DC Practicals

Name : Aditya Shivram Mahajan
Branch : Computer Science
Reg No. : 2017BCS001
Roll No. : A10
Year : Final Year

Index

Practical 1 : Design and development of a Distributed application using socket.

Title :- Design & development of distributed application using socket

Socket :

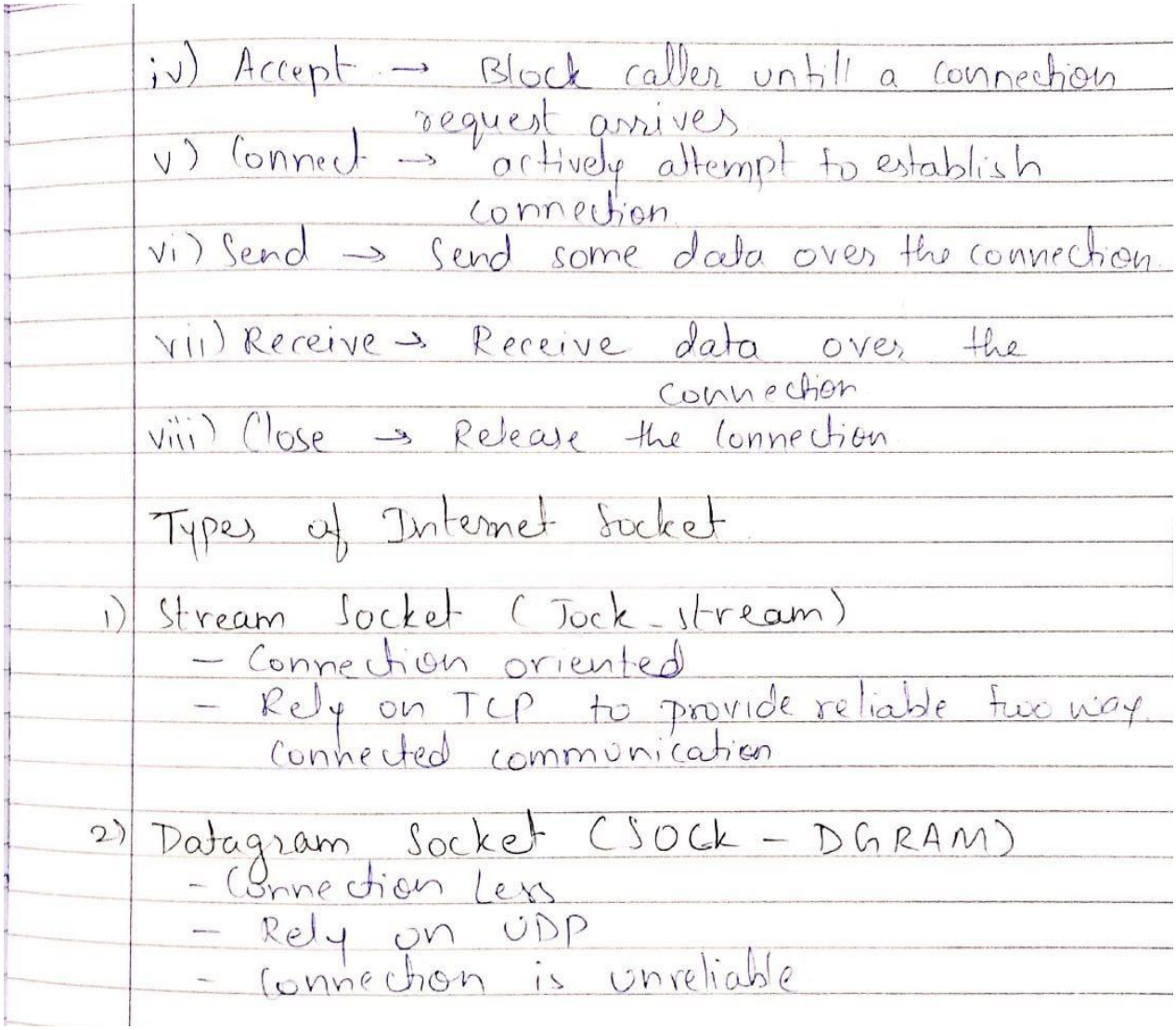
Socket allow communication between two different process on the same or different machines. To be more precise, it's a way to talk to other computers using standard unix file descriptors. In Unix, every I/O action is done by writing or reading file descriptor. A file descriptor is just a integer associated with an open file it can be network connection, a text file, a terminal or something.

Where is Socket used ?

