

## Memoria Práctica 2 - Sun RPC

El ejercicio consiste en implementar un programa distribuido que realice las operaciones básicas de una calculadora:

- El cliente realiza las llamadas al servidor dependiendo de la operación elegida.
- El servidor contiene las operaciones y realiza los cálculos.

El entorno de ejecución y desarrollo de la práctica es una máquina virtual con el sistema operativo Ubuntu 20.04. He realizado las operaciones básicas pedidas de la calculadora, es decir, la suma, resta, multiplicación y división de enteros. Los ejemplos están ejecutados en local con dos terminales abiertas simultáneamente (una emulando el cliente y otra el servidor).

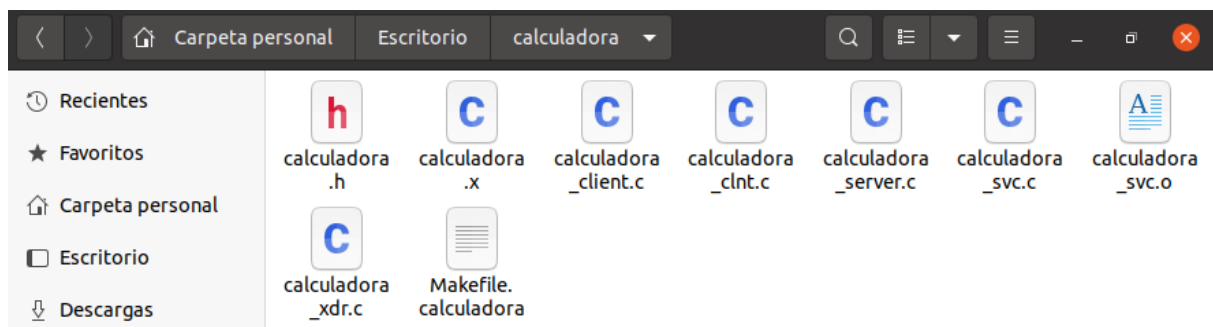
En el archivo *calculadora.x* he definido el programa, la versión y las funciones así como una estructura auxiliar llamada *inputs* que almacena los dos operandos enteros para una mayor comodidad y limpieza en el código.

```
struct inputs {
    int num1;
    int num2;
};

program CALCULADORAPROG {
    version CALCULADORAVER {
        int SUMA(inputs)=1; /* Numero de procedimiento (único entre los que hay) */
        int RESTA(inputs)=2;
        int MULTIPLICACION(inputs)=3;
        float DIVISION(inputs)=4;
    } =1; /* Numero de version */
} = 0x200000ff; /* Numero de programa */
```

Código de *calculadora.x*

Tras definir el programa, generamos los archivos necesarios para su implementación y ejecución distribuida con la herramienta *rpcgen* (*rpcgen -NCa calculadora.x*) al igual que vimos en el ejemplo de directorios de la práctica.



Archivos generados por *rpcgen*

Una vez hecho esto, pasamos a implementar el código de los dos programas: *calculadora\_client.c* y *calculadora\_server.c*. En el lado del servidor implementamos las operaciones suma, resta, multiplicación y división.

```
#include "calculadora.h"

int *
suma_1_svc(inputs arg1, struct svc_req *rqstp)
{
    static int result;

    printf("calculating %i + %i\n", arg1.num1, arg1.num2);
    result = arg1.num1 + arg1.num2;

    return &result;
}

int *
resta_1_svc(inputs arg1, struct svc_req *rqstp)
{
    static int result;

    printf("calculating %i - %i\n", arg1.num1, arg1.num2);
    result = arg1.num1 - arg1.num2;

    return &result;
}

int *
multiplicacion_1_svc(inputs arg1, struct svc_req *rqstp)
{
    static int result;

    printf("calculating %i x %i\n", arg1.num1, arg1.num2);
    result = arg1.num1 * arg1.num2;

    return &result;
}

float *
division_1_svc(inputs arg1, struct svc_req *rqstp)
{
    static float result;

    // La comprobación se hace antes de realizar la llamada, en el cliente
    printf("calculating %i / %i\n", arg1.num1, arg1.num2);
    result = (float) arg1.num1 / (float) arg1.num2;

    return &result;
}
```

Código de *calculadora\_server.c*

En el cliente implementamos el main donde tras procesar los argumentos (y comprobar que no se intente dividir por 0) se llama a nuestro procedimiento remoto *calculadoraprog\_1*.

```

int
main (int argc, char *argv[])
{
    if (argc != 5) {
        printf ("usage: %s server_host float1 +|-|x|/ float2\n", argv[0]);
        exit (1);
    }

    /* Procesamiento de argumentos */
    char *host = argv[1];
    int f1 = atoi(argv[2]);
    char op = *argv[3];
    int f2 = atoi(argv[4]);

    if(op == '/' && f2==0) {
        printf ("stop: math error, cannot divide by 0\n");
        exit (1);
    }

    printf("calling server: %s with the following operation %i %c %i\n", host, f1, op, f2);

    calculadoraprog_1 (host, f1, op, f2);

    exit (0);
}

