

## Memoria Práctica 2

---

Mario López González

2 de abril de 2022



# Índice

<b>1</b>	<b>Practica 2: Sun RPC</b>	<b>3</b>
1.1	calculadora.x . . . . .	3
1.2	Generación de los archivos . . . . .	3
1.3	Cliente y servidor . . . . .	4
1.4	Ejemplos de funcionamiento . . . . .	6
<b>2</b>	<b>Práctica 2: Thrift</b>	<b>7</b>
2.1	calculadora.thrift . . . . .	7
2.2	Generación de archivos . . . . .	7
2.3	Cliente y servidor . . . . .	7
2.4	Servidor en Java . . . . .	9
2.4.1	Ejemplo de funcionamiento Cliente.py y Servidor.java . . . . .	10

## 1. Practica 2: Sun RPC

### 1.1. calculadora.x

Defino una interfaz denominada calculadora.x que contendrá los datos y rutinas accesibles remotamente.

```
union operacion_result switch(int errno){
    case 0:
        double result;
    case 1:
        void;
};

program CALCULADORA {
    version CALCULADORA_1 {
        operacion_result sumar (double, double) = 1;
        operacion_result restar (double, double) = 2;
        operacion_result multiplicar (double, double) = 3;
        operacion_result dividir (double, double) = 4;
        operacion_result potencia (double, double) = 5;
        operacion_result raiz (double, double) = 6;
        operacion_result factorial (double) = 7;
        operacion_result modulo (double, double) = 8;
        operacion_result vAbsoluto (double) = 9;
    } = 1;
} = 0x20000155;
```

Defino una unión para obtener el resultado en caso de éxito o error en caso de que se genere algún error durante la comunicación.

Las operaciones que realizará mi calculadora serán las siguientes:

- sumas
- restas
- multiplicaciones
- divisiones
- potencias
- raíces
- factoriales
- módulos
- valores absolutos

### 1.2. Generación de los archivos

Para generar los archivos para Sun RPC hay que ejecutar el siguiente comando:

```
rpcgen -NCa calculadora.x
```

Este comando genera los archivos necesarios para la comunicación entre el cliente y el servidor.

### 1.3. Cliente y servidor

El programa cliente se encarga de filtrar las operaciones que posteriormente se enviarán al servidor. Para ello he realizado una calculadora interactiva que lee los valores paso a paso. Muestra un menú con las operaciones disponibles, se introduce por pantalla el carácter correspondiente a la operación y según sea la operación lee un número o dos:

```
ubuntu@primary: ~/Home/Desktop/UGR/Tercero/Segundo Cuatrimestre/DSD/practicasDSD/P1/calculadora_c
¿Que operacion quiere realizar?
+ : Para realizar una suma
- : Para realizar una resta
* : Para realizar una multiplicacion
/ : Para realizar una division
^ : Para realizar una potencia
v : Para realizar una raiz
! : Para realizar un factorial
% : Para realizar un modulo
a : Para realizar un valor absoluto
s : Para salir
+
Introduzca el primer numero
2
Introduzca el segundo numero
4
```

```
ubuntu@primary: ~/Home/Desktop/UGR/Tercero/Segundo Cuatrimestre/DSD/practicasDSD/P1/calculadora_c
¿Que operacion quiere realizar?
+ : Para realizar una suma
- : Para realizar una resta
* : Para realizar una multiplicacion
/ : Para realizar una division
^ : Para realizar una potencia
v : Para realizar una raiz
! : Para realizar un factorial
% : Para realizar un modulo
a : Para realizar un valor absoluto
s : Para salir
!
Introduzca el numero
5
```

Hay ciertos mecanismos de seguridad frente a lecturas erróneas como por ejemplo introducir una operación no especificada en el menú o realizar raíces con índice cero entre otros.

