Practica 2 Sun RPC

Para la práctica he hecho dos calculadoras, las he dejado por separado para que no sea tan lioso.

- La primera es la calculadora básica que hace la suma, la resta, la multiplicación y la división. Además también le he añadido el resto ya que la división solo te daba el cociente, y la potencia, para que sea un poco diferente.

- La segunda calculadora es una calculadora de fracciones, le metemos dos fracciones y podemos sumarlas, restarlas, multiplicarlas o dividirlas.

¿Cómo he creado las calculadoras?

- calculadora.x

Primero he hecho el .x de ambas calculadoras cada uno en una carpeta:

- Este es el de la calculadora básica:

```
struct inputs{
        float num1;
        float num2;
        char operador;
};

program CALCULADORA_PROG{
        version CALCULADORA_VER{
                float ADD(inputs)=1;
                float SUB(inputs)=2;
                float MUL(inputs)=3;
                float DIV(inputs)=4;
                float RES(inputs)=5;
                float POT(inputs)=6;

        }=1;
}=0x2fffffff;
```

- Este es el de la calculadora de las fracciones:

```
struct inputs{
        float num1;
        float num2;
        float num3;
        float num4;
        char operador;
};

program CALCULADORA_PROG{
        version CALCULADORA_VER{
                float ADD(inputs)=1;
                float SUB(inputs)=2;
                float MUL(inputs)=3;
                float DIV(inputs)=4;

        }=1;
}=0x2fffffff;
```

Después de crearlo pongo en la terminal el siguiente comando (tengo que tener descargado rpcgen):

```
claudiasalado@claudiasalado-VirtualBox:~/Desktop/DSD/P2_SUNRPC$ rpcgen -a -C calculadora.x
```

Y me crea varios archivos entre ellos calculadora_client.c y calculadora_server.c que son los que tengo modificar.

- calculadora_server.c

Aquí es donde declaro que hace cada método es decir como hace la suma la resta etc.

- En la calculadora básica el server me queda tal que así:

```c
#include "calculadora.h"
#include <math.h>
#include <stdio.h>

float *
add_1_svc(inputs *argp, struct svc_req *rqstp)
{
        static float  result;

        //Ponemos aqui nuestro codigo para la suma
        result = argp->num1+argp->num2;
        printf("Got Request : Suma %f y %f \n",argp->num1,argp->num2);
        printf("Sent Response : %f \n", result);

        return &result;
}

float *
sub_1_svc(inputs *argp, struct svc_req *rqstp)
{
        static float  result;

        //Ponemos aqui nuestro codigo para la resta
        result = argp->num1-argp->num2;
        printf("Got Request : Resta %f y %f \n",argp->num1,argp->num2);
        printf("Sent Response : %f \n", result);


        return &result;
}

float *
mul_1_svc(inputs *argp, struct svc_req *rqstp)
{
        static float  result;

        //Ponemos aqui nuestro codigo para la multiplicacion
        result = argp->num1*argp->num2;
        printf("Got Request : Multiplca %f y %f \n",argp->num1,argp->num2);
        printf("Sent Response : %f \n", result);

        return &result;
}
```

```c
float *
div_1_svc(inputs *argp, struct svc_req *rqstp)
{
	static float  result;

	//Ponemos aqui nuestro codigo para la division
	result = argp->num1/argp->num2;
	printf("Got Request : Divide %f y %f \n",argp->num1,argp->num2);
	printf("Sent Response : %f \n", result);

	return &result;
}

float *
res_1_svc(inputs *argp, struct svc_req *rqstp)
{
	static float  result;
	static int aux;
	static int m;

	//Ponemos aqui nuestro codigo para la division
	aux=argp->num1/argp->num2;
	printf(" %f \n", aux);

	m=argp->num2*aux;
	result=argp->num1-m;
	printf("Got Request : La raiz %f y %f \n",argp->num1,argp->num2);
	printf("Sent Response : %f \n", result);

	return &result;
}

float *
pot_1_svc(inputs *argp, struct svc_req *rqstp)
{
	static float  result;

	//Ponemos aqui nuestro codigo para la division
	for(int i=0; i<argp->num2; i++)
		result=argp->num1*argp->num1;
	printf("Got Request : Eleva %f a %f \n",argp->num1,argp->num2);
	printf("Sent Response : %f \n", result);

	return &result;
}
```

- En la calculadora de fracciones el server me queda tal que así:

