Addis Ababa Science and Technology University

Collage of Electrical and Mechanical Engineering

Department of Electrical and Computer Engineering

Computer Engineering Stream

Distributed Systems

Assignment 2: RPC Client Server Application

By

Sileshi Nibret **GSR 217/12**

Submitted to: Dr. Solomon.

June 25, 2020

Introduction

Remote Procedure Call (RPC) is a protocol that one program can use to request a service from a program located in another computer on a network without having to understand the network's details. RPC is used to call other processes on the remote systems like a local system. It is used for client-server applications.

RPC mechanisms are used when a computer program causes a procedure or subroutine to execute in a different address space, which is coded as a normal procedure call without the programmer specifically coding the details for the remote interaction. This procedure call also manages low-level transport protocol, such as UDP, TCP/IP protocol etc. It is used for carrying the message data between programs.

When making a remote procedure call:

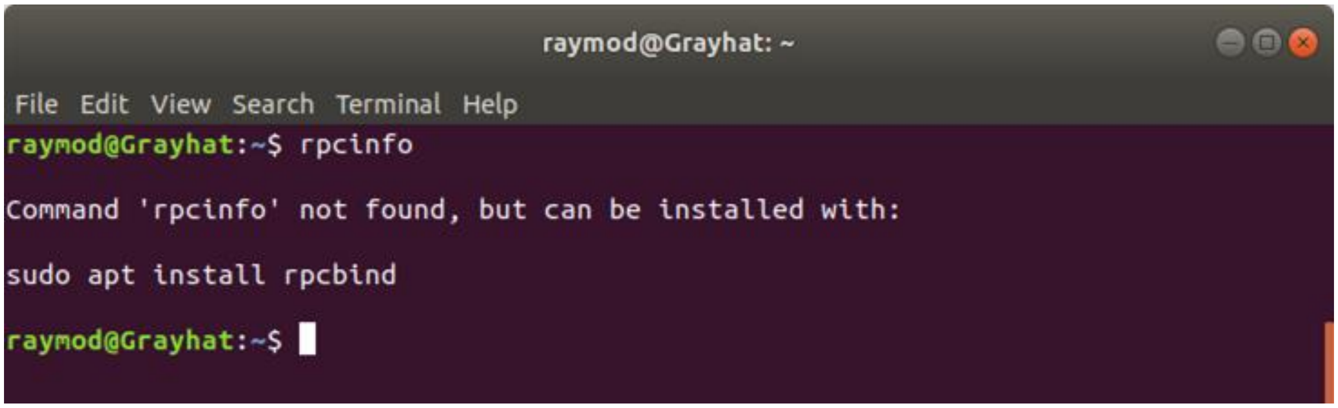
1) The calling environment is suspended, procedure parameters are transferred across the network to the system where the procedure is to execute, and the procedure is executed there.
2) When the procedure finishes and produces its results, its results are transferred back to the calling system, where execution resumes as if returning from a regular procedure call.

Installation on Ubuntu operating system

To explore remote procedure call I can use rpcgen, which is a tool on Ubuntu OS used to write server and client application.

Steps followed to install:

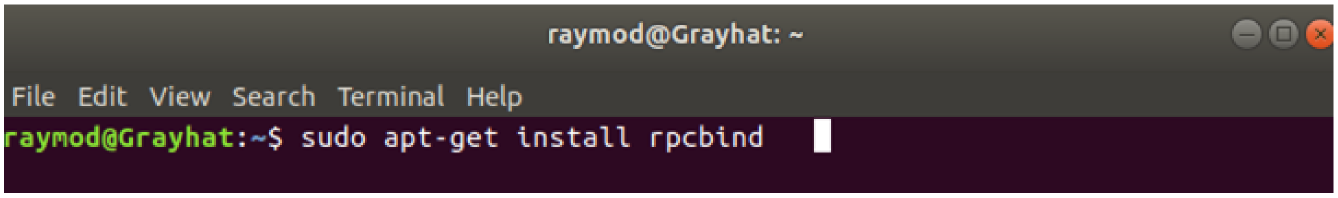
1. First checking if previously installed on my machine



```
raymod@Grayhat:~$ rpcinfo

Command 'rpcinfo' not found, but can be installed with:

sudo apt install rpcbind

raymod@Grayhat:~$
```