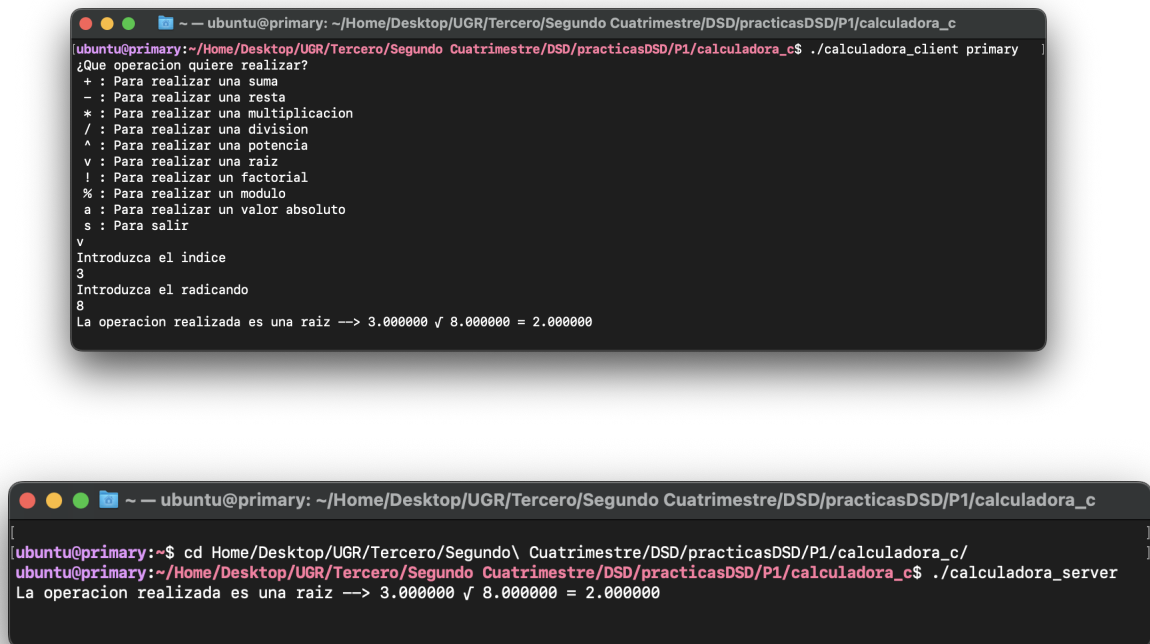
Una vez procesada la operación a realizar se invoca al método denominado `calculadora_1` pasando como argumentos el nombre del host, los dígitos leídos y el carácter introducido.

La función `calculadora_1` se encarga de filtrar las operaciones para llamar al método remoto correspondiente del servidor. El filtrado se realiza con un switch respecto del carácter pasado como argumento.

La función remota devuelve el resultado en una unión. Si el valor de la unión es void, es decir, nulo

significa que ha ocurrido un error en la comunicación y mostrará un mensaje por pantalla. En caso de éxito el cliente mostrará un mensaje indicando la operación realizada junto con su resultado.

Para comprobar que la comunicación se realiza con los mismos datos, el servidor muestra una salida de la operación usando los datos entrantes y los datos devueltos



The first terminal window shows the execution of the calculator client. The user runs `./calculadora_client primary`. The program prompts for an operation, and the user enters 'v' for square root. It then prompts for an index (3) and a radicand (8). The output is: "La operacion realizada es una raiz --> 3.000000 √ 8.000000 = 2.000000".

The second terminal window shows the execution of the calculator server. The user runs `./calculadora_server`. The output is: "La operacion realizada es una raiz --> 3.000000 √ 8.000000 = 2.000000", matching the client's output.

A continuación mostraré como se realizan las invocaciones de los métodos para la comunicación entre el cliente y el servidor y viceversa.



```
case '+':
    result = sumar_1(num1, num2, clnt);
    if (result == (operacion_result *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    printf("La operacion realizada es una suma --> %lf + %lf = %lf\n\n", num1, num2, result->operacion_result_u.result);
    break;
```

Figura 1.1: Llamada desde el cliente

```

operacion_result *
sumar_1_svc(double arg1, double arg2, struct svc_req *rqstp)
{
    static operacion_result result;

    result.operacion_result_u.result = (arg1 + arg2);

    printf("La operacion realizada es una suma --> %lf + %lf = %lf\n\n", arg1, arg2, result.operacion_result_u.result);

    return &result;
}