He tenido que hacer dos funciones a parte que son el mcd y el mcm, el máximo común divisor lo necesito para hacer el minimo comun multiplo, este último se utiliza para calcular la suma y resta de fracciones.

```c
#include "calculadora.h"

//Creonun metodo que haga el MCD que  nos sirve para sacar el MCM
float maximo_comun_divisor(float a, float b) {
   float temporal;//Para no perder b
   float aux;
    while (b != 0) {
        temporal = b;
        aux=a/b;
        b = a-(aux*b);
        a = temporal;
    }
    return a;
}

//Hago este metodo del MCM para que sea mas sencillo hacer la suma y la resta
float minimo_comun_multiplo(float a, float b) {
    return (a * b) / maximo_comun_divisor(a, b);
}


//SUMA
float *add_1_svc(inputs *argp, struct svc_req *rqstp){
        static float  result;

        //Todo lo necesario para sumar fracciones
        float mcm=minimo_comun_multiplo(argp->num2,argp->num4);       //Calcula el mcm de los denominadores
        float m1=mcm/argp->num2*argp->num1;                           //Calcula el numerador de la primera fraccion
        float m2=mcm/argp->num4*argp->num3;                           //Calcula el numerador de la segunda fraccion
        float m=m1+m2;                                                //Calcula el numerador total

        //Ponemos aqui nuestro codigo para la suma
        result = m/mcm;
        printf("Got Request : Suma %f/%f y %f/%f \n",argp->num1,argp->num2,argp->num3,argp->num4);
        printf("Sent Response : %f/%f = %f \n",m,mcm,result);

        return &result;
}
```

```c
//RESTA
float *sub_1_svc(inputs *argp, struct svc_req *rqstp){
        static float  result;

        //Todo lo necesario para la resta de Fracciones
        float mcm=minimo_comun_multiplo(argp->num2,argp->num4);       //Calcula el mcm de los denominadores
        float m1=mcm/argp->num2*argp->num1;                           //Calcula el numerador de la primera fraccion
        float m2=mcm/argp->num4*argp->num3;                           //Calcula el numerador de la segunda fraccion
        float m=m1-m2;                                                //Calcula el numerador total


        //Ponemos aqui nuestro codigo para la resta
        result = m/mcm;
        printf("Got Request : Resta %f/%f y %f/%f \n",argp->num1,argp->num2,argp->num3,argp->num4);
        printf("Sent Response : %f/%f = %f \n",m,mcm,result);

        return &result;
}


//MULTIPLICACION
float *mul_1_svc(inputs *argp, struct svc_req *rqstp){
        static float  result;

        //Todo lo necesario para la multiplicacion de fracciones
        float m=argp->num1*argp->num3;                                //Calcula el numerador final
        float d=argp->num2*argp->num4;                                //Calcula el denominador final

        //Ponemos aqui nuestro codigo para la multiplicion
        result = m/d;
        printf("Got Request : Multiplica %f/%f y %f/%f \n",argp->num1,argp->num2,argp->num3,argp->num4);
        printf("Sent Response : %f/%f = %f \n",m,d,result);
        return &result;
}


//DIVISION
float *div_1_svc(inputs *argp, struct svc_req *rqstp){
        static float  result;

        //Todo lo necesario para la division de fracciones
        float m=argp->num1*argp->num4;                                //Calcula el numerador final
        float d=argp->num2*argp->num3;                                //Calcula el denominador final

        //Ponemos aqui nuestro codigo para la division
        result = m/d;
        printf("Got Request : Divide %f/%f y %f/%f \n",argp->num1,argp->num2,argp->num3,argp->num4);
        printf("Sent Response : %f/%f = %f \n",m,d,result);

        return &result;
}
```

- **calculadora_client.c**

Aquí modificó la forma en la que acepta las entradas del cliente desde teclado y género los procedimientos remotos(add, sub, mul, div, etc) invocando resultados para el cliente.

