Assignment – 5

Simulate RPC (Create procedure on remote machine and call it from local machine)

List of programs for RPC

1. Find out the factorial of given number.
2. Implement Calculator (Basic operation).
3. Find out whether given number is Prime Number or not.
4. Print out the Fibonacci series till the given number.
5. Find the maximum value of an array of integers using RPC.

Steps to run RPC program.

Step:1

First you need to install **rpcbind** in your machine. To check whether it is installed or not use the command

```
:-$ rpcinfo
```

This will show you the registration information as below in the terminal if it is successful.

Step:2

If it is not installed use the following command to automatically download and install rpcbind. After that type rpcinfo to verify the installation.

```
:-$ sudo apt-get update
```

```
:-$ sudo apt-get install rpcbind
```

Step 3

To create the client stub and server stub we have to write the IDL file first. Following is a simple IDL file which is used to add two numbers in the server end.

```
/* This is the IDL file -- name it as add.x*/

 /*combine the arguments to be passed to the server side in a structure*/

struct numbers{

    int a;

    int b;

};

program ADD_PROG{

  version ADD_VERS{

      int add(numbers)=1;

  }=1;
}=0x4562877;
```

Step 4

Now we should compile this IDL file by using the following command. Note that you should execute this command in the directory where the IDL file resides. '-C ' in the command is for C language and '-a' is to generate all the files including samples.

```
:-$ rpcgen -a -C add.x
```

It will generate the following files.

- add.h -> header file
- add_client.c -> client program
- add_clnt.c -> client stub
- add_server.c -> server program
- add_svc.c -> server skeleton
- add_xdr.c -> XDR routines used by both the client and the server
- Makefile.add -> Makefile

If we run the server and the client now, it won't show you anything because server doesn't know what to do with a client request whenever it receives one. Also client side doesn't know how to interact with the server side. So we have to edit both client and server programs

Step:5

You need to change the **add_server.c** (server program) file as follow.

```
#include "add.h"

int *

add_1_svc(numbers *argp, struct svc_req *rqstp)

{

            static int  result;

            printf("add(%d,%d) is called\n",argp->a,argp->b);

            result = argp->a + argp->b;

            return &result;

}
```

Edit the auto generated **add_client.c** (client program) file as follow

```
#include "add.h"

void

add_prog_1(char *host, int x, int y)

{

        CLIENT *clnt;

        int  *result_1;

        numbers  add_1_arg;



#ifndef   DEBUG

        clnt = clnt_create (host, ADD_PROG, ADD_VERS, "udp");

if (clnt == NULL) {

                clnt_pcreateerror (host);

                exit (1);

        }



#endif    /* DEBUG */

    add_1_arg.a=x;

    add_1_arg.b=y;

        result_1 = add_1(&add_1_arg, clnt);

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

                clnt_perror (clnt, "call failed");
```

```
          }
     else{
       printf("Result:%d\n", *result_1 );
     }
#ifndef   DEBUG
          clnt_destroy (clnt);
#endif     /* DEBUG */
int
main (int argc, char *argv[])
{
          char *host;

          if (argc < 4) {
                    printf ("usage: %s server_host\n", argv[0]);
                    exit (1);
          }
          host = argv[1];
          add_prog_1 (host, atoi(argv[2]), atoi(argv[3]));
exit (0);
}
```

Step:6

Since we've changed those two files now we have to compile all the files again. Use the following command to do that. It will generate all the object files.

```
:-$ make -f Makefile.add
```

Now open up two terminals and run the server in a one and client in the other.

```
To start server -->    :-$ sudo ./add_server

To start client -->    :-$ sudo ./add_client localhost 15 20
```