*Taller de RPC*

*Gregori Carvajal, Davianny Torres*

*Mineroromero222@gmail.com, Daviannyt11@gmail.com*

*Sistemas Distribuidos – Universidad Nacional Experimental de Guayana*

**Resumen**

.

En el presente informe se dará a conocer los resultados obtenidos de la práctica de RPC en la cual se realizaron las pruebas de comunicación remota mediante una aplicación cliente-servidor basada en RPC, el cliente implementará la interface de usuario y el servidor el procesamiento. Esto se llevará a cabo por medio de la programación de una calculadora básica con las siguientes funciones: suma, resta, multiplicación, división, promedio, potencia y raíz cuadrada.

# Introducción

En computación distribuida, la llamada a procedimientos remotos (en inglés, Remote Procedure Call, RPC) es un programa que utiliza una computadora para ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambas. El protocolo que se utiliza para esta llamada es un gran avance sobre los sockets de Internet usados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando estas encapsuladas dentro de las RPC.

Las RPC son muy utilizadas dentro de la comunicación cliente-servidor. Siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando este de vuelta el resultado de dicha operación al cliente. En la actualidad nos encontramos cada vez con más volúmenes de datos para procesar; donde los recursos de un solo hardware no cubren la demanda en procesamiento.

# Método Experimental

Para el desarrollo del código calcu\_cliente.c y calcu\_server.c se usó la herramienta **rpcgen** que nos genera código C para implementar un protocolo RPC. La entrada a rpcgen es un lenguaje similar a C conocido como RPC Language (Lenguaje de llamada a procedimiento remoto).

Al ejecutar rpcgen nos crea algunos archivos fundamentales entre ellos:

* **calcu\_svc.c.** Este es un ejemplo concreto de servidor. Registra al servidor en el sistema unix para indicarle que atienda a las llamadas y proporciona un “**switch-case**” para llamar a cada una de las funciones al recibir la petición del cliente de la operación a realizar.

**Ejemplo:**

switch (rqstp->rq\_proc) {

case NULLPROC:

(void) svc\_sendreply (transp, (xdrproc\_t) xdr\_void, (char \*)NULL);

return;

case SUMA:

\_xdr\_argument = (xdrproc\_t) xdr\_entrada;

\_xdr\_result = (xdrproc\_t) xdr\_float;

local = (char \*(\*)(char \*, struct svc\_req \*)) suma\_1\_svc;

break;

case RESTA:

\_xdr\_argument = (xdrproc\_t) xdr\_entrada;

\_xdr\_result = (xdrproc\_t) xdr\_float;

local = (char \*(\*)(char \*, struct svc\_req \*)) resta\_1\_svc;

break;

* **calcu\_clnt.c.** Es utilizados para las llamadas a funciones de rpc desde un cliente. Se hacen a través de la función “**clnt\_call()**”. En “**calcu\_clnt.c**” rpcgen incluye unas funciones de “traducción” para hacernos más sencillas las llamadas desde nuestro cliente.

**Ejemplo:**

float \*

suma\_1(entrada \*argp, CLIENT \*clnt)

{

static float clnt\_res;

memset((char \*)&clnt\_res, 0, sizeof(clnt\_res));

if (clnt\_call (clnt, SUMA,

(xdrproc\_t) xdr\_entrada, (caddr\_t) argp,

(xdrproc\_t) xdr\_float, (caddr\_t) &clnt\_res,

TIMEOUT) != RPC\_SUCCESS) {

return (NULL);

}

return (&clnt\_res);

}

* **calcu\_xdr.c.** Como RPC permite llamadas de clientes a servidores que estén en máquinas distintas y, por tanto, puedan tener una arquitectura distinta, es necesario traducir los parámetros y resultados a un “código” universal, independiente de las máquinas. Si los parámetros son tipos básicos (int, float, char, etc), el sistema unix ya tiene unas funciones de conversión (xdr\_int(), xdr\_float(), etc). Si los parámetros, como en este caso, son estructuras definidas por nosotros, las funciones de conversión hay que hacerlas. rpcgen genera automáticamente dichas funciones y en nuestro caso, están incluidos en el fichero calcu\_xdr.c.

