```
/*
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:

a) Syntactic Transparency: In this the RPC & LPC syntax is identical.
b) Semantic Transparency: In this RPC & LPC have same semantics.

Elements of RPC:
The basic elements of RPC code are:

a) Client
b) Client Stub
c) RPC Runtime
d) Server Stub
e) Server

In this type of communication:

a) Client calls a remote procedure by sending packets containing parameters.
b) It is packed and sent to the client.
c) 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:

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

- RPC message format:

Msg ID	Type	Client ID	Prog.ID	Ver.No.	Proc.No	Arguments

- RPC Reply format:

Msg ID	Type	Status of Reply	Failure / Success reason

-Marshalling arguments and results:
a) Taking arguments of client and result.
b) Encoding a message.
c) Decoding the message.

<u>INPUT:</u> 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;
```

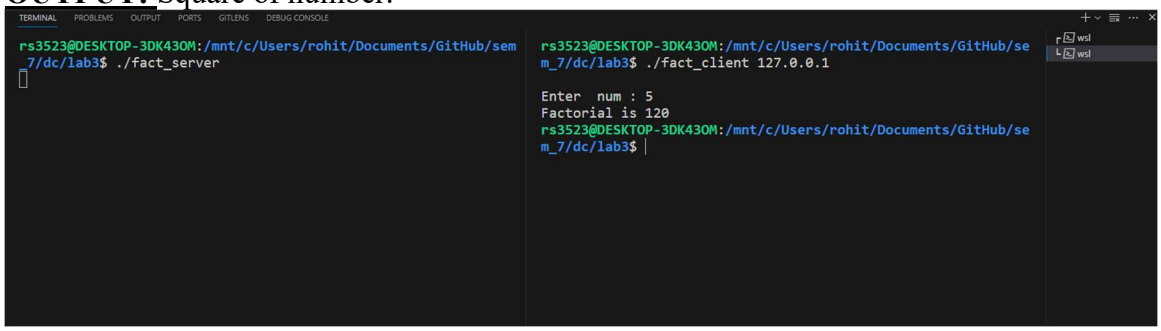
```
   int i =1;
   int fact=1;

   /*
    * 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

2. What does rpcgen do ?

3. What is meant by packing and unpacking of RPC messages

Page No.
Date

DC Lab 3

FAQ:

Q1. What is the difference between RPC and LRPC.

Aspect	RPC	LRPC
Scope of Communication:	Between processes on different machines in the network.	· Between processes on the same machine.
Communication overhead:	Involves network protocols, potentially higher overhead	· Lower overhead as it uses optimized local communiation mechanisms.
Usage:	Suitable for distributed systems where processes run on different machines.	· Optimized for local communication within a single machine.
Example:	Used in client-server applications across a network	· Suitable for inter-process communicati-on within the same machine.

Q2. What does rpcgen do?

Ans. rpcgen is a tool used to generate stub code for Remote Procedure call (RPC) programming. It simplifies the process of developing client-server applications that communicate using RPC. rpcgen function:

1. Interface Definition : rpcgen takes an interface definition file as input, this file describe procedure, their parameters and data types.
2. Code generation: rpcgen generates server and client-side stub code in C based on interface ~~design~~. definition.

Q3 What is meant by packing and unpacking of RPC messages.

Ans: Packing : Packing refers to the process of converting data from a high-level representation to a format suitable for transmission over the network.

Unpacking: Unpacking is the reverse process of packing. It involves taking the serialized data received over the ~~proc~~ network and converting it back into high level representation that the receiving program can understand.