```

Figura 1.2: Llamada desde el servidor

## 1.4. Ejemplos de funcionamiento

```

ubuntu@primary: ~/Home/Desktop/UGR/Tercero/Segundo Cuatrimestre/DSD/practicasDSD
¿Que operacion quiere realizar?
+ : Para realizar una suma
- : Para realizar una resta
* : Para realizar una multiplicacion
/ : Para realizar una division
^ : Para realizar una potencia
v : Para realizar una raiz
! : Para realizar un factorial
% : Para realizar un modulo
a : Para realizar un valor absoluto
s : Para salir
/
Introduzca el numero
5
La operacion realizada es un factorial --> 5.000000! = 120.000000

-----

¿Que operacion quiere realizar?
+ : Para realizar una suma
- : Para realizar una resta
* : Para realizar una multiplicacion
/ : Para realizar una division
^ : Para realizar una potencia
v : Para realizar una raiz
! : Para realizar un factorial
% : Para realizar un modulo
a : Para realizar un valor absoluto
s : Para salir
/
Introduzca el primer numero
3
Introduzca el segundo numero
2
La operacion realizada es una division --> 3.000000 / 2.000000 = 1.500000

-----

¿Que operacion quiere realizar?
+ : Para realizar una suma
- : Para realizar una resta
* : Para realizar una multiplicacion
/ : Para realizar una division
^ : Para realizar una potencia
v : Para realizar una raiz
! : Para realizar un factorial
% : Para realizar un modulo
a : Para realizar un valor absoluto
s : Para salir
s
ubuntu@primary:~/Home/Desktop/UGR/Tercero/Segundo Cuatrimestre/DSD/practicasDSD/P1/calculadora_c$

ubuntu@primary:~/Home/Desktop/UGR/Tercero/Segundo Cuatrimestre/DSD/practicasDSD/P1/calculadora_c$ ./calculadora_server
La operacion realizada es una multiplicacion --> 4.000000 * 5.000000 = 20.000000

La operacion realizada es un modulo --> 6.000000 % 4.000000 = 2.000000

La operacion realizada es un factorial --> 5.000000! = 120.000000

La operacion realizada es una division --> 3.000000 / 2.000000 = 1.500000

[]

```

## 2. Práctica 2: Thrift

### 2.1. calculadora.thrift

Las operaciones implementadas en esta parte de la práctica son las mismas que las realizadas en la parte de Sun RPC. Para ello he creado un archivo denominado calculadora.thrift.

```
service Calculadora{  
    double suma(1:double num1, 2:double num2),  
    double resta(1:double num1, 2:double num2),  
    double multiplicacion(1:double num1, 2:double num2),  
    double division(1:double num1, 2:double num2),  
    double potencia(1:double num1, 2:double num2),  
    double raiz(1:double num1, 2:double num2),  
    double factorial(1:double num1),  
    double modulo(1:double num1, 2:double num2),  
    double vabsoluto(1:double num1),  
}
```

### 2.2. Generación de archivos

Para generar los archivos necesarios para thrift hay que ejecutar el siguiente comando:

```
thrift -gen py calculadora.thrift
```

### 2.3. Cliente y servidor

El programa cliente realiza la misma función que el realizado en Sun RPC, filtrar la entrada de operaciones e invocar al método correspondiente del servidor y asegurarse de que no se introducen valores erróneos de operaciones o dígitos. La única diferencia en cuanto a implementación es que se filtra directamente en el main y no haciendo uno de una función externa.

```

~/Desktop/UGR/Tercero/Segundo Cuatrimestre/DSD/practicasDSD/P1/calculadora_py_java/...
mariolop gen-py (master) $ python cliente.py
*****
CALCULADORA EN PYTHON
*****
¿Que operacion quiere realizar?
+ : Para realizar una suma
- : Para realizar una resta
* : Para realizar una multiplicacion
/ : Para realizar una division
^ : Para realizar una potencia
v : Para realizar una raiz
! : Para realizar un factorial
% : Para realizar un modulo
a : Para realizar un valor absoluto
s : Para salir
/
Introduzca el primer numero
3
Introduzca el segundo numero
0
Error. Introduzca un divisor distinto de 0
-2
La operacion realizada es una division --> 3.0 / -2.0 = -1.5
*****
¿Que operacion quiere realizar?
+ : Para realizar una suma
- : Para realizar una resta
* : Para realizar una multiplicacion
/ : Para realizar una division
^ : Para realizar una potencia
v : Para realizar una raiz
! : Para realizar un factorial
% : Para realizar un modulo
a : Para realizar un valor absoluto
s : Para salir
^
Introduzca la base
2
Introduzca el exponente
3
La operacion realizada es una potencia --> 2.0 ^ 3.0 = 8.0
*****
¿Que operacion quiere realizar?
+ : Para realizar una suma
- : Para realizar una resta
* : Para realizar una multiplicacion
/ : Para realizar una division
^ : Para realizar una potencia
v : Para realizar una raiz
! : Para realizar un factorial
% : Para realizar un modulo

```

```

~/Desktop/UGR/Tercero/Segundo Cuatrimestre/DSD/practicasDSD/P1/calculadora_py_java/...
mariolop gen-py (master) $ python servidor.py
iniciando servidor...
La operacion realizada es una division --> 3.0 / -2.0 = -1.5
La operacion realizada es una potencia --> 2.0 ^ 3.0 = 8.0