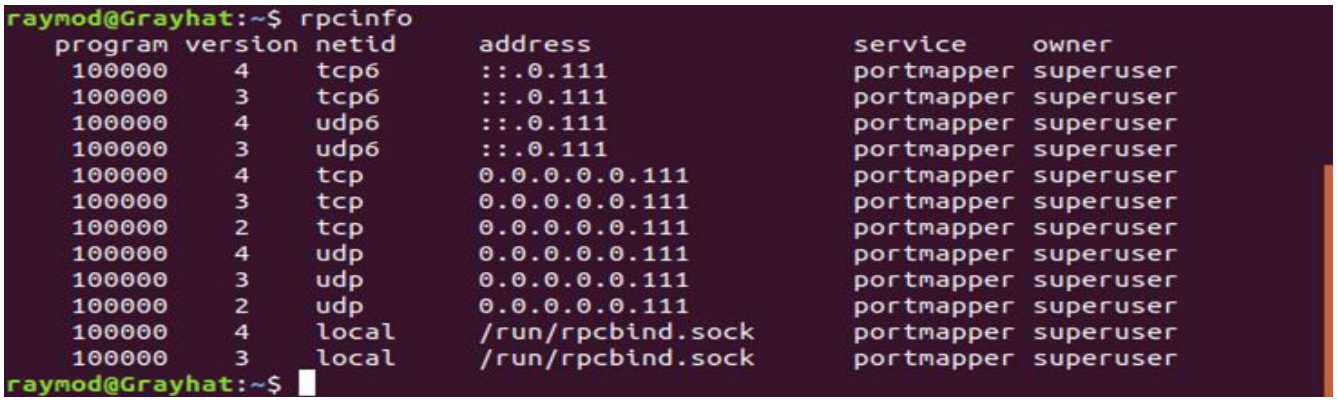
Since it was not installed it show nothing about the rpcgn info from the window then I install rpcgn

2. Then I install rpcgen as follow



```
raymod@Grayhat:~$ sudo apt-get install rpcbind
```

3. Then rpcgn is installed successfully and show the following



```
raymod@Grayhat:~$ rpcinfo
   program version netid     address                service    owner
    100000    4    tcp6      ::.0.111               portmapper superuser
    100000    3    tcp6      ::.0.111               portmapper superuser
    100000    4    udp6      ::.0.111               portmapper superuser
    100000    3    udp6      ::.0.111               portmapper superuser
    100000    4    tcp       0.0.0.0.0.111          portmapper superuser
    100000    3    tcp       0.0.0.0.0.111          portmapper superuser
    100000    2    tcp       0.0.0.0.0.111          portmapper superuser
    100000    4    udp       0.0.0.0.0.111          portmapper superuser
    100000    3    udp       0.0.0.0.0.111          portmapper superuser
    100000    2    udp       0.0.0.0.0.111          portmapper superuser
    100000    4    local     /run/rpcbind.sock      portmapper superuser
    100000    3    local     /run/rpcbind.sock      portmapper superuser
raymod@Grayhat:~$
```

As we can see from the above window when I try to run rpcinfo command it returns the rppc information like version, network id, address, services and owner. This means I am good to write the x file that can be used to write server and client application and compile it on my local system.