**Ejemplo:**

bool\_t

xdr\_entrada (XDR \*xdrs, entrada \*objp)

{

register int32\_t \*buf;

if (!xdr\_float (xdrs, &objp->arg1))

return FALSE;

if (!xdr\_float (xdrs, &objp->arg2))

return FALSE;

return TRUE;

}

bool\_t

xdr\_operacion (XDR \*xdrs, operacion \*objp)

{

register int32\_t \*buf;

int i;

if (!xdr\_entrada (xdrs, &objp->op))

return FALSE;

if (!xdr\_vector (xdrs, (char \*)objp->simbolo, 3,

sizeof (char), (xdrproc\_t) xdr\_char))

return FALSE;

if (!xdr\_float (xdrs, &objp->resultado))

return FALSE;

return TRUE;

}

* **calcu.h.** En este fichero de definen las funciones. Cualquier cliente que quiera usarlas, deberá hacer un include de este fichero. El prototipo no es exactamente como esperaríamos. Define también otras constantes como nombre de programa, variables globales, número de versión, etc, que son útiles a la hora de hacer la conexión con el servidor.

**Ejemplo:**

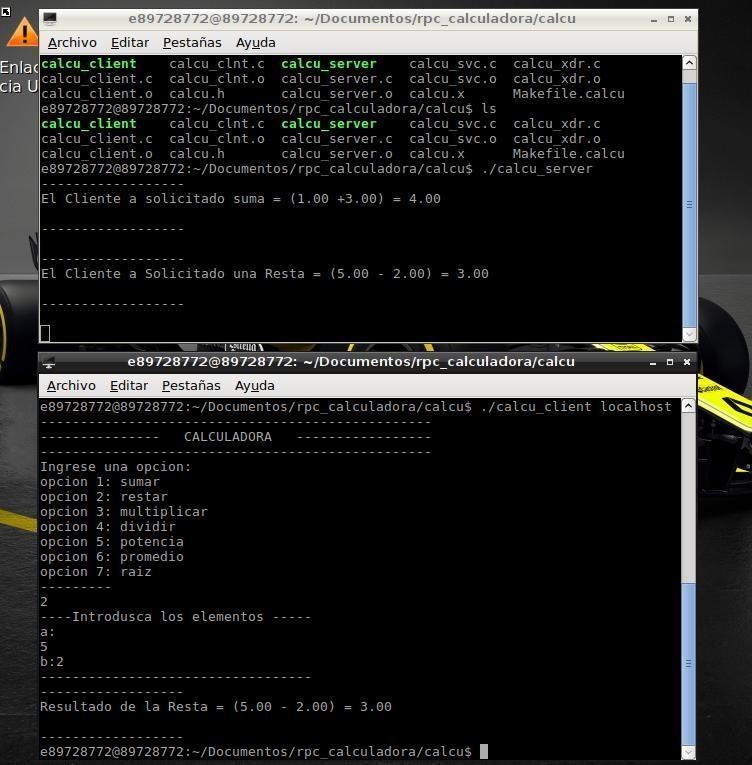
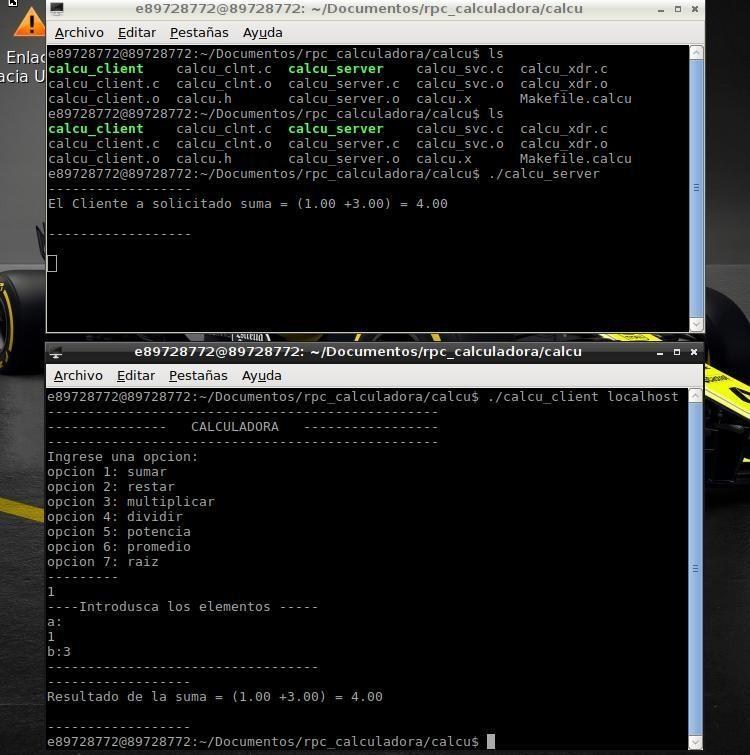
#define SUMA 1

