
        String s1=args[1];  
  
        System.out.println("the site is at "+dnsServerIntf.DNS(s1));  
  
    }  
  
    catch(Exception e) {  
  
        System.out.println("Exception:" +e); } } }
```

### **iv.DNSServer**

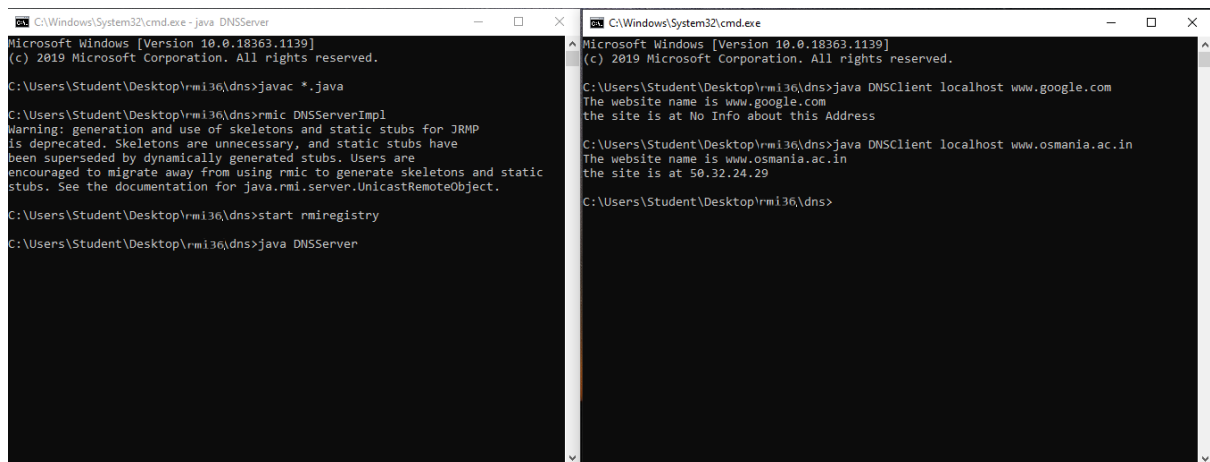
```
import java.rmi.*;  
  
import java.rmi.server.*;  
  
public class DNSServer{  
  
    public static void main(String args[])throws Exception  
  
    { try {  
  
        DNSServerImpl dnsServerImpl=new DNSServerImpl();  
  
        Naming.rebind("DNSServer",dnsServerImpl); }  
  
    catch(Exception e)  
  
    {  
  
        System.out.println("Exception:" +e);  
  
    }  
  
}
```

```
}  
  
}
```

### Execution Steps

```
C: javac *.java  
C: rmic DNSServerImpl  
C: start rmiregistry  
C: java DNSServer  
C: java DNSClient localhost www.google.com
```

### Output:



```
C:\Windows\System32\cmd.exe - java DNSServer  
Microsoft Windows [Version 10.0.18363.1139]  
(c) 2019 Microsoft Corporation. All rights reserved.  
  
C:\Users\Student\Desktop\rmi36\dns>javac *.java  
  
C:\Users\Student\Desktop\rmi36\dns>rmic DNSServerImpl  
Warning: generation and use of skeletons and static stubs for JRMP  
is deprecated. Skeletons are unnecessary, and static stubs have  
been superseded by dynamically generated stubs. Users are  
encouraged to migrate away from using rmic to generate skeletons and static  
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.  
  
C:\Users\Student\Desktop\rmi36\dns>start rmiregistry  
  
C:\Users\Student\Desktop\rmi36\dns>java DNSServer  
  
C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.18363.1139]  
(c) 2019 Microsoft Corporation. All rights reserved.  
  
C:\Users\Student\Desktop\rmi36\dns>java DNSClient localhost www.google.com  
The website name is www.google.com  
the site is at No Info about this Address  
  
C:\Users\Student\Desktop\rmi36\dns>java DNSClient localhost www.osmania.ac.in  
The website name is www.osmania.ac.in  
the site is at 50.32.24.29  
  
C:\Users\Student\Desktop\rmi36\dns>
```



9. Write a Program to implement ECHO SERVER using RPC.

**Aim: To Implement Echo Server using RPC.**

**Program:**

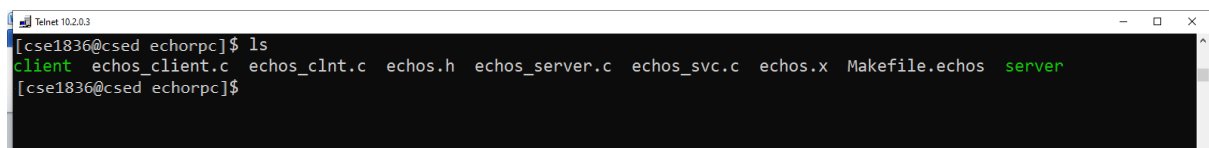
**i.Echos.x**