A unix socket used in a client-server application framework. A server is a process that performs some functions on request from a client most of the application-level protocol like FTP, SMTP, and POP3 make use of of socket to establish connection between client and server and then for exchanging data

The Socket operations for TCP/IP

i) Socket → create a new connection end point
ii) Bind → Attach a local address to source
iii) Listen → Tell operating System what the max no of pending connection request should be



Program : Server.py

```
import socket
import threading

HEADER = 64
PORT = 5050
SERVER = "192.168.1.105"
#SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"
```

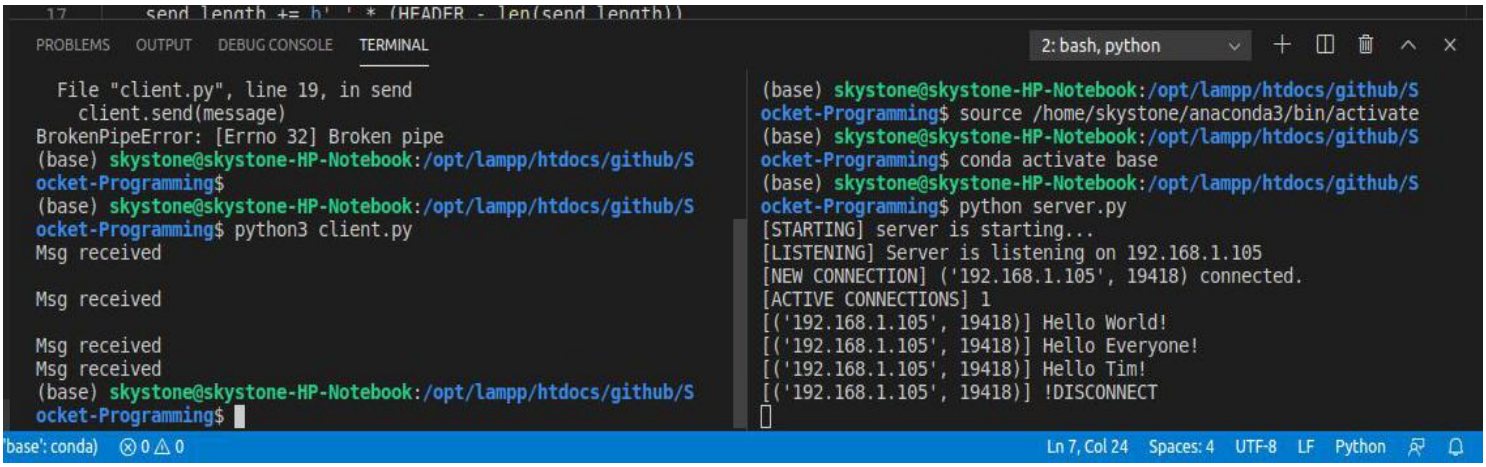
```
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDR)

def handle_client(conn, addr):
      print(f"[NEW CONNECTION] {addr} connected.")
      connected = True
      while connected:
      msg_length = conn.recv(HEADER).decode(FORMAT)
      if msg_length:
            msg_length = int(msg_length)
            msg = conn.recv(msg_length).decode(FORMAT)
            if msg == DISCONNECT_MESSAGE:
                  connected = False
            print(f"[{addr}] {msg}")
            conn.send("Msg received".encode(FORMAT))
      conn.close()


def start():
      server.listen()
      print(f"[LISTENING] Server is listening on {SERVER}")
      while True:
      conn, addr = server.accept()
      thread = threading.Thread(target=handle_client, args=(conn,
addr))
      thread.start()
      print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 1}")


print("[STARTING] server is starting...")
start()
```

Program : Client.py

```
import socket

HEADER = 64
PORT = 5050
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"
SERVER = "192.168.1.105"
```

```
ADDR = (SERVER, PORT)

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDR)

def send(msg):
      message = msg.encode(FORMAT)
      msg_length = len(message)
      send_length = str(msg_length).encode(FORMAT)
      send_length += b' ' * (HEADER - len(send_length))
      client.send(send_length)
      client.send(message)
      print(client.recv(2048).decode(FORMAT))

send("Hello World!")
input()
send("Hello Everyone!")
input()
send("Hello Tim!")

send(DISCONNECT_MESSAGE)
```

Output :



Practical 2 : Design and development of a threaded Distributed application using socket.