extern float \* suma\_1(entrada \*, CLIENT \*);

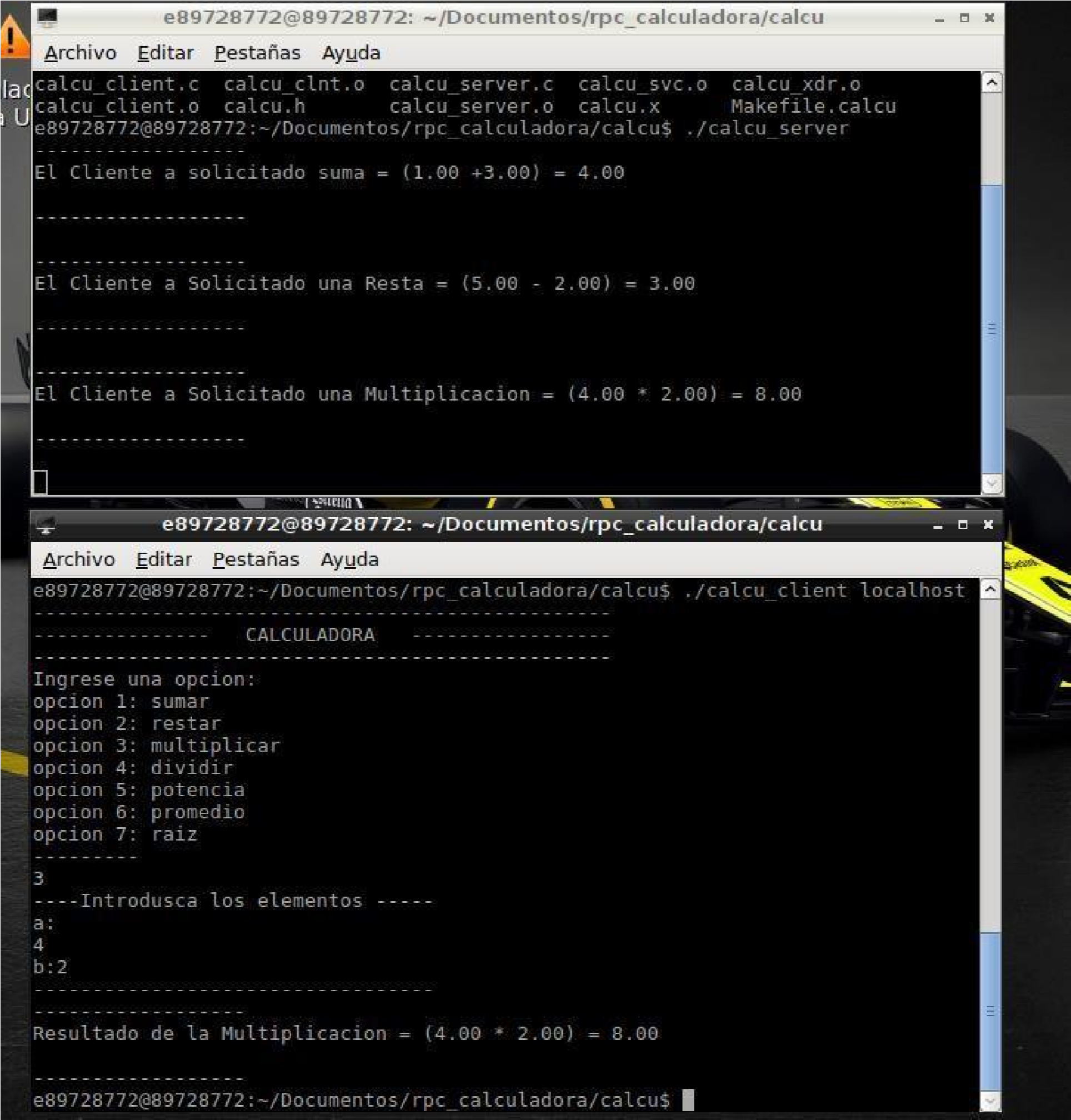
extern float \* suma\_1\_svc(entrada \*, struct svc\_req \*);