```
program ECHOSERVER_PROGRAM
{
    version ECHOSERVER_VERSION
    {
        string ECHO(string)=1;
    }=1;
}=0x21234589;
```

**Execution command**

**Command : \$ rpcgen -a echos.x**

**\$ ls**

A terminal window titled 'Telnet 10.2.0.3' showing a directory listing. The prompt is '[cse1836@cshed echorpc]\$'. The command 'ls' has been executed, and the output is: 'client echos\_client.c echos\_clnt.c echos.h echos\_server.c echos\_svc.c echos.x Makefile.echos server'. The word 'client' is highlighted in green, and 'server' is also highlighted in green.

**ii.Echos client.c**

```
#include "echos.h"

Void echoserver_program_1(char *host)
{
    CLIENT *clnt;

    char **result_1;
```

```

    char * echo_1_arg;

#ifdef DEBUG

    clnt = clnt_create (host, ECHOSERVER_PROGRAM, ECHOSERVER_VERSION, "udp");

    if (clnt == NULL) {

        clnt_pcreateerror (host);

        exit (1);

    }

#endif /* DEBUG */

    echo_1_arg=(char *)malloc(20);

    printf("\n Enter a message:");

    scanf("%s",echo_1_arg);

    result_1 = echo_1(&echo_1_arg, clnt);

    if (result_1 == (char **) NULL) {

        clnt_perror (clnt, "call failed");

    }

    else

        printf("\n The message returned is %s",*result_1);

#ifdef DEBUG

    clnt_destroy (clnt);

#endif /* DEBUG */

}

int

main (int argc, char *argv[])

```

```

{

    char *host;

    if (argc < 2) {

        printf ("usage: %s server_host\n", argv[0]);

        exit (1);

    }

    host = argv[1];

    echoserver_program_1 (host);

exit (0);}

```

### **iii.Echos\_server.c**

```

#include "echos.h"

char **

echo_1_svc(char **argp, struct svc_req *rqstp)

{

    static char * result;

    /* insert server code here

    */

    result=*argp;

    return &result;

}

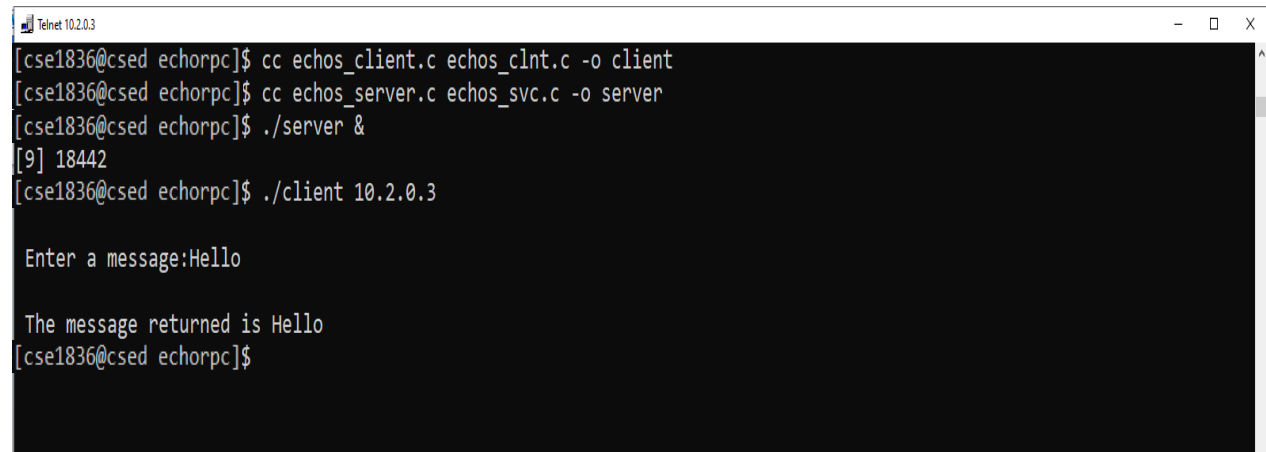
```

### Execution Steps:

```
$ cc echos_client.c echos_clnt.c -o client
```

```
$ cc echos_server.c echos_svc.c -o server
```

### Output:



```
Telnet 10.2.0.3
[cse1836@cse1836-echo]$ cc echos_client.c echos_clnt.c -o client
[cse1836@cse1836-echo]$ cc echos_server.c echos_svc.c -o server
[cse1836@cse1836-echo]$ ./server &
[9] 18442
[cse1836@cse1836-echo]$ ./client 10.2.0.3

Enter a message:Hello

The message returned is Hello
[cse1836@cse1836-echo]$
```

10. Write a Program to find GCD using RPC.

**Aim:** To Find GCD using RPC.

**Program:**

**i.gcd.x**

```
struct num  
  
{  
  
long a;  
  
long b;  
  
};  
  
program gcd_prog{  
  
version gcd_vers{  
  
long gcd_fn(num)=1;  
  
}=1;  
  
}=0x30000001;
```

**Execution Steps:**

```
$rpcgen gcd.x
```

Generate server stub program

```
$rpcgen -Ss gcd.x>gcd_server.c
```

Client stub

```
$rpcgen -Sc gcd.x>gcd_client.c
```

```
$ ls
```



```
Telnet 10.2.0.3  
[cse1836@cse1836 gcdrpc]$ ls  
gcd_client  gcd_client.c  gcd_clnt.c  gcd.h  gcd_server  gcd_server.c  gcd_svc.c  gcd.x  gcd_xdr.c  
[cse1836@cse1836 gcdrpc]$
```

## ii.gcd\_client code:

```
#include "gcd.h"

void gcd_prog_1(char *host,num number)
{
    CLIENT *clnt;
    long *result_1;
    num gcd_fn_1_arg;
    gcd_fn_1_arg.a=number.a;
    gcd_fn_1_arg.b=number.b;
#ifdef DEBUG
    clnt = clnt_create (host, gcd_prog, gcd_vers, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }

#endif /* DEBUG */

    result_1 = gcd_fn_1(&gcd_fn_1_arg, clnt);
    if (result_1 == (long *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    printf("gcd is %d",*result_1);
#ifdef DEBUG
    clnt_destroy (clnt);
#endif /* DEBUG */
}

Int main (int argc, char *argv[])
{
    char *host;
    num n;
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    n.a=atol(argv[2]);
    n.b=atol(argv[3]);

    gcd_prog_1 (host,n);
    exit (0);
}
```

## iii.gcd\_server code:

```
int gcd(int a ,int b){
    if (b==0)
```

```
return a;  
return gcd(b,a%b);}
```

```
#include "gcd.h"
```

```
long * gcd_fn_1_svc(num *argp, struct svc_req *rqstp)  
{
```

```
    static long result;
```

```
    /*  
    * insert server code here  
    */
```

```
result=gcd((*argp).a,(*argp).b);
```

```
    return &result;
```


```
}
```

### **Execution Steps:**

```
$ cc -o gcd_server gcd_server.c gcd_svc.c gcd_xdr.c -lnsl
```

```
$ cc -o gcd_client gcd_client.c gcd_clnt.c gcd_xdr.c -lnsl
```

### **Output:**



```
Telnet 10.2.0.3  
[cse1836@cse1836]$ cc -o gcd_server gcd_server.c gcd_svc.c gcd_xdr.c -lnsl  
[cse1836@cse1836]$ cc -o gcd_client gcd_client.c gcd_clnt.c gcd_xdr.c -lnsl  
[cse1836@cse1836]$ ./gcd_server &  
[10] 18643  
[cse1836@cse1836]$ ./gcd_client 10.2.0.3 125 50  
gcd is 25  
[cse1836@cse1836]$
```

11. Write a Program to perform NFS.

**Aim:** To perform Client Server File sharing using NFS.

**Program:**

### Setting Up NFS Server And Client On Linux

**NFS**, stands for **N**etwork **F**ile **S**ystem, is a server-client protocol used for sharing files between linux/unix to unix/linux systems. NFS enables you to mount a remote share locally. You can then directly access any of the files on that remote share.

### Scenario

In this how-to, I will be using two systems which are running with RHEL. The same steps are applicable for Scientific Linux 7 distributions.

Here are my testing nodes details.