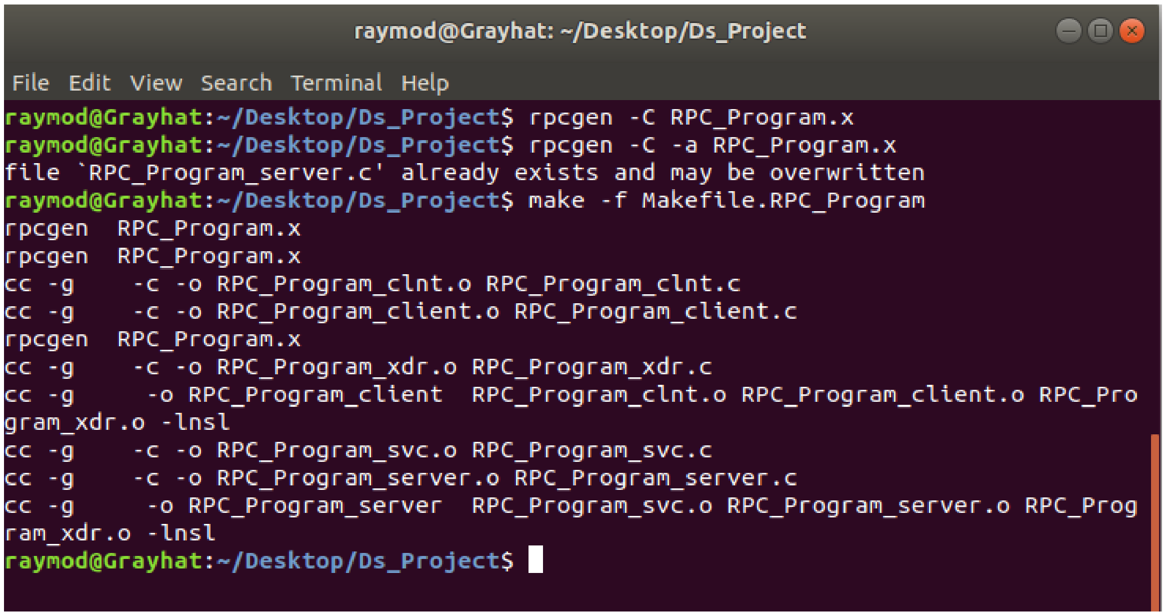
Writing the .x file

This file is generally the structure of the program it is used to rpcgen which will generate stub server and client code, which can then be used to implement an RPC server for the protocol and/or an RPC client. It can even generate example client and server code.

This file contains the returning data type, argument data type to be accepted and memory location that the procedure runs. It also used to declare the procedure.

Server/ Client program

After I defined the .x file I run the following commands to compile and generate the server and client files that are used to define and procedures.


raymod@Grayhat: ~/Desktop/Ds_Project
File Edit View Search Terminal Help

```
raymod@Grayhat:~/Desktop/Ds_Project$ rpcgen -C RPC_Program.x
raymod@Grayhat:~/Desktop/Ds_Project$ rpcgen -C -a RPC_Program.x
file `RPC_Program_server.c' already exists and may be overwritten
raymod@Grayhat:~/Desktop/Ds_Project$ make -f Makefile.RPC_Program
rpcgen  RPC_Program.x
rpcgen  RPC_Program.x
cc -g    -c -o RPC_Program_clnt.o RPC_Program_clnt.c
cc -g    -c -o RPC_Program_client.o RPC_Program_client.c
rpcgen  RPC_Program.x
cc -g    -c -o RPC_Program_xdr.o RPC_Program_xdr.c
cc -g     -o RPC_Program_client  RPC_Program_clnt.o RPC_Program_client.o RPC_Pro
gram_xdr.o -lnsl
cc -g    -c -o RPC_Program_svc.o RPC_Program_svc.c
cc -g    -c -o RPC_Program_server.o RPC_Program_server.c
cc -g     -o RPC_Program_server  RPC_Program_svc.o RPC_Program_server.o RPC_Prog
ram_xdr.o -lnsl
raymod@Grayhat:~/Desktop/Ds_Project$ 
```

After executing the above command, the rpcgen tool generates

- ✓ **Header file:** It contains the definitions common to the server and the client
- ✓ **XDR file:** This file holds routines that translate each data type defined in the header file
- ✓ **Server Stub:** program for the server (RPC_Program_server in my case) script
- ✓ **Client Stub:** A stub program for the client (RPC_Program_client in my case) script
- ✓ **Makefile:** The makefile contains all the flags and standard libraries. For example, you may want to add the -C flag to RPCGENFLAGS, or indicate that the math library should be linked. Also, If a compiler other than the default compiler gcc on most systems is to be used, you would add the notation in this section. The make utility will assume the Makefile it is to process is called makefile

