UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

PRÁCTICA 2:
LLAMADA REMOTA A PROCEDIMIENTO (RPC)

Realizado por: Mario Martínez Sánchez

Curso académico: 2022/2023

INTRODUCCIÓN:

En esta práctica se ha implementado un programa distribuido, haciendo uso de rpcgen, el cual realiza diversas operaciones como:

- Operaciones básicas (suma, resta, división, multiplicación).
- Operaciones trigonométricas (seno, coseno, arcotangente…).
- Operaciones con vectores (suma, resta, producto escalar y producto vectorial).
- Operaciones con matrices (suma, resta, multiplicación y división).
- Otras operaciones (potencias, raíces cuadradas).

Para ello, he completado 'calculadora_server.c', que realiza la función del servidor, y 'calculadora_client.c', que actúa como el cliente.

SERVIDOR:

Este es el encargado de realizar la operación y devolver el resultado de cada una de las funciones implementadas.
Por ejemplo, SUMA(a, b) se encarga de sumar los valores a y b, y devolver el resultado.

Respecto a los métodos que realizan operaciones trigonométricas, se ha requerido la biblioteca #include<math.h>, para poder así hacer uso de las funciones sin, cos, acos, etc.

Finalmente, para los métodos de operaciones de vectores, se declaró en 'calculadora.x' un struct denominado 'float_array', el cual contiene un vector de float y un entero que indica el tamaño de dicho vector.

CLIENTE:

Este es el encargado de recibir por parámetro las operaciones que la calculadora deberá resolver y, dependiendo de la operación, actuará de una forma u otra.

El formato de paso de parámetros será el siguiente:

<programa> <máquina> <tipo_operación> <parámetros>

De este modo, el parámetro <tipo_operación> indicará si la operación será básica, trigonométrica, vectores, etc. Además, dependiendo de esto, los parámetros posteriores serán unos u otros.

Las opciones de <tipo_operación> serán las siguientes:

- ‘0’ → operaciones básicas, cuyos parámetros serán los siguientes:

  <valor> <operador> <valor>

  - <operador> puede tomar los siguientes valores:
    - ‘+’: para realizar una suma.
    - ‘-’: para realizar una resta.
    - ‘x’: para realizar una multiplicación.
    - ‘/’: para realizar una división.
- ‘1’ → operaciones trigonométricas. cuyos parámetros serán los siguientes:

  <operador> <valor>

  - <operador> puede tomar los siguientes valores:
    - ‘sin’: para realizar el seno.
    - ‘cos’: para realizar el coseno.
    - ‘tan’: para realizar la tangente.
    - ‘asin’: para realizar el arcoseno.
    - ‘acos’: para realizar el arcocoseno.
    - ‘atan’: para realizar la arcotangente.
  - <valor> debe ser proporcionado en grados.
- ‘2’ → raíces y potencias, cuyos parámetros serán los siguientes:

  <operador> <valor> <valor_posible>

  - <operador> puede tomar los siguientes valores:
    - ‘sqrt’: para realizar la raíz cuadrada.
    - ‘pow’: para realizar una potencia.
  - <valor_posible> sólo podrá ser enviado en caso de haber indicado que la operación es una potencia (‘pow’), en cuyo caso corresponderá al exponente.
- ‘3’ → operaciones con vectores, cuyos parámetros serán los siguientes:

  <valor1> … <valorN> <operador> <valor1_2> … <valorN_2>

  - <operador> puede tomar los siguientes valores:
    - ‘+’: para realizar una suma.
    - ‘-’: para realizar una resta.
    - ‘x’: para realizar un producto vectorial, en cuyo caso ambos vectores deberán tener 3 componentes.
    - ‘o’: para realizar un producto escalar.
  - N corresponde al número de componentes de ambos vectores (debe ser el mismo).
- ‘4’ → operaciones con matrices, cuyos parámetros serán los siguientes:

  <N> <M> <valor1> … <valorNM> <operador> <N> <M> <valor1_2> … <valorNM_2>

  - <operador> puede tomar los siguientes valores:
    - ‘+’: para realizar una suma.
    - ‘-’: para realizar una resta.
    - ‘x’: para realizar una multiplicación.
    - ‘/’: para dividir entre un valor, en cuyo caso el primer parámetro debe ser una matriz y el segundo un número real.

Para la implementación del código, he realizado tres métodos para la calculadora.

El primero de estos es el encargado de realizar operaciones básicas, trigonométricas, raíces y potencias.

El segundo se encarga de los cálculos con vectores (suma, resta, producto), para el cual se ha utilizado un struct con un vector y un entero que indicará el tamaño de este.

En último lugar, la calculadora para matrices, en la cual se ha utilizado otro struct, pero con un vector y dos enteros, uno indicando el número de filas y otro para el número de columnas.