Practical 2

Title :- Design & Development of a multithreaded distributed application using socket

Theory -

An implementation of multithreaded client - server with TCP & IPV4 in pure C. It allows multiple clients to connect to and interact with the server simultaneously

Server code basically starts a server & accepts connections from clients Upon accepting a client connection, it dispatches a thread to interact with the client

Normally you would use write() to send a message to the client, and use read() to receive a message from the client

Client side we use write() to send message to the server and use read() to receive a message from the server

Program : Client.py

```
import socket
def Main():
    s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((socket.gethostname() ,1024))
    while True:
        num1 =input("Enter first numbers:")
        s.send(num1.encode('ascii'))
        num2 =input("Enter second numbers:")
        s.send(num2.encode('ascii'))
        data = s.recv(1024)
        print('\n',str(data.decode('ascii')))
        ans = input('\nDo you want to continue operation with other inputs(y/n) :')
        if ans == 'y':
              continue
        else:
              break
    s.close()
if __name__ == '__main__':
    Main()
```

Program : Server.py

```
import socket
from _thread import *
import threading
print_lock = threading.Lock()

def threaded(c):
    while True:
        num1 = c.recv(1024)
        num2 = c.recv(1024)
        if not (num1 or num2):
        print_lock.release()
        break
        add=int(num1)+int(num2)
        sub=int(num1)-int(num2)
        mul=int(num1)*int(num2)
        div=int(num1)/int(num2)
        mod=int(num1)%int(num2)
        exp=int(num1)**int(num2)
        c.send(("addition: "+str(add)+"\nSubstraction:
        "+str(sub)+"\nMultiplication: "+str(mul)+"\nDivision: "+str(div)+"\nModulus:
"+str(mod)+"\nExponent: "+str(exp)).encode())
    c.close()

def Main():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
s.bind((socket.gethostname(),1024))
    s.listen(5)
    while True:
        c, addr = s.accept()
        print_lock.acquire()
        print('Connected to :', addr[0], ':', addr[1])
        start_new_thread(threaded, (c,))
    s.close()

if __name__ == '__main__':
    Main()
```

Output :

skystone@skystone-HP-Notebook:**~/Desktop/DC Prac/Practical 2**$ python3 server.py
Connected to : 127_0_0_1 : 60012

skystone@skystone-HP-Notebook:**~/Desktop/DC Prac/Practical 2**$ python3 client.py
Enter first numbers:5
Enter second numbers:9

Addition: 14
Substraction: -4
Multiplication: 45
Division: 0.5555555555555556
Modulus: 5
Exponent: 1953125

Do you want to continue operation with other inputs(y/n) : y

Enter first numbers:3
Enter second numbers:4

Addition: 7
Substraction: -1
Multiplication: 12
Division: 0.75
Modulus: 3
Exponent: 81

Do you want to continue operation with other inputs(y/n) :n
skystone@skystone-HP-Notebook:**~/Desktop/DC Prac/Practical 2**$

Practical 3 : Installation and study of Mobile agent using Aglet platform.

Title :- Installation & study of mobile agent using aglet platform

Mobile Agent :-

Mobile Agents are the pieces of codes that are used to store data and are independent in nature. They are self driven and does not require corresponding node for communicaton as they are capable of functioning even if user get disconnected from the network.

They are also called as transportable agents. They can be broadly classified into two types.

1) Agents with predefined path
2) Agents with undefined path

Installation of aglet:

1) Decompress the archive →

jar xvf aglets-2.0.2.jar

Once extraction is completed the folder will be visible.

bin :- It will contain executables
cnf :- Contains configuration files.
public :- Contains a few examples of aglet
lib :- contains aglet library

2) Install the platform

```
$ cd bin
$ chmod 755 ant
$ ./ant
```

3) Set up policy & environment variable.

```
$ export AGLETS_HOME = /java/aglet
$ export AGLETS_PATH = $AGLETS_HOME
$ export PATH = $PATH : $AGLETS_HOME/bi
$ ant install_home
```

4) Run aglets server

```
$ aglet sd
```

Conclusion: Thus we studied about aglet and installed. The aglet on the system using agents we studied using mobile agents.

Practical 4 : Design and development of a Distributed application using Agelt platform.

Practical 4

Title :- Design and development of distributed application using aglet platform.

Theory :-

The following class shows a simple aglet that, once loaded, prints a few messages on the standard output

```
import com.ibm.aglet.*;

public class FirstAglet extends Aglet
{
    public void run()
    {
        System.out.println("\n It Hello\n");
        for(int i=0; i<10; i++){
            System.Out.println("\n i is"+i);
        }
    }
}
```

you can compile it from commandline, after ensuring you have the Aglets library in your classpath.

export classpath = $classpath: /java/aglets/lib/aglets-2.0.2.jar

After that, just compile your agent from the command line,

$ javac FirstAglet.java

Even if aglets are like other java classes they can not be run as standalone programs, thus you have to run aglets into the platform. Before that, you must make agents reachable by the platform itself, that means you must have the agent in the server public root, that is by default the folder public of the aglet plaform installation

Practical 5 : Design and development of a Distributed application using RPC.