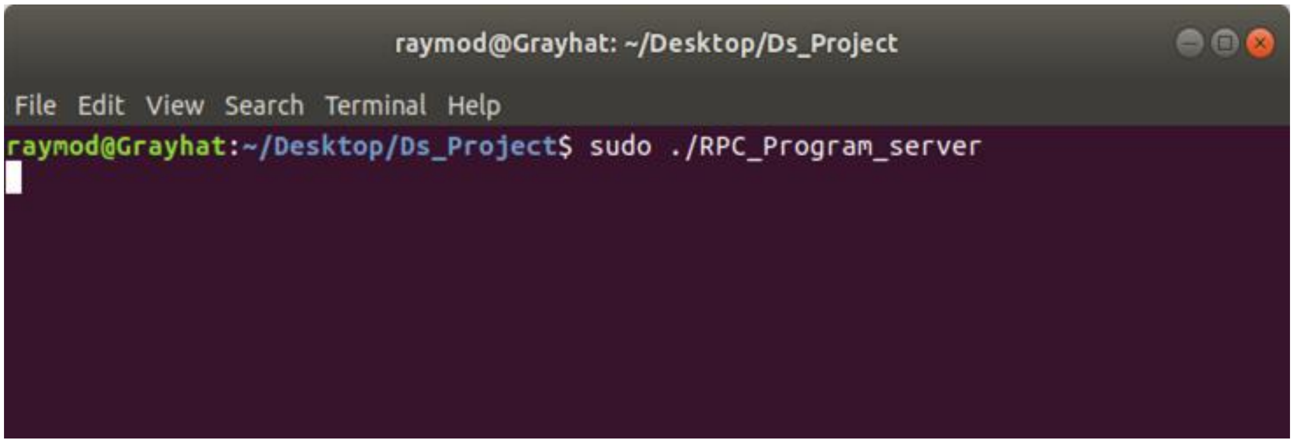
Defining Server and Client programs

Since I have to write two procedures on a server (date converter and cosine series) I defined to structures in the .x file. So, I have defined the two procedures on the server that can be invoked by the client program.

In the client side I have written a code that can accepts cosine in degree and series iteration and date in as a string of format (dd/mm/yy) and invoke the procedure defined on the local server.

Running the program

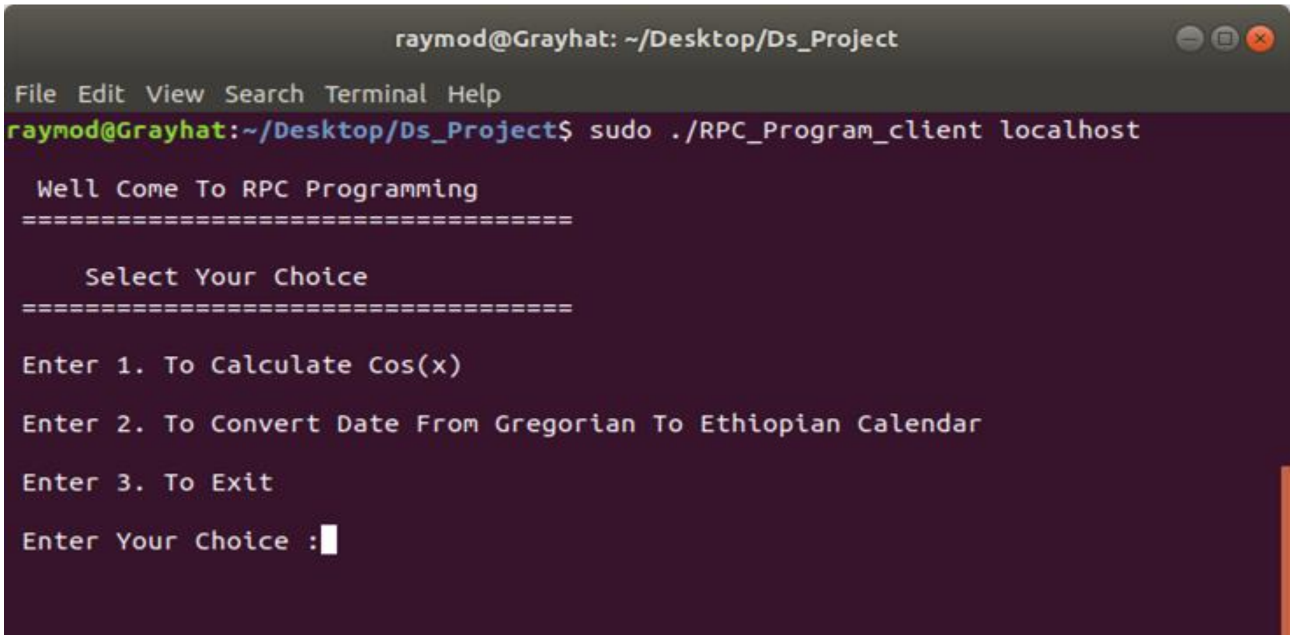
1. ***First, I run the server side***



raymod@Grayhat: ~/Desktop/Ds_Project
File Edit View Search Terminal Help

```
raymod@Grayhat:~/Desktop/Ds_Project$ sudo ./RPC_Program_server
```