EJEMPLOS DE EJECUCIÓN DE OPERACIONES:

1. Operaciones básicas:

```
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 0 2.34 + 7.1
Operacion: 2.340000 + 7.100000 = 9.440000
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 0 5 x 7
Operacion: 5.000000 x 7.000000 = 35.000000
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 0 40 / 0
Error, no se puede dividir entre 0.
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 0 6 - 20
Operacion: 6.000000 - 20.000000 = -14.000000
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$
```

2. Operaciones trigonométricas:

```
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 1 sin 90
Operacion: sin(90.000000) = 1.000000
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 1 cos 45
Operacion: cos(45.000000) = 0.707107
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 1 tan 70
Operacion: tan(70.000000) = 2.747478
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 1 asin 0.5
Operacion: asin(0.500000) = 30.000000
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 1 acos 0.75
Operacion: acos(0.750000) = 41.409622
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 1 atan 1
Operacion: atan(1.000000) = 45.000000
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$
```

3. Raíces y potencias:

```
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 2 sqrt 121
Operacion: sqrt(121.000000) = 11.000000
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 2 sqrt 121 3
Error en el paso de parametros: las raices solo necesitan un valor.
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 2 pow 5 2
Operacion: pow(5.000000,2.000000) = 25.000000
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 2 pow 2 5
Operacion: pow(2.000000,5.000000) = 32.000000
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$
```

4. Operaciones con vectores:

```
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 3 2 3 4.5 + 4 1.1 3
Operacion: { 2.000000 3.000000 4.500000 } + { 4.000000 1.100000 3.000000 } = { 6.000000 4.100000 7.500000 }
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 3 2 3 4.5 - 4 1.1 3
Operacion: { 2.000000 3.000000 4.500000 } - { 4.000000 1.100000 3.000000 } = { -2.000000 1.900000 1.500000 }
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 3 2 4 6 x 1 2 3
Operacion: { 2.000000 4.000000 6.000000 } x { 1.000000 2.000000 3.000000 } = { 0.000000 0.000000 0.000000 }
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 3 1 2 3 x 2 3 4
Operacion: { 1.000000 2.000000 3.000000 } x { 2.000000 3.000000 4.000000 } = { -1.000000 2.000000 -1.000000 }
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 3 1 2 3 1 x 2 3 4 5
Para hacer productos vectoriales, los vectores deben ser de 3 elementos.
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 3 1 2 3 1 o 2 3 4 5
Operacion: { 1.000000 2.000000 3.000000 1.000000 } o { 2.000000 3.000000 4.000000 5.000000 } = 25.000000
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$
```

5. Operaciones con matrices:

```
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 4 3 2 1 2 3 4 5 6 + 3 2 9 8 7 6
5 4
Operacion:
Primera matriz:
| 1.000000   2.000000   |
| 4.000000   5.000000   |
| 0.000000   0.000000   |

Operador: +

Segunda matriz:
| 9.000000   8.000000   |
| 6.000000   5.000000   |
| 0.000000   0.000000   |

Resultado:
| 10.000000   10.000000   |
| 10.000000   10.000000   |
| 0.000000   0.000000   |

mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 4 3 2 1 2 3 4 5 6 + 2 2 7 6 5 4
Para realizar suma de matrices, deben tener el mismo tamaño.
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 4 3 2 1 2 3 4 5 6 - 3 2 9 8 7 6
5 4
Operacion:
Primera matriz:
| 1.000000   2.000000   |
| 4.000000   5.000000   |
| 0.000000   0.000000   |

Operador: -

Segunda matriz:
| 9.000000   8.000000   |
| 6.000000   5.000000   |
| 0.000000   0.000000   |

Resultado:
| -8.000000   -6.000000   |
| -2.000000   0.000000   |
| 0.000000   0.000000   |

mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$
```

```
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 4 3 2 1 2 3 4 5 6 x 3 2 9 8 7 6
5 4
Para realizar producto de matrices, el numero de columnas de la primera matriz debe ser igual al numero de filas de
la segunda.
mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 4 3 2 1 2 3 4 5 6 x 2 3 9 8 7 6
5 4
Operacion:
Primera matriz:
| 1.000000   2.000000   |
| 4.000000   5.000000   |
| 0.000000   0.000000   |

Operador: x

Segunda matriz:
| 9.000000   8.000000   7.000000   |
| 7.000000   6.000000   5.000000   |

Resultado:
| 38.000000   32.000000   |
| 101.000000   86.000000   |
| 42.000000   51.000000   |

mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$ ./calculadora_client localhost 4 3 3 1 2 3 1 2 3 2 3 4 / 2
Operacion:
Matriz:
| 1.000000   2.000000   3.000000   |
| 1.000000   2.000000   3.000000   |
| 2.000000   3.000000   4.000000   |

Operador: /

Valor: 2.000000

Resultado:
| 0.500000   1.000000   1.500000   |
| 0.500000   1.000000   1.500000   |
| 1.000000   1.500000   2.000000   |

mario@mariopc:~/DSD/practica-2-1-codigo/calculadora$
```