- En la calculadora básica el cliente me queda tal que así:

```c
#include "calculadora.h"
#include <math.h>
#include <stdio.h>

float
calculadora_prog_1(char *host, float a, float b, char op, CLIENT *clnt)
{
	float  *result_1;
	inputs  add_1_arg;
	float  *result_2;
	inputs  sub_1_arg;
	float  *result_3;
	inputs  mul_1_arg;
	float  *result_4;
	inputs  div_1_arg;
	float  *result_5;
	inputs  res_1_arg;
	float  *result_6;
	inputs  pot_1_arg;


//SUMA
	if (op=='+'){
		add_1_arg.num1=a;		//El primer sumando va a ser el primer float que pasamos.
		add_1_arg.num2=b;		//El segundo sumando sera el segundo float.
		add_1_arg.operador=op;		//El char que pasemos sera el operador de la suma.

		result_1 = add_1(&add_1_arg, clnt);		//Calcula la suma
		if(result_1 == (float *) NULL){		//Comprueba que se hace bien la suma, si esta vacio el resultado es que no se ha hecho
			clnt_perror(clnt, "SYNTAX ERROR");	//Nos da un mensaje de error
		}

		return *result_1;		//Nos devuelve el resultado de la suma;
	}

//RESTA
	if (op=='-'){
		sub_1_arg.num1=a;		//El minuendo va a ser el primer float que pasamos.
		sub_1_arg.num2=b;		//El sustraendo sera el segundo float.
		sub_1_arg.operador=op;		//El char que pasemos sera el operador de la resta.

		//Viene por defecto al crearse con rpcgen, lo dejo asi.
		result_2= sub_1(&sub_1_arg, clnt);		//Calcula la resta
		if(result_2 == (float *) NULL){		//Comprueba que se hace bien la resta, si esta vacio el resultado es que no se ha hecho
			clnt_perror(clnt, "SYNTAX ERROR");	//Nos da un mensaje de error
		}

		return *result_2;		//Nos devuelve el resultado de la resta;
	}

```

```
//MULTIPLICACION
	if (op =='*'){
		mul_1_arg.num1=a;		//El primer factor va a ser el primer float que pasamos.
		mul_1_arg.num2=b;		//El segundo factor sera el segundo float.
		mul_1_arg.operador=op;		//El char que pasemos sera el operador de la multiplicación.

		//Viene por defecto al crearse con rpcgen, lo dejo asi.
		result_3= mul_1(&mul_1_arg, clnt);		//Calcula la multiplicacion
		if(result_3 == (float *) NULL){		//Comprueba que se hace bien la multiplicacion, si esta vacio el resultado es que no se ha hecho
			clnt_perror(clnt, "SYNTAX ERROR");	//Nos da un mensaje de error
		}

		return *result_3;		//Nos devuelve el producto de la multiplicacion;
	}

//DIVISION
	if (op =='/'){
		div_1_arg.num1=a;		//El dividendo va a ser el primer float que pasamos.
		div_1_arg.num2=b;		//El divisor sera el segundo float.
		div_1_arg.operador=op;		//El char que pasemos sera el operador de la division.

		//Tenemos que tener en cuenta que si el divisor es 0 nos tiene que dar error por tanto
		if (b == 0){
			printf("SYNTAX ERROR \n");
			exit(0);
		}else{

		//Viene por defecto al crearse con rpcgen, lo dejo asi.
			result_4= div_1(&div_1_arg, clnt);		//Calcula la division
			if(result_4 == (float *) NULL){		//Comprueba que se hace bien la division, si esta vacio el resultado es que no se ha hecho
				clnt_perror(clnt, "SYNTAX ERROR");	//Nos da un mensaje de error
			}

			return *result_4;	//Nos devuelve el cociente de la division;
		}
	}

//RESTO
	if (op =='r'){
		res_1_arg.num1=a;		//El dividendo va a ser el primer float que pasamos.
		res_1_arg.num2=b;		//El divisor sera el segundo float.
		res_1_arg.operador=op;		//El char que pasemos sera el operador del resto.

		//Tenemos que tener en cuenta que si el divisor es 0 nos tiene que dar error por tanto
		if (b == 0){
			printf("SYNTAX ERROR \n");
			exit(0);
		}else{

		//Viene por defecto al crearse con rpcgen, lo dejo asi.
			result_5= res_1(&res_1_arg, clnt);		//Calcula la division
			if(result_5 == (float *) NULL){		//Comprueba que se hace bien la division, si esta vacio el resultado es que no se ha hecho
				clnt_perror(clnt, "SYNTAX ERROR");	//Nos da un mensaje de error
			}

			return *result_5;	//Nos devuelve el resto de la division;
		}
	}

//POTENCIA
	if (op =='^'){
		pot_1_arg.num1=a;		//La base va a ser el primer float que pasamos.
		pot_1_arg.num2=b;		//El exponente sera el segundo float.
		pot_1_arg.operador=op;		//El char que pasemos sera el operador de la potencia.

		//Viene por defecto al crearse con rpcgen, lo dejo asi.
		result_6 = pot_1(&pot_1_arg, clnt);
		if (result_6 == (float *) NULL) {
			clnt_perror (clnt, "call failed");
		}

		return *result_6;		//Nos devuelve el resultado de la potencia;
	}


}
```

```c
int main (int argc, char *argv[]){
        //Añadimos las variables que faltan
        char *host;
        float a, b;
        char op;
        CLIENT *clnt;

        if (argc < 2) {
                printf ("usage: %s server_host\n", argv[0]);
                exit (1);
        }

        //Creamos un pequeño menu
        printf("Bienvenido estas usando la calculadora de Claudia \n");
        printf("Recuerda: \n + para la SUMA \n - para la RESTA \n * para la multiplicacion \n / para la division \n porcentaje para el resto \n ^ para la potencia ");
        printf("Indique el primer numero \n");
        scanf("%f", &a);
        printf("Indique el segundo numero \n");
        scanf("%f", &b);
        printf("Indique el operador \n");
        scanf("%s", &op);

        //Viene por defecto al crearse con rpcgen, lo dejo asi.
        host = argv[1];
        clnt = clnt_create (host, CALCULADORA_PROG, CALCULADORA_VER, "udp");
        if (clnt == NULL) {
                clnt_pcreateerror (host);
                exit (1);
        }
        //Imprimimos el resultado
        printf("El resultado es: %f \n",calculadora_prog_1(host,a,b,op,clnt));

        //Viene por defecto al crearse con rpcgen, lo dejo asi.
        clnt_destroy (clnt);
        exit (0);
}
```