2. Then running client using localhost as parameter i.e. the server is running on a localhost

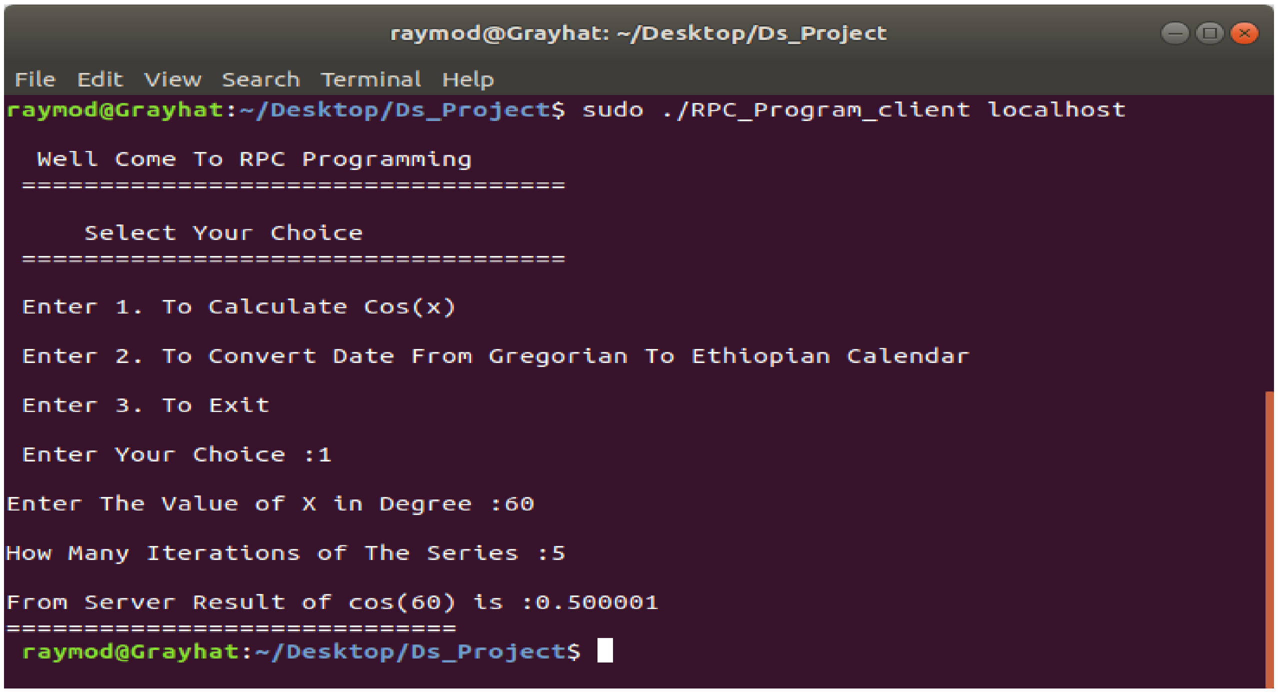


raymod@Grayhat: ~/Desktop/Ds_Project
File Edit View Search Terminal Help

```
raymod@Grayhat:~/Desktop/Ds_Project$ sudo ./RPC_Program_client localhost

 Well Come To RPC Programming
===================================

    Select Your Choice
===================================

Enter 1. To Calculate Cos(x)

Enter 2. To Convert Date From Gregorian To Ethiopian Calendar

Enter 3. To Exit

Enter Your Choice :
```