```

Código del main de *calculadora\_client.c*

El procedimiento *calculadoraprog\_1*:

- (1) Crea el proceso cliente para establecer la conexión con el servidor.
- (2) Analiza el operador y realiza la llamada correspondiente al operador.
- (3) Analiza el resultado: hay dos variables (*int \*result\_1*, *float \*result\_2*) puesto que la división devuelve un número real y el resto de operaciones devuelven números enteros. Se presentan tres casos entonces dependiendo del valor de estas variables tras haber realizado las llamadas correspondientes al servidor:
  - (a) Si ambas son nulas, es porque no se ha realizado ninguna operación (error: call failed).
  - (b) Si sólo *int \*result\_1* es nula, es porque se ha realizado una división. Se imprime por pantalla el valor de *result\_2*.
  - (c) Si sólo *float \*result\_2* es nula, es porque se ha realizado una suma, resta o multiplicación. Se imprime por pantalla el valor de *result\_1*.
- (4) Cierra la conexión y libera al cliente.

```

void
calculadoraprog_1(char *host, int num1, char op, int num2)
{
    // Cliente
    CLIENT *clnt;

    #ifndef DEBUG
        clnt = clnt_create (host, CALCULADORAPROG, CALCULADORAVER, "udp");
        if (clnt == NULL) {
            clnt_pcreateerror (host);
            exit (1);
        }
    #endif /* DEBUG */

    // Argumentos de entrada
    inputs arg;
    arg.num1 = num1;
    arg.num2 = num2;

    // Resultado(s)
    int *result_1 = (int*)NULL; // resultado para la suma, resta y multiplicacion
    float *result_2 = (float*)NULL; // resultado para la división

    switch(op) {
        case('+'): result_1 = suma_1(arg, clnt); break;
        case('-'): result_1 = resta_1(arg, clnt); break;
        case('x'): result_1 = multiplicacion_1(arg, clnt); break;
        case('/'): result_2 = division_1(arg, clnt); break;
        default:
            clnt_perror (clnt, "invalid operator, [valid operators: +|-|x|/ ]\n");
            exit(-1);
    }

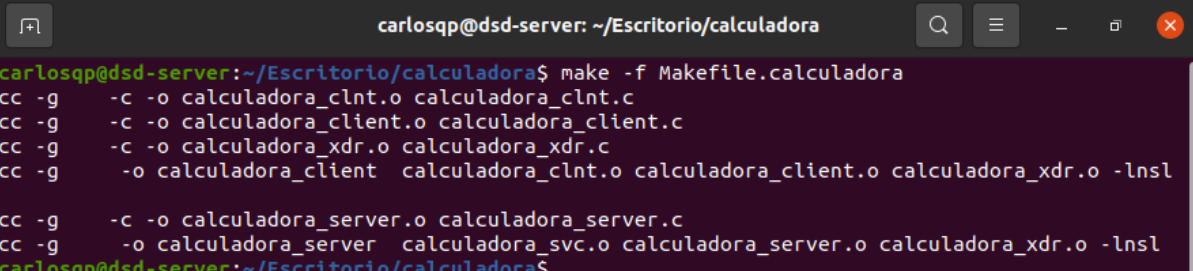
    if((result_1==(int *)NULL) && (result_2==(float*)NULL)) {
        clnt_perror (clnt, "call failed");
        exit(-1);
    } else if(result_1==(int *)NULL)
        printf("%i %c %i = %.2f\n", num1, op, num2, *result_2); // Resultado de division
    else
        printf("%i %c %i = %i\n", num1, op, num2, *result_1); // Resultado de suma, mul

    #ifndef DEBUG
        clnt_destroy (clnt);
    #endif /* DEBUG */
}

```

Procedimiento `calculadoraprog_1` del archivo `calculadora_client.c`

Para compilar el programa utilizamos el makefile generado por la herramienta `rpcgen` (`make -f Makefile.calculadora`).



```

carlosqp@dssd-server: ~/Escritorio/calculadora
carlosqp@dssd-server:~/Escritorio/calculadora$ make -f Makefile.calculadora
cc -g -c -o calculadora_clnt.o calculadora_clnt.c
cc -g -c -o calculadora_client.o calculadora_client.c
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_server.o calculadora_server.c
cc -g -o calculadora_server calculadora_svc.o calculadora_server.o calculadora_xdr.o -lnsl
carlosqp@dssd-server:~/Escritorio/calculadora$