Practical 5

Title :- Design and development of distributed application using RPC

Remot procedure Call :-

A remote procedure call is an interprocess communication technique that is used for client server-based application. It is also known as subroutine call or a function call A client has a request message that the RPC transate and send to the server Their request may be a procedure or a function call to a remote server. When server receives the request, it sends the required response back to client. The client is blaved while the server is procesing the call & only resumed execution after the server is finished.

The sequence of event in a remote procedure call are given as follows.

- The client stub is called by the Client.
- The client stub maker a system call to sent the message to the server and puts the parameter in the message.
- The message is sent from the client to the server by the client operating system.

- The message is passed to the server stub by the server operating sytem.

- The parameter are removed from the message by server stub.

- Then server procedure is called by the server stub.

Running Commands.

1) rpcgen calculate.x
2) cc -g -c -o calculate_cnt -o calculate_ant.c
3) cc -g -c -o calculate_client.o calculate_client.c
4) cc -g -c -o calculate_xdr.o calculate_xdr.c
5) cc -g -o calculate_client calculate_ant.o calculate_client -o calculate_xdr.
6) cc -g -c -o calculate_svc.o calculate_svc.
7) cc -g -o calculate_server calculate_svc.o calculate_server.o calculate_xdr -lns

8) ./calculate_server.
9) ./calculate_client localhost

Enter operation +
Enter 1st number 20
Enter 2nd number 30
result 50.

Program : ArCompt.java

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

```
import java.util.*;
import java.lang.*;
// Creating an Interface
public interface ArCompt
extends java.rmi.Remote {
	// Declaring the method
	public int factorial(int x) throws java.rmi.RemoteException;
	public double calfun(double x, int n) throws java.rmi.RemoteException;
}
```

Program : ArComptImpl.java

```
import java.util.Scanner;
import java.rmi.*;
import java.rmi.server.*;
import java.util.*;
import java.lang.*;
public class ArComptImpl
extends java.rmi.server.UnicastRemoteObject
implements ArCompt {
	public ArComptImpl()
	throws java.rmi.RemoteException {
		super();
	}
	public int factorial(int x)
	throws java.rmi.RemoteException {
	if (x == 0) {
			return 1;
	} else {
			return x * factorial(x - 1);
	}
	}
	public double calfun(double x1, int y) throws java.rmi.RemoteException {
	double radians = Math.toRadians(x1);
	if (y == 1) {
			return Math.sin(radians);
	} else if (y == 2) {
			return Math.cos(radians);
	} else if (y == 3) {
			return Math.tan(radians);
	} else {
			return 0;
	}
	}
}
```

Program : ArComptClient.java

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

```
import java.util.*;
import java.io.*;
import java.lang.*;
import java.net.*;
public class ArComptClient {
	public static void main(String[] args) {
	Scanner s = new Scanner(System.in);
	try {
		ArCompt c = (ArCompt) Naming.lookup("rmi://localhost/ArCompt");
		System.out.print("\nEnter the Number to find factorial:");
		int num = s.nextInt();
		System.out.print("\nFactorial: " + c.factorial(num));
		System.out.print("\nEnter the value (in degree) to find sin():");
		double a = s.nextDouble();
		System.out.print("\nSin(): " + c.calfun(a, 1));
		System.out.print("\nEnter the value (in degree) to find cos():");
		double b = s.nextDouble();
		System.out.print("\ncos(): " + c.calfun(b, 2));
		System.out.print("\nEnter the value (in degree) to find tan():");
		double t = s.nextDouble();
		System.out.print("\ntan(): " + c.calfun(t, 3));
	} catch (Exception e) {
			System.out.print(e);
	}
	}
}
```

Program : ArComptServer.java

```
import java.rmi.*;
import java.rmi.server.*;
public class ArComptServer {
	public static void main(String[] args) {
	try {
		ArComptImpl stub = new ArComptImpl();
		Naming.rebind("rmi:///ArCompt", stub);
	} catch (Exception e) {
			System.out.print(e);
	}
	}
}
```

