Desarrollo de Sistemas Distribuidos

- Práctica 2 -

Ignacio Yuste López. DSD1.

2.1. Sun RPC

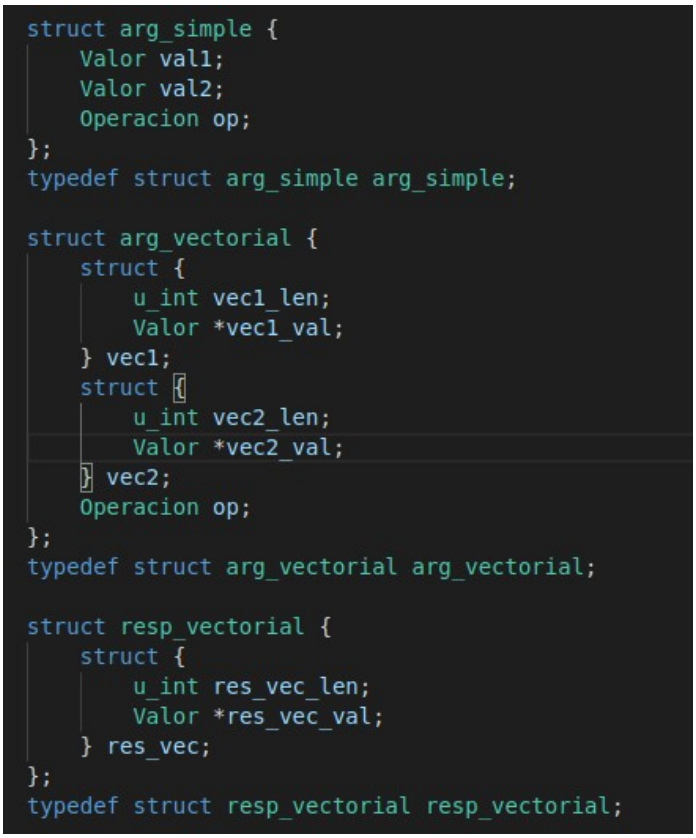
Para la práctica he implementado un cliente y el servidor de una calculadora. Esta calculadora puede hacer las operaciones básicas y algunas operaciones vectoriales.

La calculadora tiene dos opciones, una para cálculos simples y otra para cálculos con vectores. He declarado una serie de estructuras en RPC Sun para poder pasar más de un argumento a los métodos remotos. Estas estructuras son:

```
typedef float Valor;
typedef char Operacion;

struct arg_simple {
    Valor val1;
    Valor val2;
    Operacion op;
};

struct arg_vectorial {
    Valor vec1<>;
    Valor vec2<>;
    Operacion op;
};

struct resp_vectorial {
    Valor res_vec<>;
};
```

1. arg_simple: Estructura que comprende dos valores en coma flotante y la operación a realizar con ellos. Es usada como argumento de la calculadora simple.
2. arg_vectorial: Estructura con dos listas de números en coma flotante y la operación a realizar. Permite el uso de listas de tamaño variable. Es usada como argumento de la calculadora vectorial.
3. resp_vectorial: Estructura con una lista de números en coma flotante y la operación a realizar. Permite el uso de listas de tamaño variable. Es usada como retorno de la calculadora vectorial.

Gracias a rpcgen, estas estructuras se transforman en tipos de datos que podemos utilizar para la implementación tanto del cliente como del servidor. Si observamos el archivo *.h* generado vemos cómo se implementan en el lenguaje C:



```
struct arg_simple {
    Valor val1;
    Valor val2;
    Operacion op;
};
typedef struct arg_simple arg_simple;

struct arg_vectorial {
    struct {
        u_int vec1_len;
        Valor *vec1_val;
    } vec1;
    struct {
        u_int vec2_len;
        Valor *vec2_val;
    } vec2;
    Operacion op;
};
typedef struct arg_vectorial arg_vectorial;

struct resp_vectorial {
    struct {
        u_int res_vec_len;
        Valor *res_vec_val;
    } res_vec;
};
typedef struct resp_vectorial resp_vectorial;
```

Vemos como cada estructura almacena un puntero y la longitud de la lista. De esta forma maneja RPC las listas con tamaño dinámico.

En cuanto a los métodos implementados en ambas calculadoras tenemos:

1. Simple.
    1. Suma. Suma dos valores en coma flotante.
    2. Resta. Resta dos valores en coma flotante.
    3. División. Divide dos valores en coma flotante.
    4. Multiplicación. Multiplica dos valores en coma flotante.
    5. Módulo. Calcula el entre dos dos valores enteros.
    6. División entera. Calcula el divisor de dos números enteros.
2. Vectorial.
    1. Suma. Suma los elementos de dos vectores uno a uno.
    2. Resta. Resta los elementos de dos vectores uno a uno.
    3. División. Divide los elementos de dos vectores uno a uno.
    4. Multiplicación. Multiplica los elementos de dos vectores uno a uno.

Para la implementación de la calculadora vectorial he usado memoria dinámica, dando la opción al usuario de decidir el tamaño de los vectores con lo que calcular. Para esto, rpc provee de la función *xdr_free*, la cual libera la memoria de manera que no se producen goteras de memoria.

En cuanto a la interfaz he optado por una implementación sencilla con texto, la cual deja decidir el tipo de calculadora, la operación y los datos con los que calcular. Todo ello implementado en el cliente, más en concreto en las funciones *interfazSimple* e *interfazVectorial*.

Las llamadas al servidor remoto se hacen desde las funciones *calculadora_simple_call* y *calculadora_vectorial_call.*

2.2. Apache Thrift.

He implementado el cliente y el servidor de una calculadora en Python. En esta ocasión no ha sido necesario implementar nuevos tipos de estructuras en el arhivo .thrift ya que se pueden pasar más de un argumento en los métodos de llamada remota. Sin embargo he decidido definir el tipo *Matriz* para mayor comodidad a la hora de trabajar con listas de listas.

Esta vez la calculadora además de poder trabajar con datos simples y vectoriales también implementa dos opciones para matrices. Los métodos implementados en el servidor son los siguientes:

1. Simple.
    1. Suma. Suma dos valores en coma flotante.
    2. Resta. Resta dos valores en coma flotante.
    3. División. Divide dos valores en coma flotante.
    4. Multiplicación. Multiplica dos valores en coma flotante.
    5. Módulo. Calcula el entre dos dos valores enteros.
    6. División entera. Calcula el divisor de dos números enteros.
2. Vectorial.
    1. Máximo. Devuelve el elemento mayor del vector.
    2. Mínimo. Devuelve el elemento menor del vector.
    3. VectorPorEscalar. Multiplica los valores de un vector por un número escalar.
    4. VectorEntreEscalar. Divide los valores de un vector por un número escalar.
    5. VectorMasEscalar. Suma a los valores de un vector un número escalar.

6. VectorMenosEscalar. Resta a los valores de un vector un número escalar.

3. Matricial.
    1. SumaColumnas. Devuelve un vector con los valores de la suma de todas las columnas de la matriz.
    2. SumaFilas. Devuelve un vector con los valores de la suma de todas las filas de la matriz.

Al igual que en el caso anterior, he optado por una interfaz de texto que permite elegir la calculadora, las operaciones e introducir los datos que el usuario desee.