- En la calculadora de fracciones el cliente me queda tal que así:

```c
#include "calculadora.h"


float
calculadora_prog_1(char *host, float a, float b, float c, float d,char op, CLIENT *clnt)
{
        float  *result_1;
        inputs  add_1_arg;
        float  *result_2;
        inputs  sub_1_arg;
        float  *result_3;
        inputs  mul_1_arg;
        float  *result_4;
        inputs  div_1_arg;
        float  *result_5;
        inputs  pot_1_arg;

//SUMA
if(op=='+'){

        add_1_arg.num1=a;
        add_1_arg.num2=b;
        add_1_arg.num3=c;
        add_1_arg.num4=d;
        add_1_arg.operador=op;

        result_1 = add_1(&add_1_arg, clnt);
        if (result_1 == (float *) NULL) {
                clnt_perror (clnt, "call failed");
        }
        return *result_1;
}

//RESTA
if(op=='-'){

        sub_1_arg.num1=a;
        sub_1_arg.num2=b;
        sub_1_arg.num3=c;
        sub_1_arg.num4=d;
        sub_1_arg.operador=op;

        result_2 = sub_1(&sub_1_arg, clnt);
        if (result_2 == (float *) NULL) {
                clnt_perror (clnt, "call failed");
        }
        return *result_2;
}
```

```c
//MULTIPLICACION
if(op=='*'){

        mul_1_arg.num1=a;
        mul_1_arg.num2=b;
        mul_1_arg.num3=c;
        mul_1_arg.num4=d;
        mul_1_arg.operador=op;

        result_3 = mul_1(&mul_1_arg, clnt);
        if (result_3 == (float *) NULL) {
                clnt_perror (clnt, "call failed");
        }
        return *result_3;
}

//DIVISION
if(op=='/'){

        div_1_arg.num1=a;
        div_1_arg.num2=b;
        div_1_arg.num3=c;
        div_1_arg.num4=d;
        div_1_arg.operador=op;

        result_4 = div_1(&div_1_arg, clnt);
        if (result_4 == (float *) NULL) {
                clnt_perror (clnt, "call failed");
        }
        return *result_4;
}

}

int
main (int argc, char *argv[])
{
        //Añadimos las variables que faltan
        char *host;
        float a, b, c, d;
        char op;
        CLIENT *clnt;

        if (argc < 2) {
                printf ("usage: %s server_host\n", argv[0]);
                exit (1);
        }

        //Creamos un pequeño menu
        printf("Bienvenido estas usando la calculadora de Claudia \n");
        printf("Recuerda: \n + para la SUMA \n - para la RESTA \n * para la multiplicacion \n / para la division \n");
        printf("Indique el numerador de la primera fraccion \n");
        scanf("%f", &a);
        printf("Indique el denominador de la primera fraccion \n");
        scanf("%f", &b);
        printf("Indique el numerador de la segunda fraccion \n");
        scanf("%f", &c);
        printf("Indique el denominador de la segunda fraccion \n");
        scanf("%f", &d);
        printf("Indique el operador \n");
        scanf("%s", &op);

        //Viene por defecto al crearse con rpcgen, lo dejo asi.
        host = argv[1];
        clnt = clnt_create (host, CALCULADORA_PROG, CALCULADORA_VER, "udp");
        if (clnt == NULL) {
                clnt_pcreateerror (host);
                exit (1);
        }

        //Imprimimos el resultado
        printf("El resultado es: %f \n",calculadora_prog_1(host,a,b,c,d,op,clnt));

        //Viene por defecto al crearse con rpcgen, lo dejo asi.
        clnt_destroy (clnt);
        exit (0);
}
```