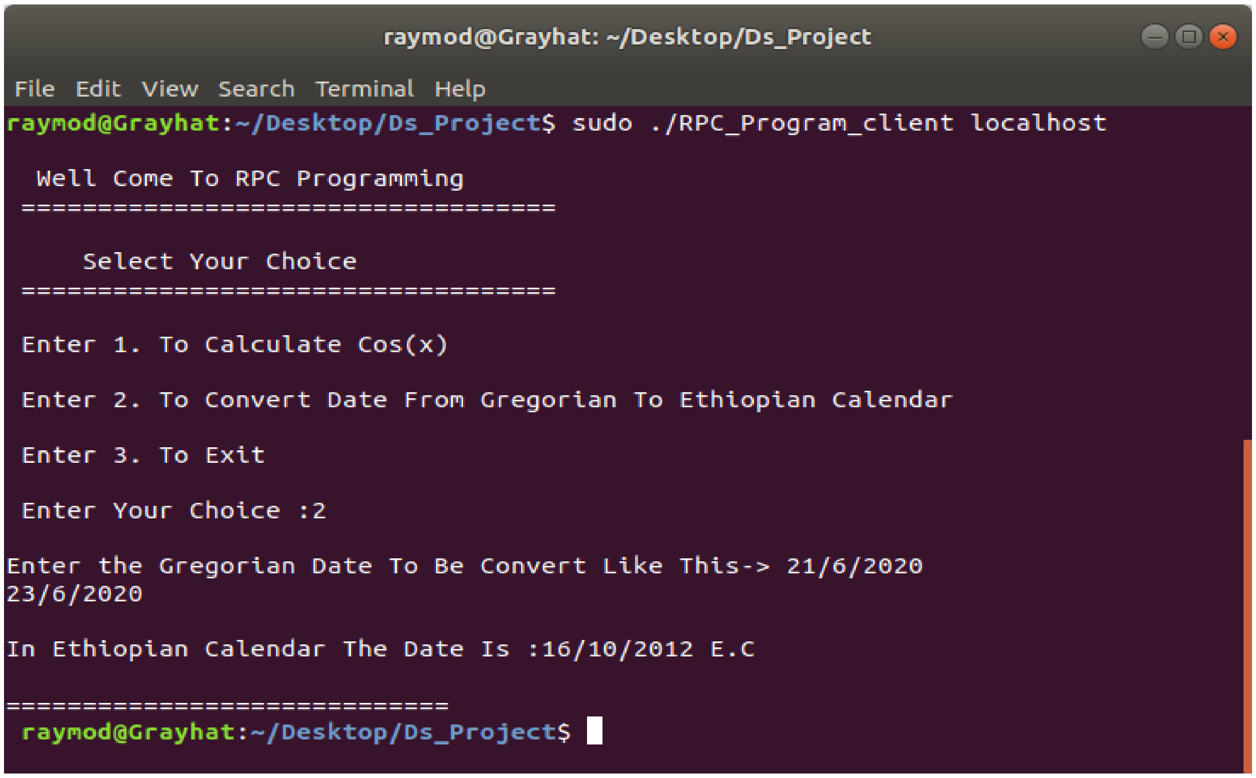
3 Selecting the task and invoke the procedure to do the task

3.1 Task 1 to calculate the cosine series


raymod@Grayhat: ~/Desktop/Ds_Project
File Edit View Search Terminal Help

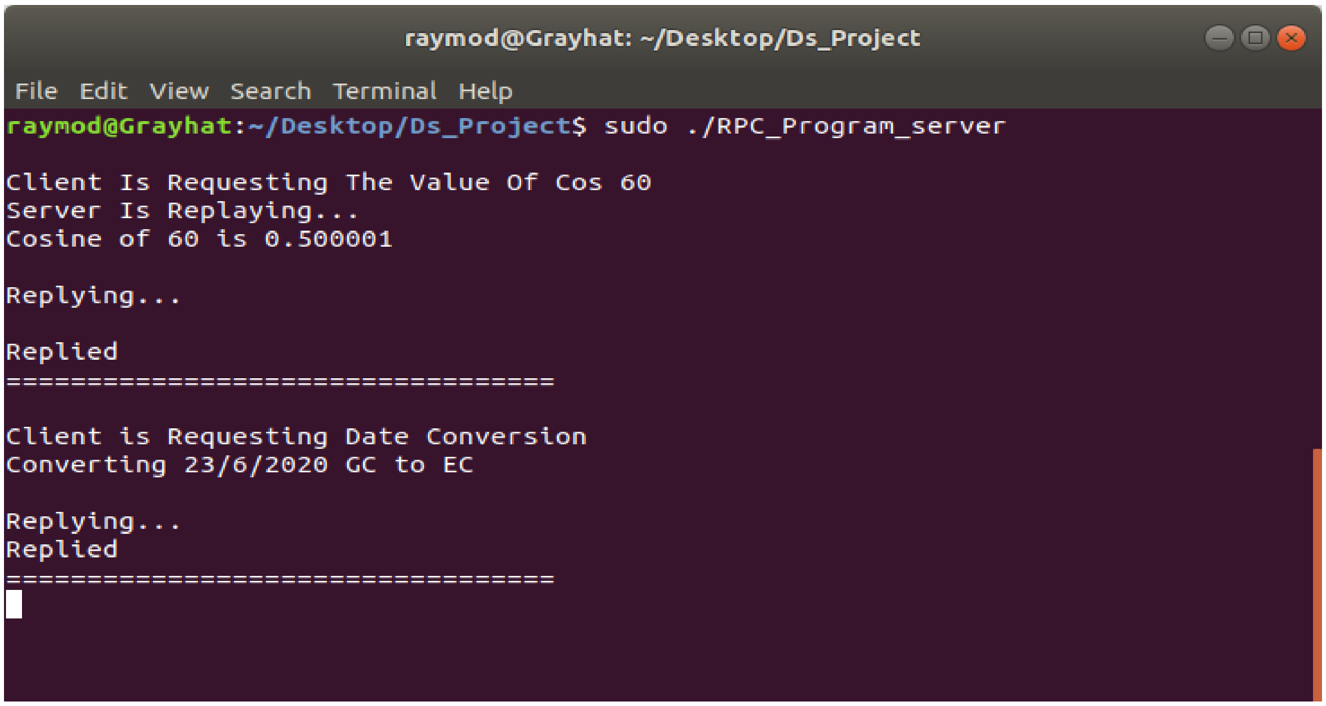
```
raymod@Grayhat:~/Desktop/Ds_Project$ sudo ./RPC_Program_client localhost

  Well Come To RPC Programming
 ===================================

     Select Your Choice
 ===================================

 Enter 1. To Calculate Cos(x)

 Enter 2. To Convert Date From Gregorian To Ethiopian Calendar

 Enter 3. To Exit

 Enter Your Choice :1

Enter The Value of X in Degree :60

How Many Iterations of The Series :5

From Server Result of cos(60) is :0.500001
=============================
 raymod@Grayhat:~/Desktop/Ds_Project$
```