Output :

```
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 5$ javac ArCompt.java
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 5$ javac ArComptImpl.java
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 5$ javac ArComptServer.java
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 5$ javac Arfomptflient.java
```

```
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 5$ javac ArComptClient.java
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 5$ rmic ArComptImpl
Warning: generation and use of skeletons and static stubs for JR. is deprecated. Skeletons are
unnecessary. and static stubs have been superseded by dynamically generated stubs. Users are
encouraged to migrate amay from using rmic to generate skeletons and static stubs. See the
documentation for java.rmi.server.UnicastRemoteObject.
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 5$ rmiregistery
```

```
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 5$ java ArComptServer
```

```
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 5$ java ArComptClient

Enter the Number to find factorial:3
Factorial: 6

Enter the value (in degree) to find sin():30
Sin(): 0.49999099999999994

Enter the value (in degree) to find cos():60
cos(): 0.5000000000000001

Enter the value (in degree) to find tan():45
tan(): 0.9999999999999999dell

skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 5$
```

Practical 6 : Design and development of a Distributed application using RMI.

Practical 6

Title : Design & Development of a distributed application using RMI

Theory :

The RMI (Remote Method Invocation) an API that provides a mechanism to create distributed application in java The RMI allowed an object to invoke methods on an object running in another JVM

The RMI provides remote communication between the applications using 2 objects stub and skeleton

A remote object is an object whose method can be invoked from another JVM.

1. Stub :-

The stub is an object, acts as a gateway for the client side. All the outgoing request are returned through it. It resides at at the client side and represents the remote object when the caller invokes method on the stub object, it does the following tasks

- It initiates a connection with remote VM (JVM)
- It waits for the result.

- It reads the return value or exception
- It finally, returns the value to the caller

2. Skeleton:

It is an object, acts as a gateway for the server side object. All the increasing requests are routed through it. When the skeleton receives the incoming request, it does the following tasks.

- It reads the parameters for the remote method
- It invokes the method on the actual remote object
- It writes & transmits the result to the caller.

* Steps to write the RMI program:

1. Create the remote interface.
2. Provide the implementation of the remote interface.
3. Compile the implementation class and create the stub an skeleton objects using rmic tool
4. Start the registry service by rmiregistry tool
5. Create & Start the remote application.
6. Create & start the client application

Program:

Calculator Interface :

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface CalcInterface extends Remote {
      public int add(int x, int y) throws RemoteException;
      public int sub(int x, int y) throws RemoteException;
      public int mul(int x, int y) throws RemoteException;
      public int div(int x, int y) throws RemoteException;
}
```

Calculator RMI :

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class CalcRmi extends UnicastRemoteObject implements
CalcInterface {
      public CalcRmi() throws RemoteException {
      int a, b;
      }
      public int add(int a, int b) throws RemoteException {
      return a + b;
      }
      public int sub(int a, int b) throws RemoteException {
      return a - b;
      }
      public int mul(int a, int b) throws RemoteException {
      return a * b;
      }
      public int div(int a, int b) throws RemoteException {
      return a / b;
      }
}
```

Calculator Server :

```
import java.rmi.registry.Registry;
public class CalcServer {
```

```
      public static void main(String args[]) {
      try {
            Registry r =
java.rmi.registry.LocateRegistry.createRegistry(1099);
            r.rebind("Calc", new CalcRmi());
            System.out.println("Server connected");
      } catch (Exception e) {
                  System.out.println("Server not connected " + e);
            }
      }
}
```

Calculator Client :

```
import java.rmi.Naming;
import java.util.Scanner;
public class CalcClient {
      public static void main(String args[]) {
      System.out.println("MENU");
      System.out.println("");
      System.out.println("Enter 1 for addition");
      System.out.println("Enter 2 for substraction");
      System.out.println("Enter 3 for multiplication");
      System.out.println("Enter 4 for divition");
      System.out.println("");
      System.out.println("Enter your choice");
      Scanner sc = new Scanner(System.in);
      try {
            CalcInterface c = (CalcInterface)
Naming.lookup("//localhost/Calc");
            int choice = sc.nextInt();
            int x, y;
            switch (choice) {
                  case 1:
                  {
                        System.out.println("Enter the first value");
                        x = sc.nextInt();
                        System.out.println("Enter the second value");
                        y = sc.nextInt();
                        System.out.println("Answer is : " + c.add(x, y));
                        Break;
```

```
                }
                case 2:
                {
                      System.out.println("Enter the first value");
                      x = sc.nextInt();
                      System.out.println("Enter the second value");
                      y = sc.nextInt();
                      System.out.println("Answer is : " + c.sub(x, y));
                      Break;
                }
                case 3:
                {
                      System.out.println("Enter the first value");
                      x = sc.nextInt();
                      System.out.println("Enter the second value");
                      y = sc.nextInt();
                      System.out.println("Answer is : " + c.mul(x, y));
                      Break;
                }
                case 4:
                {
                      System.out.println("Enter the first value");
                      x = sc.nextInt();
                      System.out.println("Enter the second value");
                      y = sc.nextInt();
                      System.out.println("Answer is : " + c.div(x, y));
                      Break;
                }
          }
     } catch (Exception e) {
                System.out.println(e);
          }
     }
}
```