```
NFS Server IP Address: 10.2.0.5
```

```
NFS Client IP Address: 10.2.4.30
```

### Server Side Configuration

Install NFS packages in your Server system by using the following command:

```
yum install nfs-utils nfs-utils-lib
```

Enable and start NFS services:

```
systemctl enable rpcbind
```

```
systemctl enable nfs-server
```

```
systemctl enable nfs-lock
```

```
systemctl enable nfs-idmap
```

```
systemctl start rpcbind
```

```
systemctl start nfs-server
```

```
systemctl start nfs-lock
```

```
systemctl start nfs-idmap
```

Now, let us create some shared directories in server.



Create a shared directory named **'/var/newshare'** in server and let the client users to read and write files in that directory.

```
mkdir /var/newshare
```

```
chmod 777 /var/newshare/
```

Export shared directory on NFS Server:

Edit file **/etc/exports**,

```
vi /etc/exports
```

Add the following line:

```
/var/newshare/ 10.2.4.1/(rw,sync,no_root_squash,no_all_squash)
where,
```

**/var/newshare** - shared directory

**10.2.4.1/24** - IP address range of clients

**rw** - Writable permission to shared folder

**sync** - Synchronize shared directory

**no\_root\_squash** - Enable root privilege

**no\_all\_squash** - Enable user's authority

Restart the NFS service:

```
Service nfs restart
```

### Client Side Configuration

Install NFS packages in your client system by using the following command:

```
yum install nfs-utils nfs-utils-lib
```

Enable and start NFS services:

```
systemctl enable rpcbind
```

```
systemctl enable nfs-server
```

```
systemctl enable nfs-lock
```

```
systemctl enable nfs-idmap
```

```
systemctl start rpcbind  
  
systemctl start nfs-server  
  
systemctl start nfs-lock  
  
systemctl start nfs-idmap
```

## Mount NFS shares On clients

Create a mount point to mount the shared folder '**var/newshare**' which we've created before in the server.

```
mkdir /var/new1
```

Mount the share from server to client as shown below

```
mount -t nfs 10.2.0.5:/var/newshare/ /var/new/
```

Sample Output:

```
mount.nfs: Connection timed out
```

Probably, it will show a **connection timed out error** which means that the firewall is blocking our NFS server. To access NFS shares from remote clients, we must allow the following nfs ports in the NFS server iptables/firewall.

If you don't know which ports to allow through firewall, run the following command:

You should allow the above ports.

To do that, go to the NFS server, and run the following commands:

```
firewall-cmd --permanent --add-port=111/tcp  
  
firewall-cmd --permanent --add-port=54302/tcp  
  
firewall-cmd --permanent --add-port=20048/tcp  
  
firewall-cmd --permanent --add-port=2049/tcp  
  
firewall-cmd --permanent --add-port=46666/tcp  
  
firewall-cmd --permanent --add-port=42955/tcp  
  
firewall-cmd --permanent --add-port=875/tcp
```

Restart firewalld service to take effect the changes:

```
firewall-cmd --reload
```

Again mount the share in client system with command:

```
mount -t nfs 10.2.0.5:/var/newshare/ /var/new/
```

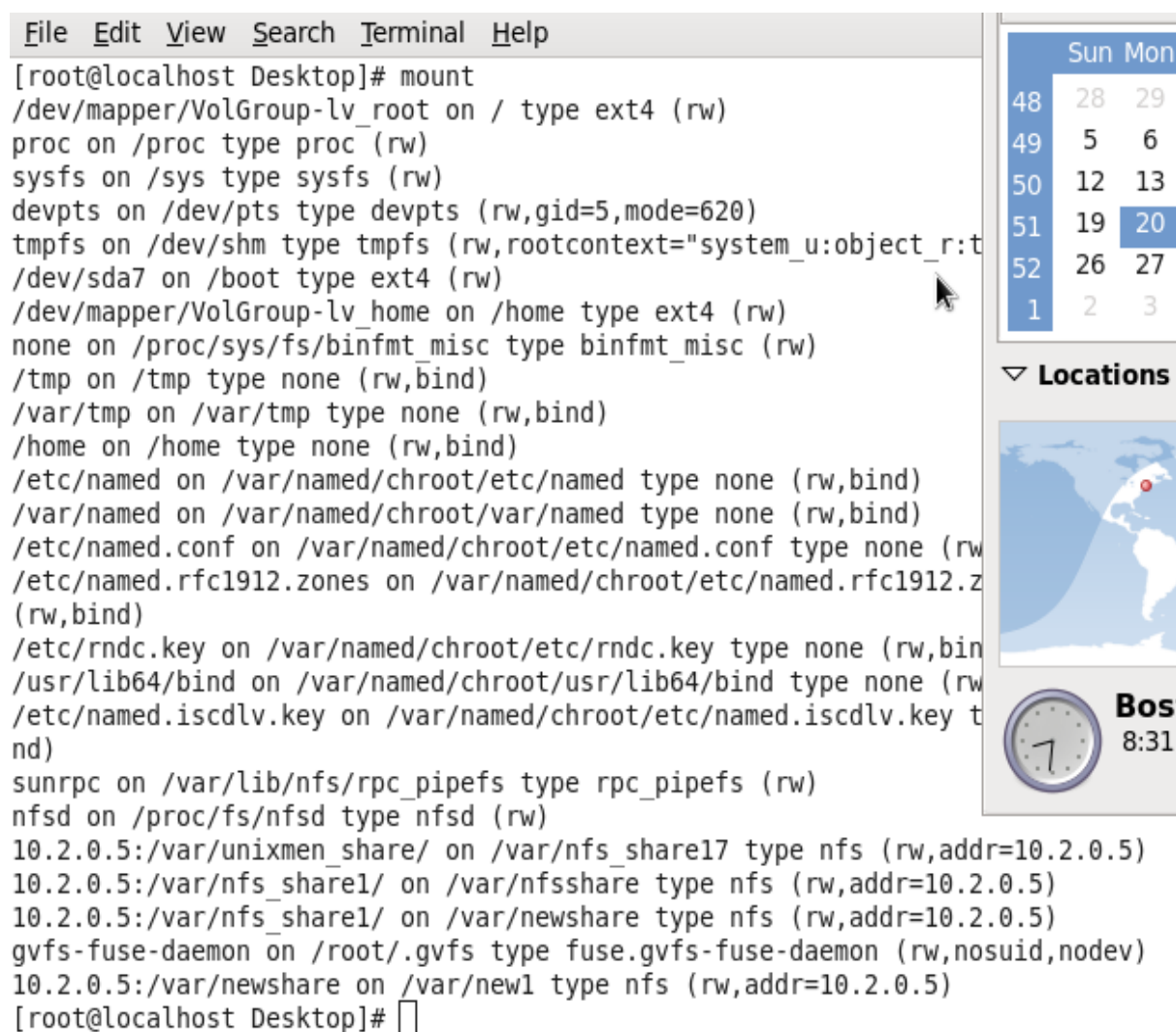
Now the NFS share will mount without any connection timed out error.

### Verifying NFS Shares On Clients

Verify the share from the server is mounted or not using 'mount' command.

```
mount
```

Sample output:



The screenshot shows a terminal window with the following output for the 'mount' command:

```
[root@localhost Desktop]# mount
/dev/mapper/VolGroup-lv_root on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw,rootcontext="system_u:object_r:tmpfs")
/dev/sda7 on /boot type ext4 (rw)
/dev/mapper/VolGroup-lv_home on /home type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
/tmp on /tmp type none (rw,bind)
/var/tmp on /var/tmp type none (rw,bind)
/home on /home type none (rw,bind)
/etc/named on /var/named/chroot/etc/named type none (rw,bind)
/var/named on /var/named/chroot/var/named type none (rw,bind)
/etc/named.conf on /var/named/chroot/etc/named.conf type none (rw,bind)
/etc/named.rfc1912.zones on /var/named/chroot/etc/named.rfc1912.zones type none (rw,bind)
/etc/rndc.key on /var/named/chroot/etc/rndc.key type none (rw,bind)
/usr/lib64/bind on /var/named/chroot/usr/lib64/bind type none (rw,bind)
/etc/named.iscdlv.key on /var/named/chroot/etc/named.iscdlv.key type none (rw,bind)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
nfsd on /proc/fs/nfsd type nfsd (rw)
10.2.0.5:/var/unixmen_share/ on /var/nfs_share17 type nfs (rw,addr=10.2.0.5)
10.2.0.5:/var/nfs_share1/ on /var/nfsshare type nfs (rw,addr=10.2.0.5)
10.2.0.5:/var/nfs_share1/ on /var/newshare type nfs (rw,addr=10.2.0.5)
gvfs-fuse-daemon on /root/.gvfs type fuse.gvfs-fuse-daemon (rw,nosuid,nodev)
10.2.0.5:/var/newshare on /var/new1 type nfs (rw,addr=10.2.0.5)
[root@localhost Desktop]#
```

On the right side of the terminal window, there is a desktop widget. It includes a calendar for the month of November, showing the 20th as the current date. Below the calendar is a 'Locations' section with a map of the world and a red dot indicating a location. At the bottom of the widget is a clock showing the time as 8:31 and the location as 'Boston'.