3.2 Task 2 to convert Gregorian date to Ethiopian date and display as dd/mm/yy format


raymod@Grayhat: ~/Desktop/Ds_Project
File Edit View Search Terminal Help

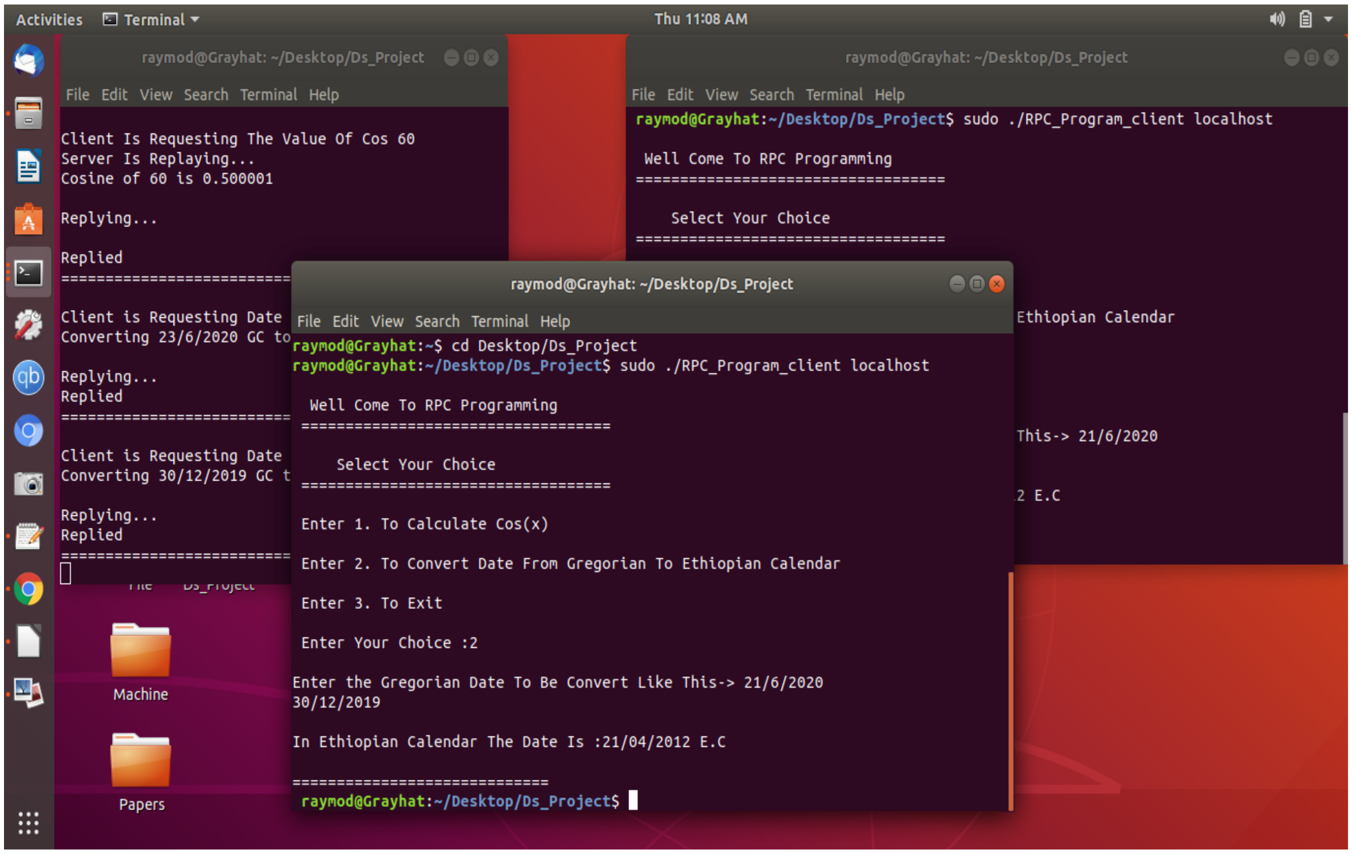
```
raymod@Grayhat:~/Desktop/Ds_Project$ sudo ./RPC_Program_client localhost

  Well Come To RPC Programming
 ===================================

     Select Your Choice
 ===================================

 Enter 1. To Calculate Cos(x)

 Enter 2. To Convert Date From Gregorian To Ethiopian Calendar

 Enter 3. To Exit

 Enter Your Choice :2

Enter the Gregorian Date To Be Convert Like This-> 21/6/2020
23/6/2020

In Ethiopian Calendar The Date Is :16/10/2012 E.C

=============================
 raymod@Grayhat:~/Desktop/Ds_Project$
```