*El procedimiento que se llevó a cabo para la realización de la práctica fue el siguiente:*

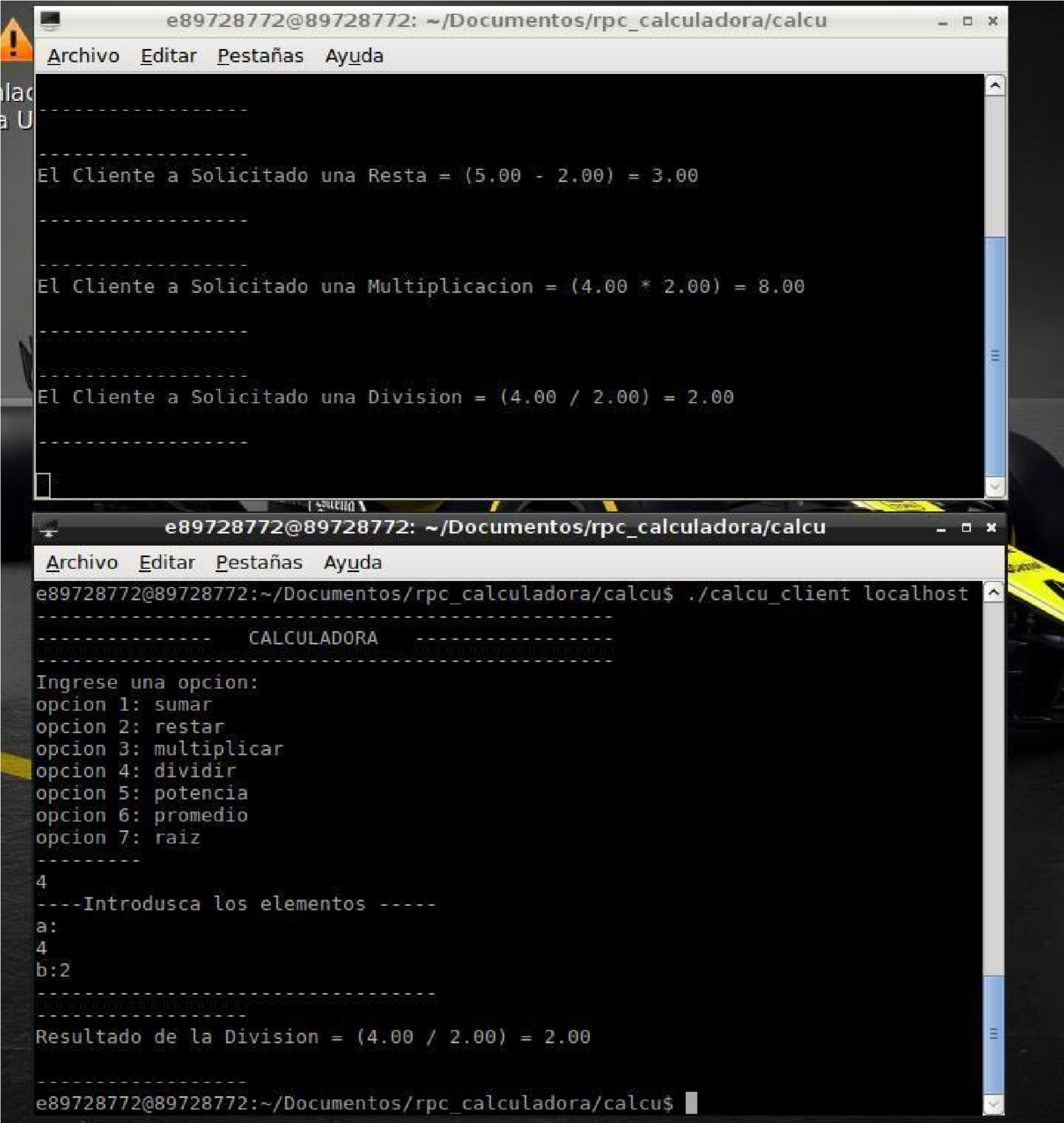
*Figura 1. Se estableció el cliente y servidor, el Servidor cumple la función de realizar los cálculos.*