Output :

skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 6$ javac CalcInterface.java
Picked up _JAVAOPTIONS: -DJava.net.preferIPv6Addresses=true
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 6$ javac CalcRmi.java

```
Picked up _JAVAOPTIONS: -DJava.net.preferIPv6Addresses=true
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 6$ javac CalcServer.java
Picked up _JAVAOPTIONS: -DJava.net.preferIPv6Addresses=true
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 6$ javac CalcClient.java
Picked up _JAVAOPTIONS: -DJava.net.preferIPv6Addresses=true
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 6$ java CalcServer
Picked up _JAVAOPTIONS: -DJava.net.preferIPv6Addresses=true
Server Connected
^Cskystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 6$
```

```
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 6$ Java CalcClient
Picked up JAVA OPTIONS: -Djava.net.preferIPv6Addresses=true

MENU
Enter 1 for addition
Enter 2 for substraction
Enter 3 for multiplication
Enter 4 for division

Enter your choice: 2

Enter the first value: 12
Enter the second value: 21
Answer is : -9
skystone@skystone-HP-Notebook:~/Desktop/DC Prac/Practical 6$
```

Practical 7 : Study and Installation of Apache Web Server.

Practical 7

Title :- Study & Installation of Apache web server.

Apache Web Server :-

The apache web server is the most widely upset web server in the word. It provide many powerful features like dynamically loadable Modules, robust media support, and extensive integsation with other popular software.

Installation of apache Web server

1) Updating the package
   $ sudo apt update

2) Install the apache2 package
   $ sudo apt install apache2.

3) Adjusting the firewall
   - List the ufw application profile by typing.

```
$ sudo ufw applist
$ sudo ufw allow 'Apache'
$ sudo ufw status
```

4) Checking the web server.

$ sudo systemctl status apache2.

5) Then by writing
http://localhost the web server basic page can be loaded.

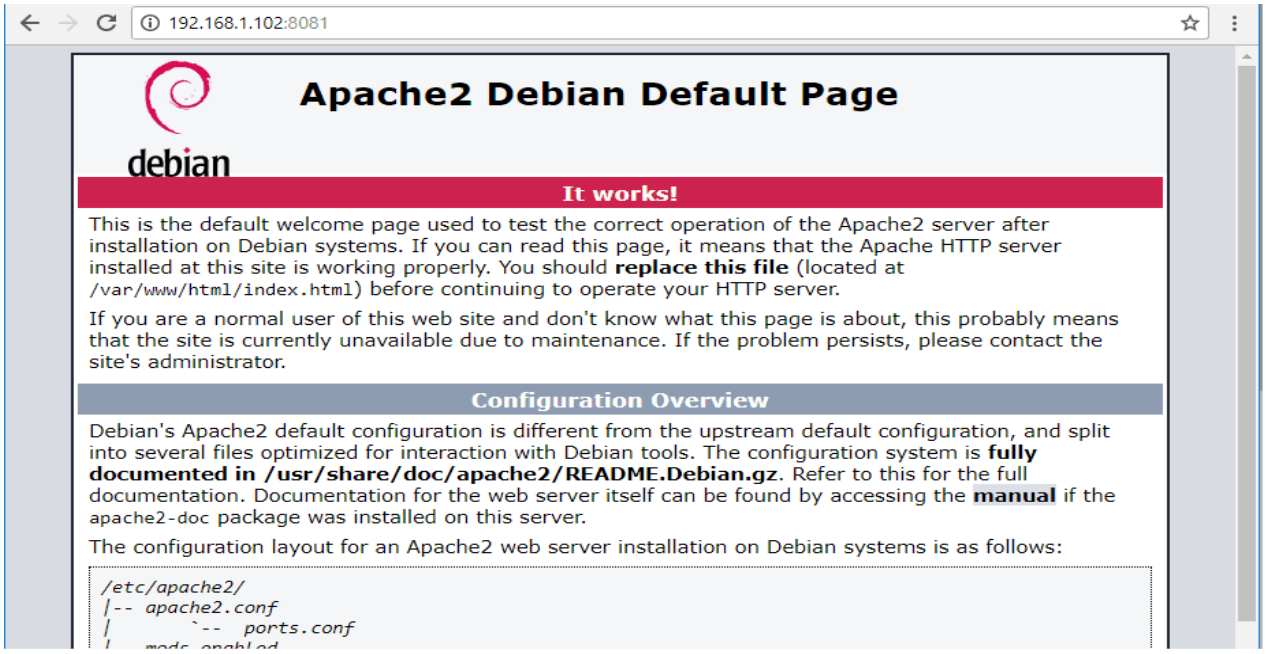
6) Other commands:

$ sudo systemctl stop apache2.
⇒ Stop the web server.

