

/\*

Name: Rohit Saini

Erp: 1032200897

Panel: C

RollNo: PC-41 \*/

**ASSIGNMENT TITLE:** Design a distributed application using RPC

**AIM:** To demonstrate the use of Remote Procedure Call (RPC) using client server architecture to calculate the square of given number.

**OBJECTIVES:** To study and implement RPC and client server architecture.

**THEORY:**

Remote Procedure Call:

Basic characteristic of RPC is transparency. Transparency is of two types:

1. Syntactic Transparency: In this the RPC & LPC syntax is identical.
2. Semantic Transparency: In this RPC & LPC have same semantics.

Elements of RPC:

The basic elements of RPC code are:

1. Client
2. Client Stub
3. RPC Runtime
4. Server Stub
5. Server

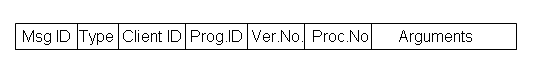
In this type of communication:

1. Client calls a remote procedure by sending packets containing parameters.
2. It is packed and sent to the client.
3. The server receives the packet, unpacks it, computes the result and sends he result back to the client

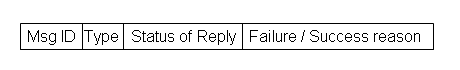
Stub Generation: Generation of strubs for client and server is of two types:

1. Manual: Provide a set of transmission functions from which user constructs his own stubs.
2. Automatic: It uses Interface Definition Language (IDL) i.e. List of procedures selected.

* RPC message format:



* RPC Reply format:



-Marshalling arguments and results:

a) Taking arguments of client and result.

b) Encoding a message.

c) Decoding the message.

**INPUT:** Number whose square is to be found.

**fact.x**

struct InputInfo {

int num1;

};

struct OutputInfo {

int result;

};

program RPCPROGRAM {

version FACTVERSION {

struct OutputInfo performAddition(struct InputInfo iInfo)=1;

}=1;

}=22222222;

**fact\_client.c**

#include "fact.h"

/\*Display function \*/

void display\_result(int a ,char \* type){

printf("%s is %d\n",type,a);

}

void

rpcprogram\_1(char \*host,int a)//added

{

CLIENT \*clnt;

struct OutputInfo \*result\_1;

struct InputInfo performaddition\_1\_arg;

performaddition\_1\_arg.num1=a; // added

#ifndef DEBUG

clnt = clnt\_create (host, RPCPROGRAM, FACTVERSION, "udp");

if (clnt == NULL) {

clnt\_pcreateerror (host);

exit (1);

}

#endif /\* DEBUG \*/

result\_1 = performaddition\_1(&performaddition\_1\_arg, clnt);

if (result\_1 == (struct OutputInfo \*) NULL) {

clnt\_perror (clnt, "call failed");

}

else{

display\_result(result\_1->result,"Factorial"); //added

}

#ifndef DEBUG

clnt\_destroy (clnt);

#endif /\* DEBUG \*/

}

int

main (int argc, char \*argv[])

{

char \*host;

int a;

if (argc < 2) {

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

exit (1);

}

host = argv[1];

/\* added \*/

printf("\nEnter num : ");

scanf("%d",&a);

rpcprogram\_1 (host,a);

exit (0);

}

**fact\_server.c**

#include "fact.h"

struct OutputInfo \*

performaddition\_1\_svc(struct InputInfo \*argp, struct svc\_req \*rqstp)

{

static struct OutputInfo result;

int i =1;

int fact=1;

/\*

\* insert server code here

\*/

while(i<=argp->num1){

fact=fact\*i;

i++;

}

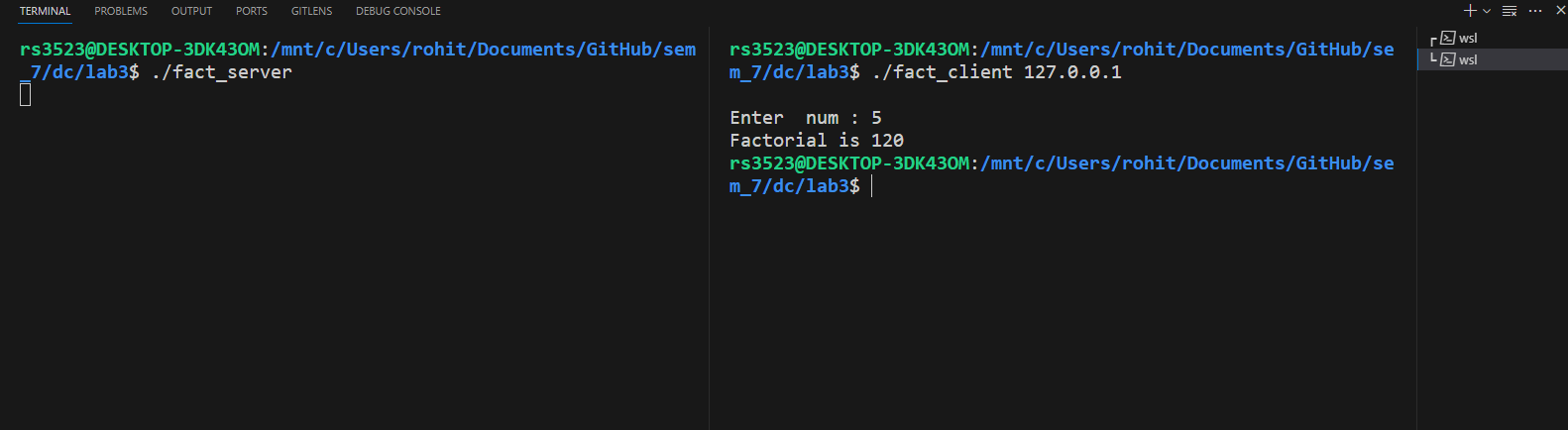
result.result=fact;

printf("Hello from Server");

return &result;

}

**OUTPUT:** Square of number.



**PLATFORM:** Linux.

**CONCLUSION:** Thus, RPC has been studied and implemented on Linux platform.

**FAQs :** 1. What is the difference between RPC and LRPC

1. What does rpcgen do ?
2. What is meant by packing and unpacking of RPC messages