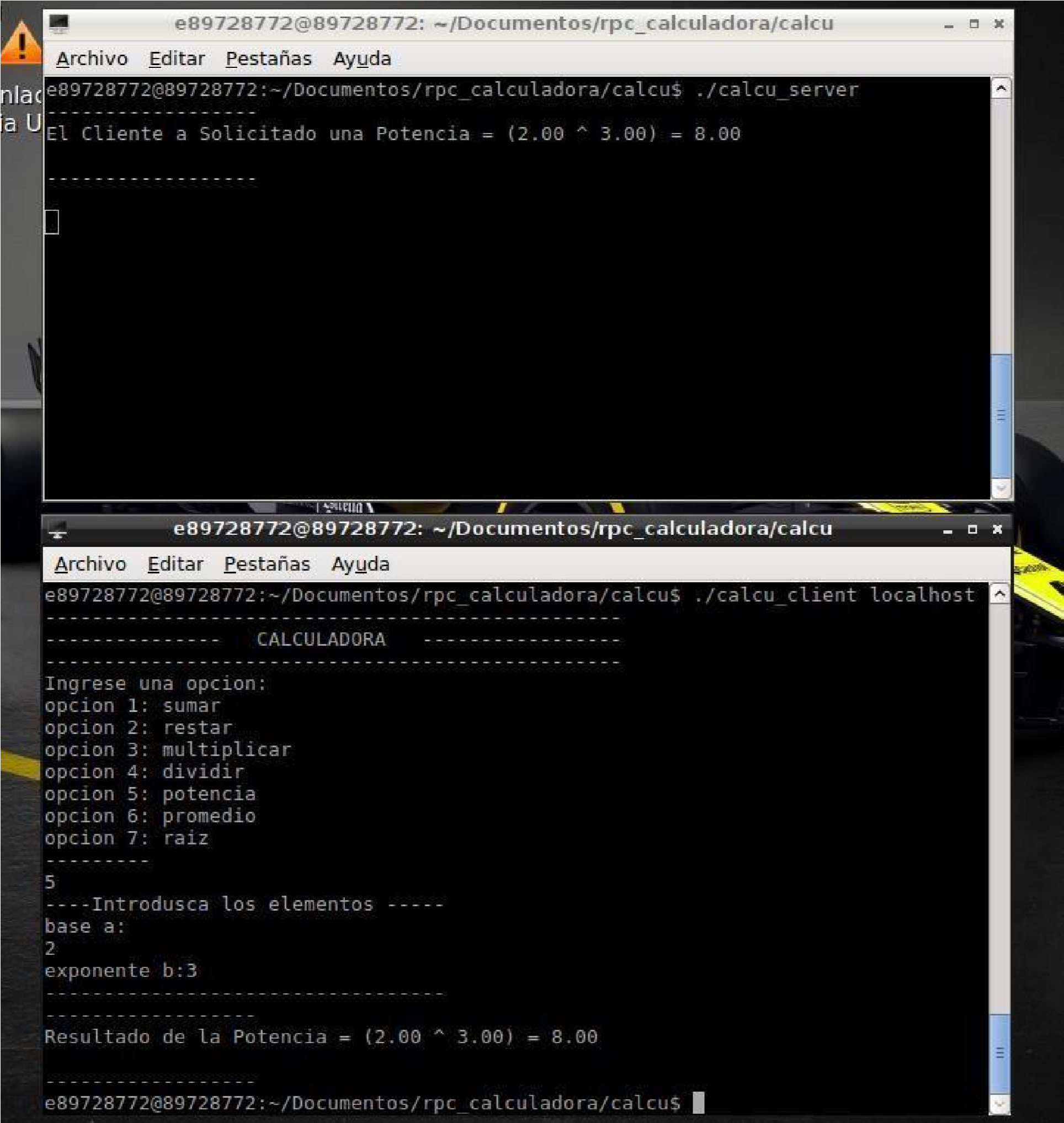
*Sumar, restar, multiplicar, dividir, potencia, promedio y raíz.*



