**1** Write a distributed application for registering and logging in on a server using RPC mechanism. Below are the services provided by the remote procedures that you would write:

1. Register on the server using a „username‟ and a „password‟.

2. Log in to the server using a „username‟ and „password‟.

The usernames and passwords should be stored in a database. The database can be a file which the server can read from and write to. It is up to you to decide how the database should be. This database should contain a username „admin‟ by default and a password of your choice.

When the server gets request for registration:

a. If the username is already in the database, send a message to the client saying “Username already exists. Try a different one” (without quotes).

b. If the username is not in the database, send a message to the client saying “Registration successful” and create a corresponding entry in the database.

After the registration is successful, then the client program should ask the user if he/she wants to login. If yes, then follow the flow mentioned below.

When the server gets request for a login:

a. If the username and password match, send a message to the client saying “Welcome <username>!” (without quotes) where <username> is the username entered by the user.

b. If the username and password match for the user „admin‟, send a message to the client saying “Welcome admin! Number of registered users are <number\_of\_users>”, where <number\_of\_users> are the number of users in the database.

The program should exit as soon as the flow for login is complete. Write code for printing corresponding messages received by the server appropriately. Also, attach screenshots showing the ports on which the rpc server and client are bound to.

**Answer:** I have used JSch - Java Secure Channel to call the Remote procedure which is on a linux server in cloud.

String command= "sh /root/test/test\_kinkar.sh"; // "8/13/2015 7:45:18 AM"

//String command= "moshell";

Session session = **null**;

**try** {

session = jsch.getSession(user2,"10.184.43.100", 22);

session.setPassword(passworduser);

java.util.Properties config = **new** java.util.Properties();

config.put("StrictHostKeyChecking", "no");

session.setConfig(config);

} **catch** (JSchException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

The Remote procedure name is test\_kinkar.sh and the output print is like below.

USER ericPASSWORD eric123 Calling RPC method

Going to execute

USER is correct

Welcome eric!

Password of user is eric123

GET THE OUTPUT OF Remote Procedure > EXECUTION SUCCESSFULL USING RPC

Done...

The script is simple which resides in linux system.

root@pasapp ~ # cat /root/test/test\_kinkar.sh

#!/bin/bash

echo "EXECUTION SUCCESSFULL USING RPC"

root@pasapp ~ #

I have used a properties file to store the password. Sample is given below.

eric=eric123

#sample properties

#Mon Feb 15 15:26:24 IST 2016

eric123=eric1234

#

# sample properties

#Mon Feb 15 15:42:30 IST 2016

kink=kink123

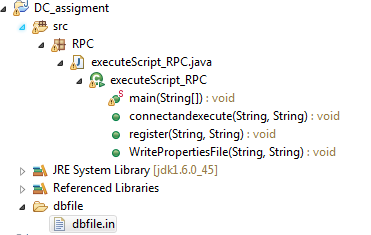
#

# sample properties

#Mon Feb 15 15:49:38 IST 2016

admin=admin123

The directory structure for property file is like below.



**Some sample output print:**

**UC-1> input in command line argument> admin admin123**

USER >admin PASSWORD >admin123

Going to register User & Password

WELCOME TO REGISTER

Registration successful with admin

**UC-2> input in command line argument> eric eric123**

USER >eric PASSWORD >eric123

Going to register User & Password

WELCOME TO REGISTER

Registration successful with eric

**UC-3> input in command line argument> kink kinkar123**

USER >kink PASSWORD >kinkar123

Going to register User & Password

WELCOME TO REGISTER

Registration successful with kink

**UC-4> input in command line argument> admin admin123**

USER >admin PASSWORD >admin123

Going to register User & Password

WELCOME TO REGISTER

Username already exists. Try a different one

**UC-5> input in command line argument> admin admin123 RPC**

USER adminPASSWORD admin123 Calling RPC method

Going to execute

USER is correct

Welcome admin!

Welcome to admin!

Number of registered users are 3

Password of user is admin123

com.jcraft.jsch.JSchException: Auth fail (as there is no admin password is not known)

**UC-6> input in command line argument> eric eric123 RPC**

USER ericPASSWORD eric123 Calling RPC method

Going to execute

USER is correct

Welcome eric!

Password of user is eric123

GET THE OUTPUT OF Remote Procedure > **EXECUTION SUCCESSFULL USING RPC**

Done...

In UC-6 we execute the RPC.

* Jar file is attached.



* The attached project export.



**Q.2** Write a distributed application which implements Lamport‟s Logical Clocks. You have to write rules for updating the logical clock at each process when an event occurs. An event could be one of two: (1) Local event, i.e. event local to the process (2) Send/Receive events involving 2 processes across the distributed system.

Each process maintains a logical clock in the form of an integer variable, which gets incremented by 1 each time an event occurs. You have to initialize this integer variable to 0 (zero). When a process sends a message to another, the integer variable at the sending process get incremented by 1 and this logical time is sent along with the message. At the receiving process, a comparison of the current logical clock value and the received logical clock value is carried out, and maximum of the two is considered. Then the receiving process increments this value by 1 and set the new value as its current logical clock value.

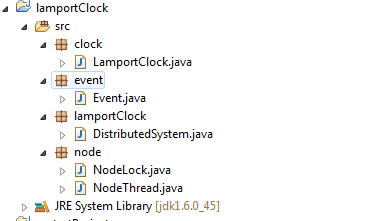
Simulate the below scenario of a distributed system comprising 3 processes (or threads). At each event (local or send/receive), the logical clock value of the process involving in the events should be printed on the console along with the process name. The format for the same is:

**<process name> : <logical clock value> : <event type>**

Where <event type> is one of the event types (local, send or receive).

Send/Receive events can be implemented by inter-thread communication (eg: via sockets). You are free to make any assumptions if you feel it is needed.

**Answer>** Directory structure is like below.



* Executable jar



* Project export



**UC-1>** Please provide input in this format: L,L,L,S,R-2,S

L: Local event

S: Send event

R: Receive event. eg: (R-2 indicates node 2 as the event sender)

Enter the events for the node 1: L,S,L,R-2,L

Enter the events for the node 2: S,S,L,R-2,L

Enter the events for the node 3: S,R-1,L,L,L

Node: 1 : clock time - 1 : Event - LOCAL

Node: 1 : clock time - 2 : Event - SEND

Node: 2 : clock time - 1 : Event - SEND

Node: 3 : clock time - 1 : Event – SEND

**UC-2>**

Please provide input in this format: L,L,L,S,R-2,S

L: Local event

S: Send event

R: Receive event. eg: (R-2 indicates node 2 as the event sender)

Enter the events for the node 1: L,L,L,S,R-2,S

Enter the events for the node 2: L,L,S,R-3,L,L,S

Enter the events for the node 3: S,S,L,R-1,L,R-2,S,L

Node: 1 : clock time - 1 : Event - LOCAL

Node: 1 : clock time - 2 : Event - LOCAL

Node: 1 : clock time - 3 : Event - LOCAL

Node: 1 : clock time - 4 : Event - SEND

Node: 2 : clock time - 1 : Event - LOCAL

Node: 2 : clock time - 2 : Event - LOCAL

Node: 2 : clock time - 3 : Event - SEND

Node: 3 : clock time - 1 : Event - SEND