- ## Funcionamiento

Para comprobar cómo funcionan hacemos el siguiente comando en ambas lo mismo:

```
claudiasalado@claudiasalado-VirtualBox:~/Desktop/DSD/P2_SUNRPC/Calculadora_Fracciones$ make -f Makefile.calculadora
cc -g    -c -o calculadora_xdr.o calculadora_xdr.c
cc -g     -o calculadora_client  calculadora_clnt.o calculadora_client.o calculadora_xdr.o -lnsl
cc -g    -c -o calculadora_svc.o calculadora_svc.c
cc -g    -c -o calculadora_server.o calculadora_server.c
```

Se nos van a crear dos ejecutables calculadora_client y calculadora_server ejecutamos ambos cada uno en una terminal.

```
claudiasalado@claudiasalado-VirtualBox:~/Desktop/DSD/P2_SUNRPC/Calculadora_Fr
cc
ones$ sudo ./calculadora_client localhost
[sudo] password for claudiasalado:
```

```
claudiasalado@claudiasalado-VirtualBox:~/Desktop/DSD/P2_SUNRPC/Calculadora_Fracc
ones$ sudo ./calculadora_server
[sudo] password for claudiasalado:
```

Ahora pruebo las calculadoras:

- ## Calculadora Básica

Voy a probar con una potencia.
En el cliente nos aparece el menú y ponemos los numero en este caso primero la base después el exponente y el operador y nos devuelve el resultado.

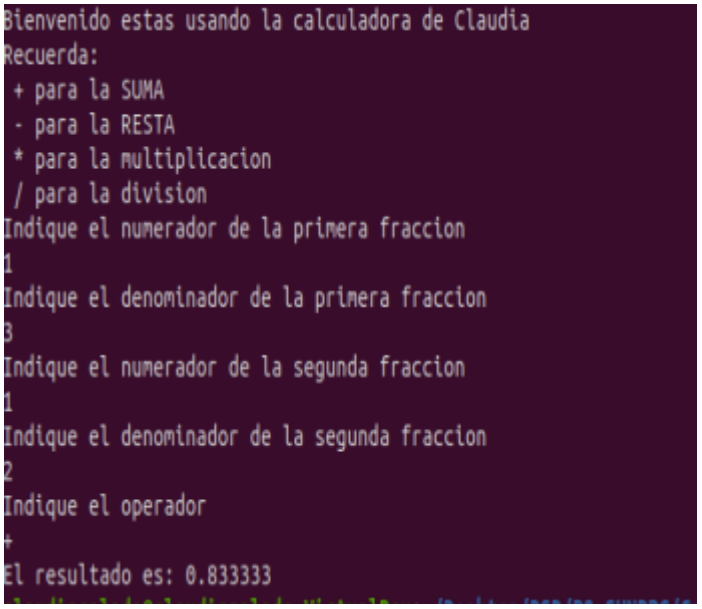
```
Bienvenido estas usando la calculadora de Claudia
Recuerda:
 + para la SUMA
 - para la RESTA
 * para la multiplicacion
 / para la division
 r para el resto
 ^ para la potencia
Indique el primer numero
5
Indique el segundo numero
2
Indique el operador
^
El resultado es: 25.000000
```

En el server nos muestra la operación que pedimos que realice y el resultado que manda al cliente.

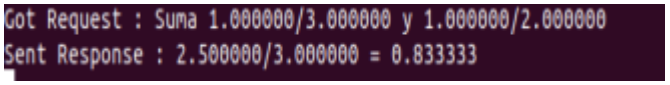
```
Got Request : Eleva 5.000000 a 2.000000
Sent Response : 25.000000
```

- **Calculadora de Fracciones**

Voy a probar una suma.
En el cliente nos aparece el menú y ponemos las fracciones y el operador y nos devuelve el resultado.



En el server nos muestra la operación que pedimos que realice y el resultado que manda al cliente.



Nota:

Al enviar los archivos fuera del zip de cada calculadora los he tenido que renombrar para identificandolos con calculadora_basica y calculadora_fracciones.
Si se cogen esos seguramente no funcionen ya que tienen nombres distintos a como vienen en el makefile.

Otra cosa que me he dado cuenta al final es que en el básico cuando puse los .txt y los .c fuera del zip hay una cosa del menú que cambia con respecto a los del zip. pone “porcentaje para el resto” mientras que en los del zip pone “r para el resto”, hay que poner r no el símbolo del porcentaje.