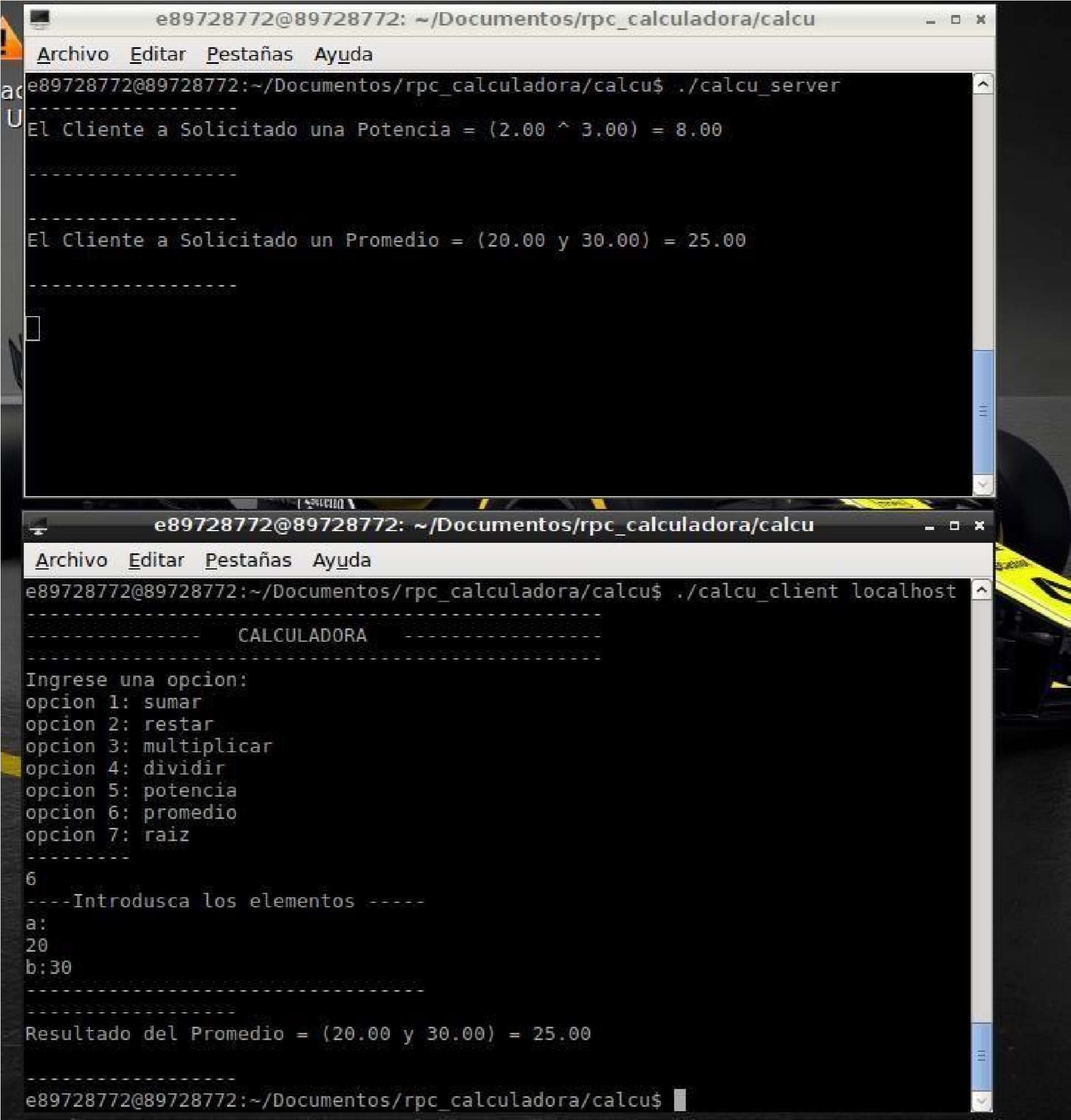
*Figura 2. El cliente permite seleccionar el tipo de operación a realizar y realiza una multiplicación de ejemplo.*