```

A continuación mostraré como se realizan las invocaciones de los métodos para la comunicación entre el cliente y el servidor y viceversa.

```

if(operador == "^"):
    print("Introduzca la base")
    n1 = float(input())

    print("Introduzca el exponente"):
    n2 = float(input())

    result = client.potencia(n1, n2)
    print("La operacion realizada es una potencia --> " + str(n1) + " ^ " + str(n2) + " = " + str(result))

```

Figura 2.1: Llamada desde el cliente

```

def potencia(self, n1, n2):
    result = n1 ** n2
    print("La operacion realizada es una potencia --> " + str(n1) + " ^ " + str(n2) + " = " + str(result))

```

Figura 2.2: Llamada desde el servidor



## 2.4. Servidor en Java

Como parte extra he implementado el servidor en Java. Gracias a thrift, el cliente implementado en Python se puede comunicar con el Servidor implementado en Java. Para generar los archivos necesarios para el funcionamiento del servidor hay que ejecutar el siguiente comando:

```
thrift -r --gen java tutorial.thrift
```

Este comando genera la clase calculadora que está compuesta por la interfaz Iface que será la que implemente el servidor.

Además, es necesario descargar las siguientes librerías:

- Librería libthrift-0.16.0.jar
- Librería slf4j-api-1.7.9.jar
- Librería slf4j-simple-1.7.9.jar

La inclusión de las librerías las he realizado con NetBeans. El servidor implementa dos clases: CalculadoraHandler que se encarga de implementar la interfaz Calculadora.Iface; y, la clase Servidor, una hebra que realiza las llamadas correspondientes de la clase CalculadoraHandler.

```
public class Servidor {  
  
    public static CalculadoraHandler handler;  
  
    public static Calculadora.Processor processor;  
  
    public static void main(String[] args) {  
        try{  
            handler = new CalculadoraHandler();  
            processor = new Calculadora.Processor(handler);  
  
            TServerTransport serverTransport = new TServerSocket(9090);  
            TServer server = new TSimpleServer(new Args(serverTransport).processor(processor));  
            System.out.println("Iniciando servidor...");  
            server.serve();  
        } catch (Exception e) { e.printStackTrace(); }  
    }  
}
```

Figura 2.3: Clase Servidor en Java

```

@Override
public double factorial(double num1) throws TException {
    double result = 1;
    for(int i = 2; i <= num1; i++)
        result *= i;
    System.out.println("La operacion realizada es un factorial --> " + num1 + "! = " + result);
    return result;
}

```

Figura 2.4: Implementación de llamadas en CalculadoraHandler

### 2.4.1. Ejemplo de funcionamiento Cliente.py y Servidor.java

```

~/Desktop/UGR/Tercero/Segundo Cuatrimestre/DSD/practicadSD/P1/calculadora_py_java/...
Introduzca el segundo numero
7
La operacion realizada es una suma --> 2.0 + 7.0 = 9.0

*****

¿Que operacion quiere realizar?
+ : Para realizar una suma
- : Para realizar una resta
* : Para realizar una multiplicacion
/ : Para realizar una division
^ : Para realizar una potencia
v : Para realizar una raiz
! : Para realizar un factorial
% : Para realizar un modulo
a : Para realizar un valor absoluto
s : Para salir
a
Introduzca el numero
-5
La operacion realizada es un valor absoluto --> abs(-5.0) = 5.0

*****

¿Que operacion quiere realizar?
+ : Para realizar una suma
- : Para realizar una resta
* : Para realizar una multiplicacion
/ : Para realizar una division
^ : Para realizar una potencia
v : Para realizar una raiz
! : Para realizar un factorial
% : Para realizar un modulo
a : Para realizar un valor absoluto
s : Para salir
^
Introduzca la base
4
Introduzca el exponente
0.5
La operacion realizada es una potencia --> 4.0 ^ 0.5 = 2.0

*****

¿Que operacion quiere realizar?
+ : Para realizar una suma
- : Para realizar una resta
* : Para realizar una multiplicacion
/ : Para realizar una division
^ : Para realizar una potencia
v : Para realizar una raiz
! : Para realizar un factorial
% : Para realizar un modulo
a : Para realizar un valor absoluto
s : Para salir

```

```

Output - servidor (run) x
run:
Iniciando servidor...
La operacion realizada es una suma --> 2.0 + 7.0 = 9.0
La operacion realizada es un valor absoluto --> abs(-5.0) = 5.0
La operacion realizada es una potencia --> 4.0 ^ 0.5 = 2.0

```