$ sudo systemctl start apache2
⇒ start the web server.

7) $ sudo systemctl restart apache2.
⇒ to restart web server.

OUTPUT :



Practical 8 : Installation and study of Hadoop Platform.

Practical 8

Title. Installation & study of Hadoop platform.

What is Hadoop?

An apache hadoop software library is a framework that allows for distributed processing of charge data sets across clusters of computers using simple programming models. It is designed to scale up from simple server to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high availability, the library itself is designed to detect & handle failure at the application layer, so delivering a highly available service on top of cluster of computers each of which may be prone to failure.

The Hadoop platform has several benfits which makes it the platform of choice for big data analytics. Hadoop is flexible and cost effective, as it has the ability to store and process huge amount of any kind of data (structured, Unstructured) quickly and effectiently by using a cluster of commodity hardware.

Installation :-

$ tar xzf hadoop-x.y.z.tar.gz

$ export JAVA_HOME = /usr/lib/jvm/java-6-sun

$ export HADOOP-INSTALL = /home/hadoop-x.y.z

$ export PATH = $PATH : $HADOOP_INSTALL/bin:
HADOOP_INSTALL/sbin

$ hadoop -version

* Starting the Hadoop Cluster

$ hdfs namenode -format.

format the namenode before using it for the first time As HDFS user run the below command format the namenode

Starting the DFS startup script to start HDFS.
./start-dfs.sh

Starting the yarn services to start YARN
./start-yarn.sh.

Practical 9 : Installation and study of Hadoop Cluster.

Aim: Installation & study of Hadoop Cluster

Theory: Getting Started

We create a Hadoop Master server with 4 vCPUs, 4GB of memory and 40 GB of hard drive space.

We will also create 2 Hadoop Nodes with 4 vCPUs, 8 GB of memory & 40 GB of hard drive space for each node.

Make sure that you configure each server with a static IP address & either Internal DNS resolution or add each server to /etc/hosts file.

Preparing the Hadoop Servers :-
First we need to install Oracle Java 8:

```
$add-apt-repository ppa:webupd8team/java
$apt update
$apt install -y oracle-java8-set-default
```

Accept the licence terms.

Next download Hadoop Binaries
$wget URL

Next we need to update our default environment variables to include the JAVA_HOME & Hadoop binary directories.

First we need to know where Java was installed to. Run the following command to find out:

```
$ update-alternative --display java
```

open /etc/environment & update the PATH line to include the Hadoop binary directories

Also add a line for the JAVA_HOME variable:

```
JAVA_HOME="/usr/lib/jvm/java-8-oracle/jre"
```

Next we will add a hadoop user and give them the correct permissions.

```
$ adduser hadoop
$ usermod -aG hadoop hadoop
$ chown hadoop:root -R /usr/local/hadoop
$ chmod g+rwx -R /usr/local/hadoop
```

Login as the hadoop user and generate an SSH key. You only need to complete this step on the Hadoop Master.

```
$ su - hadoop
$ ssh-keygen -t rsa.
```

Accept all defaults for ssh-keygen

Configuring the Hadoop Master :-

Open the /usr/local/hadoop/etc/hadoop/core-site.xml file & edit the file.

Save & exit

Next, open the /usr/local/hadoop/etc/hadoop/hdfs-site.xml file & edit the file.

open the /usr/local/hadoop/etc/hadoop/workers file and add these 2 lines:

hadoop2.addintome.lab
hadoop3.admintomelab

Copy the configuration files to each of your Hadoop Nodes from your Hadoop master.

Format the HDFS file system

$ source /etc/environment
$ hdfs namenode -format

Now you can start HDFS:

$ start-dfs.sh

Validate that everything started right by running the jps command as the Hadoop user on all your Hadoop servers

On the Hadoop Master you should see this: jps
and on each of your Hadoop Nodes you should see: jps.

HDFS Web UI :-

You can now access the HDFS web UI by browsing to yours Hadoop Master Server port 9870:

http://hadoop1.admintome.lab:9870.