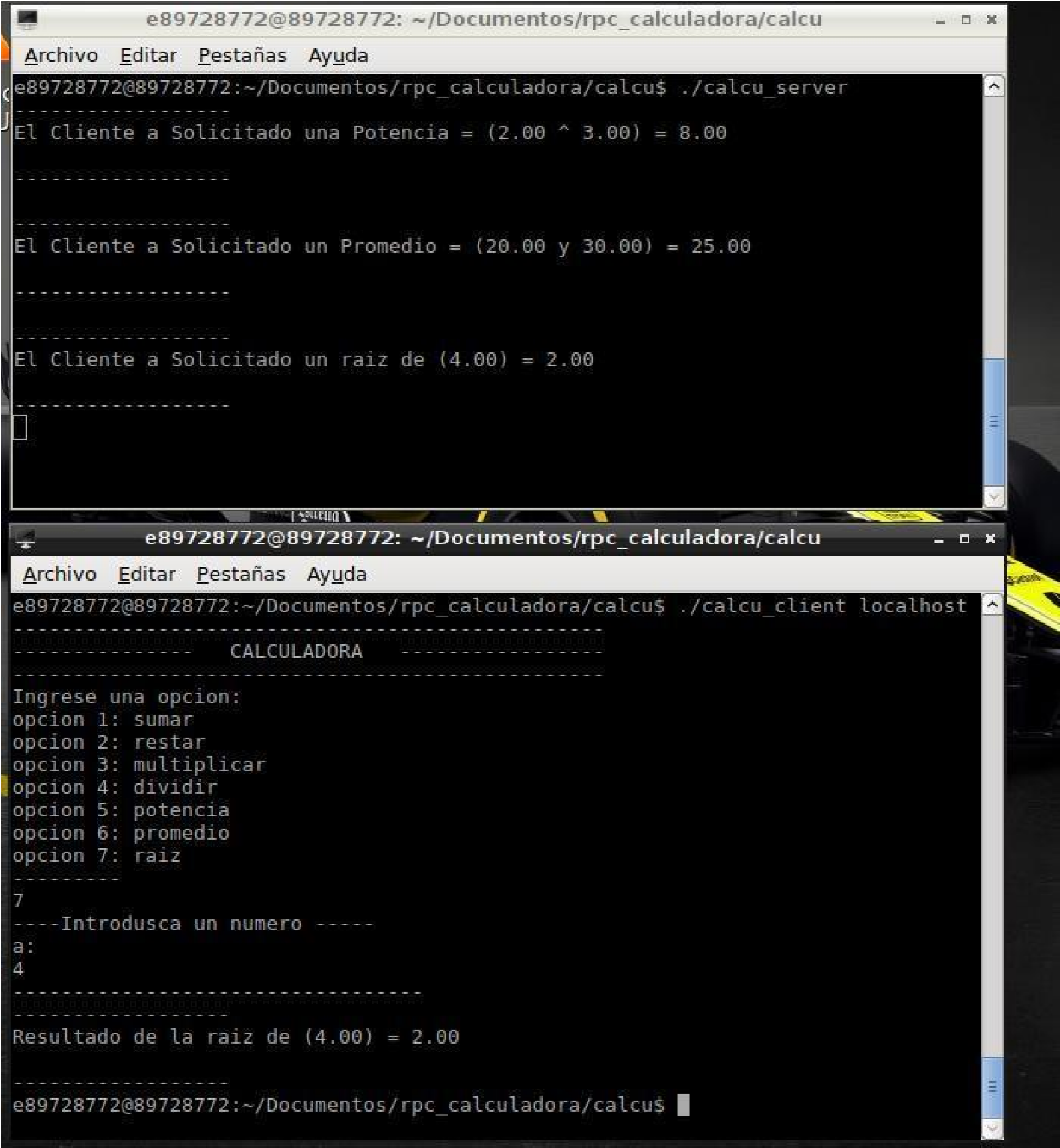
*Figura 3. El cliente permite seleccionar el tipo de operación a realizar, Obteniendo el resultado del servidor de la división de 4/2=2.*