```

Por último, en las pruebas de ejecución del programa he incluido un script (*run.sh*) que compila y ejecuta en local el programa de forma automática. Las salidas del proceso cliente las almacena en *cliente.txt* y las del servidor las muestra por pantalla. Si se desea cambiar las operaciones, el servidor o los operandos, el orden de parámetros es el siguiente:

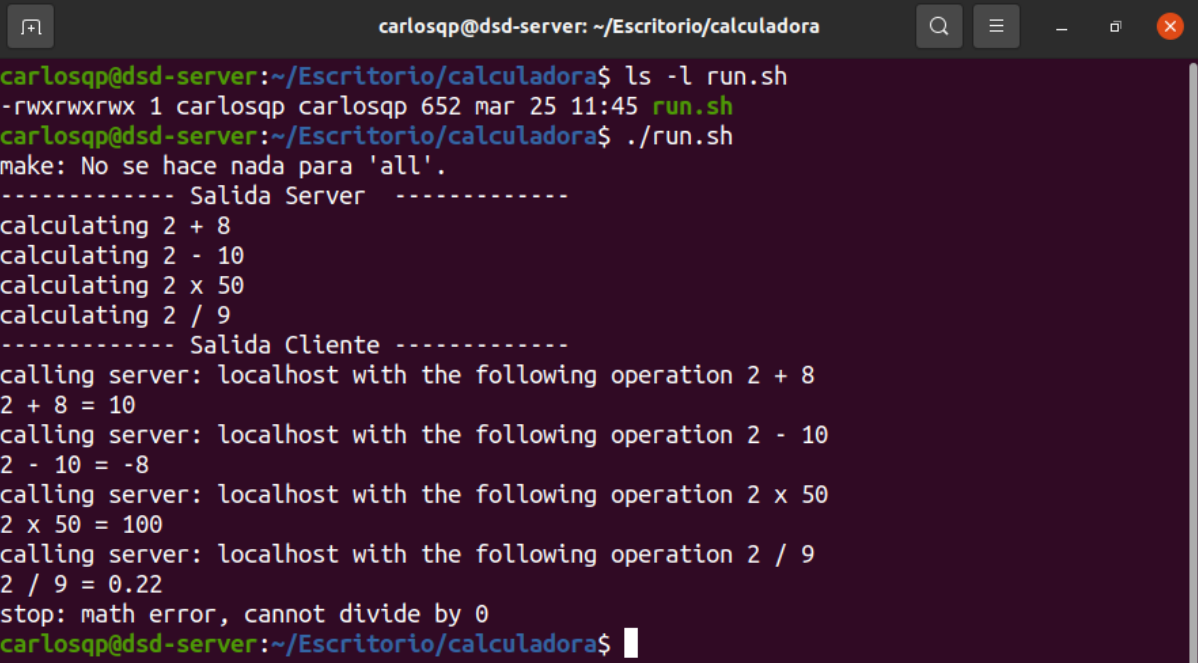
*./calculadora\_client <server> <entero> <operador> <entero>*



```

1 #!/bin/bash
2
3 # Compilar
4 make -f Makefile.calculadora
5
6 # Lanza el servidor en bg y muestra su salida por terminal
7 echo '----- Salida Server -----'
8 ./calculadora_server &
9
10 # Lanza varias veces el cliente con distintas operaciones
11 # y guarda la salida en el archivo cliente.txt
12 ./calculadora_client localhost 2 + 8 > cliente.txt
13 ./calculadora_client localhost 2 - 10 >> cliente.txt
14 ./calculadora_client localhost 2 x 50 >> cliente.txt
15 ./calculadora_client localhost 2 / 9 >> cliente.txt
16 ./calculadora_client localhost 2 / 0 >> cliente.txt
17
18 #Imprime la salida del cliente
19 echo '----- Salida Cliente -----'
20 cat cliente.txt
  
```

Script *run.sh*



```

carlosqp@dssd-server: ~/Escritorio/calculadora
carlosqp@dssd-server:~/Escritorio/calculadora$ ls -l run.sh
-rwxrwxrwx 1 carlosqp carlosqp 652 mar 25 11:45 run.sh
carlosqp@dssd-server:~/Escritorio/calculadora$ ./run.sh
make: No se hace nada para 'all'.
----- Salida Server -----
calculating 2 + 8
calculating 2 - 10
calculating 2 x 50
calculating 2 / 9
----- Salida Cliente -----
calling server: localhost with the following operation 2 + 8
2 + 8 = 10
calling server: localhost with the following operation 2 - 10
2 - 10 = -8
calling server: localhost with the following operation 2 x 50
2 x 50 = 100
calling server: localhost with the following operation 2 / 9
2 / 9 = 0.22
stop: math error, cannot divide by 0
carlosqp@dssd-server:~/Escritorio/calculadora$
  
```

Resultado de ejecución del script