4. Server-side replays for the above two tasks


raymod@Grayhat: ~/Desktop/Ds_Project
File Edit View Search Terminal Help

```
raymod@Grayhat:~/Desktop/Ds_Project$ sudo ./RPC_Program_server

Client Is Requesting The Value Of Cos 60
Server Is Replaying...
Cosine of 60 is 0.500001

Replying...

Replied
==================================

Client is Requesting Date Conversion
Converting 23/6/2020 GC to EC

Replying...
Replied
==================================
```

5. Running multiple clients for different tasks


Activities
Terminal
Thu 11:08 AM
raymod@Grayhat: ~/Desktop/Ds_Project
File Edit View Search Terminal Help
Machine
Papers

```
Client Is Requesting The Value Of Cos 60
Server Is Replaying...
Cosine of 60 is 0.500001

Replying...

Replied
==========================

Client is Requesting Date
Converting 23/6/2020 GC to

Replying...
Replied
==========================

Client is Requesting Date
Converting 30/12/2019 GC t

Replying...
Replied
==========================
```

```
raymod@Grayhat:~/Desktop/Ds_Project$ sudo ./RPC_Program_client localhost

 Well Come To RPC Programming
===================================

    Select Your Choice
===================================

Ethiopian Calendar

This-> 21/6/2020

2 E.C
```

```
raymod@Grayhat:~$ cd Desktop/Ds_Project
raymod@Grayhat:~/Desktop/Ds_Project$ sudo ./RPC_Program_client localhost

 Well Come To RPC Programming
===================================

    Select Your Choice
===================================

 Enter 1. To Calculate Cos(x)

 Enter 2. To Convert Date From Gregorian To Ethiopian Calendar

 Enter 3. To Exit

 Enter Your Choice :2

Enter the Gregorian Date To Be Convert Like This-> 21/6/2020
30/12/2019

In Ethiopian Calendar The Date Is :21/04/2012 E.C

==============================
 raymod@Grayhat:~/Desktop/Ds_Project$
```

Conclusion

As I showed the screen shoots while running a server and client, I can execute cosine series calculation and data conversion on a server and return the values to the clients which calls the procedures. The structure was defining on two different address spaces (1fffffff and 12345678) in which the two procedure’s arguments are reside on that memory location. Whenever the client invokes the procedures it passes the arguments to that memory segment and the server takes the needed arguments from there.