*Figura 4. Se lleva a cabo la potencia.*



*Figura 5. Se lleva a cabo el promedio.*



*Figura 6. Se lleva a cabo la raíz.*

Instrucciones para Compilar y Ejecutar en la consola de Linux:

Compilar: make -f makefile.calcu

Ejecutar Servidor: ./calcu\_server

Ejecutar cliente: ./calcu\_cliente localhost

# Conclusión

Gracias a esta práctica pudimos ampliar nuestros conocimientos en cuanto a RPC, por medio de la realización de una sencilla calculadora, implementada por cliente y servidor. Una llamada de procedimiento remoto (RPC) consiste en un protocolo que permite a un software o programa ejecutar código en otra máquina remota sin preocuparse por la comunicación, por lo regular es bastante utilizado en el paradigma cliente y servidor. Existen varios tipos de RPC.

Un servidor RPC consiste en una colección de procedimientos que un cliente puede solicitar por el envío de una petición RPC al servidor junto con los parámetros del procedimiento. El servidor invocará el procedimiento indicado en nombre del cliente, entregando el valor de retorno, si hay alguno. Para ser independiente de la máquina, todos los datos intercambiados entre el cliente y el servidor se convierten al formato External Data Representation (XDR) por el emisor, y son reconvertidos a la representación local por el receptor. RPC confía en sockets estandard UDP y TCP para transportar los datos en formato XDR hacia el host remoto. Sun amablemente ha puesto RPC en el dominio público.

RPC es dependiente del lenguaje de programación que se utilice. Si quisiéramos utilizarlo para ofrecer algún servicio web, por ejemplo, es probable que necesitemos utilizar más de un lenguaje de programación del lado del cliente y del servidor, en ese momento RPC deja de ser una alternativa. Sin embargo, si todos los módulos del servicio que vamos a ofrecer a los clientes están basados en el mismo lenguaje de programación, RPC es una muy buena opción.

**Referencias**

* <https://codingornot.com/que-es-rpc-llamada-a-procedimiento-remoto>
* https://www.tldp.org/LDP/nag/node128.html
* https://users.cs.cf.ac.uk/Dave.Marshall/C/node33.html
* https://users.cs.cf.ac.uk/Dave.Marshall/C/node34.html