Starting YARN:-

Hadoop on its own, can schedule any jobs so we need to run yarn so we can schedule jobs on our Hadoop cluster.

Run this command to start yarn.

$start-yarn.sh

We can verify that it started correctly by running this command.

yarn nod -list.

Hadoop Web UI

You can view the Hadoop Web UI by going to URL:

http://hadoop1.admintome.lab:8088/cluster.

Practical 10 : Study of MapReduce Application to process weather report dataset.

Practical 10

Title : Study of Nap Reduce Application to process weather report dataset

MapReduce :-

Map Reduce is a Progsamming model for data processing

Hadoop can run NapReduce progsams written in various languages. We look at the same progsams expressed in Java, Ruby, Python and C++ most important, Mapreduce progsam are inherently parallel, thus putting very large scale data analysis into the hands of anyone with enough maintainer.

Weather Dataset

What's the highest recorded global temp for each year in dataset? to take advantage of parallel processing that Hadoop provides we need to express our query as a mapReduce job.

MapReduce works by breaking the processing into two prosses: This help in map phase & the reduce phase.

Each phase has key value pairs as input and output the type of which may be chosen by the progsammer

Practical 11 : Design and development of a Distributed application using EJB

Practical 11

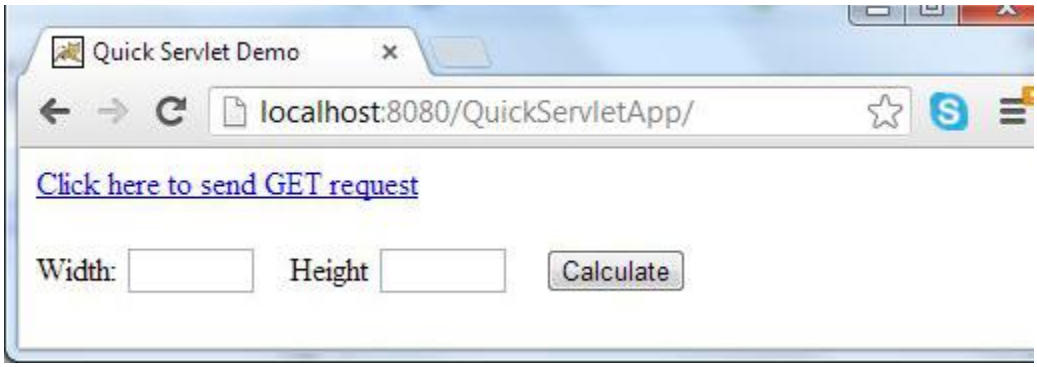
Title: Study and Implementation of distributed system using EJB.

Enterprise Java Beans:

EJB is one of several java APIs for modular construction of enterprise software EJB is a server side software component that encapsulates business logic of an application. An EJB web container provide runtime environment for web related software component, including computer security, java servelet lifecyck mangement transaction processing and other web services. The EJB specification is subset of Java EE specification The EJB specification details how an application server provides following responsibilities.

- Transaction processing
- Integration with the persistance service offered
- Concurrency Control
- Event-driven programming using Jakarta messaging & Jakarta Connector
- Asynchronous method invocations
- Job scheduling.
- Security.
- Deployment of software component in an application server.

Output :


Quick Servlet Demo
localhost:8080/QuickServletApp/
Click here to send GET request
Width:
Height
Calculate

Practical 12 : Case Study Google as a Distributed system.

Practical 12

Title :- Case Study of Google as a distributed System.

Theory

Google has diversified as well as providing a search engine is now a major player in cloud computing.

- 88 billion queries a month by the end of 2010
- The user can expect query result in 0.2 sec.

Good performance in terms of scalability, reliability, performance & openness

* Google Search Engine

It consists of a set of services

1. Crawling
2. Indexing
3. Ranking.

* Googles physical Model

Physical Architecture of Google is constructed as following

- Comodity PCs are organised in racks with between 40 to 80 PCs in a given rack. Each rack has a ethernet switch

- 30 or more racks are organised into cluster, which are key unit of management for placement & replication of services. each cluster has 2 switched connected the outside would or other data centers

- Clusters are housed in data centers that spread around the world

Key Requirements
1. Scalability
2. Reliability
3. Performance
4. Openness

Google Infrastructure

- The underlying communication paradigms including services for both remote invocation and indirect communication
- Data and coordination services providing unstructured and semi-structured interaction for storage of data coupled with services to support access to the data
- Distributed computation services providing means for carrying out parallel and distributed computation over the physical infrastructure.