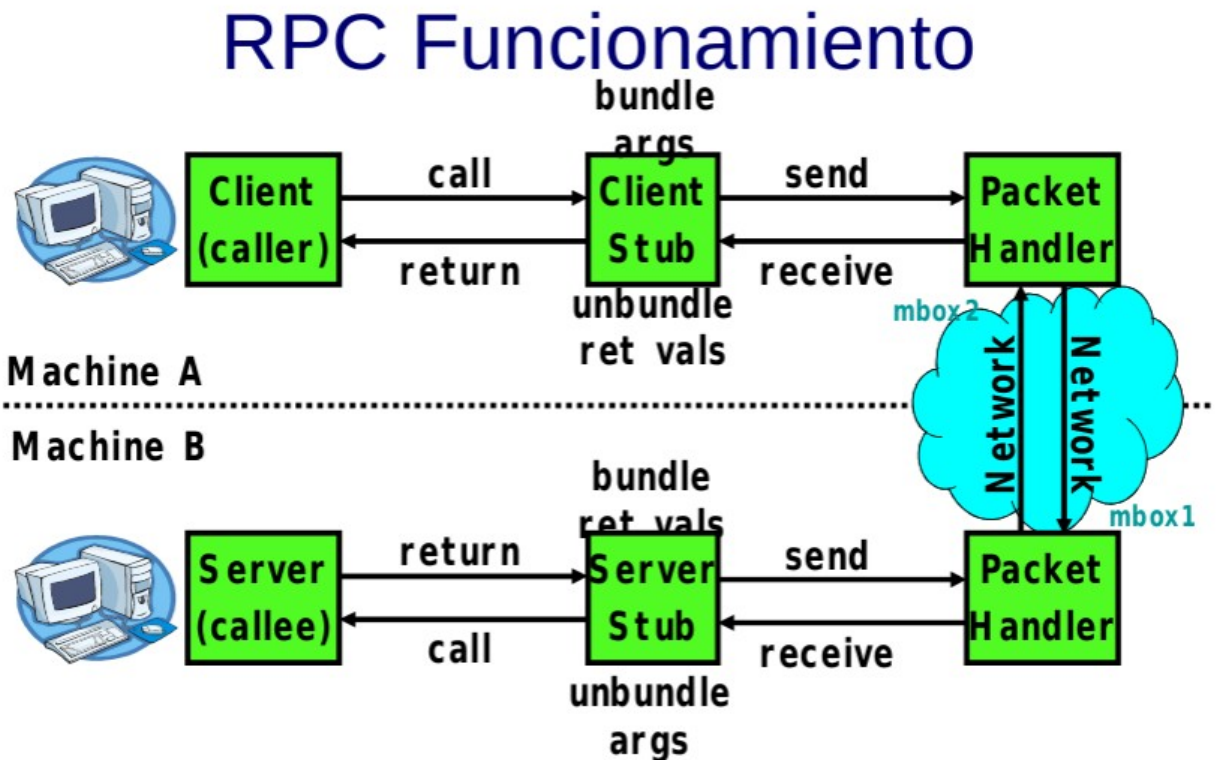
Práctica 2 RPC

Calculadora

- Alumno: José Carlos López Aguilar
- Curso: 2022/2023
- Asignatura: Diseño de Sistemas Distribuidos (DSD)



1 Introducción

Para esta práctica se ha planteado desarrollar una calculadora. El programa cliente recibe las operaciones matemáticas, establece conexión con el servidor y pide que le resuelva los cálculos.

Se va a utilizar el middleware RPC de Sun, una tecnología software que implementa la ***invocación remota de procedimientos***. Esta capa software se encarga de establecer la conexión con el servidor, codificar los datos en un formato estándar y portable (XDR) y del paso de mensajes.

2 Solución propuesta y entregada

Se entregan un conjunto de ficheros fuente para la construcción de los programas cliente y servidor en los que se cumple los siguientes roles:

1. Cliente: se encarga de pedir al usuario las operaciones matemáticas y mostrar el resultado. Para obtener el resultado establece conexión con el servidor e invoca un procedimiento remoto.
2. Servidor: encargado del cómputo en sí. Es la calculadora. Realiza las operaciones matemáticas que el cliente solicita. También gestiona e informa de errores como:
    1. División por 0
    2. Algunos errores sintácticos como son poner un operador inválido o un número decimal con dos puntos decimales (1.2243.2 o 1.2 .3).

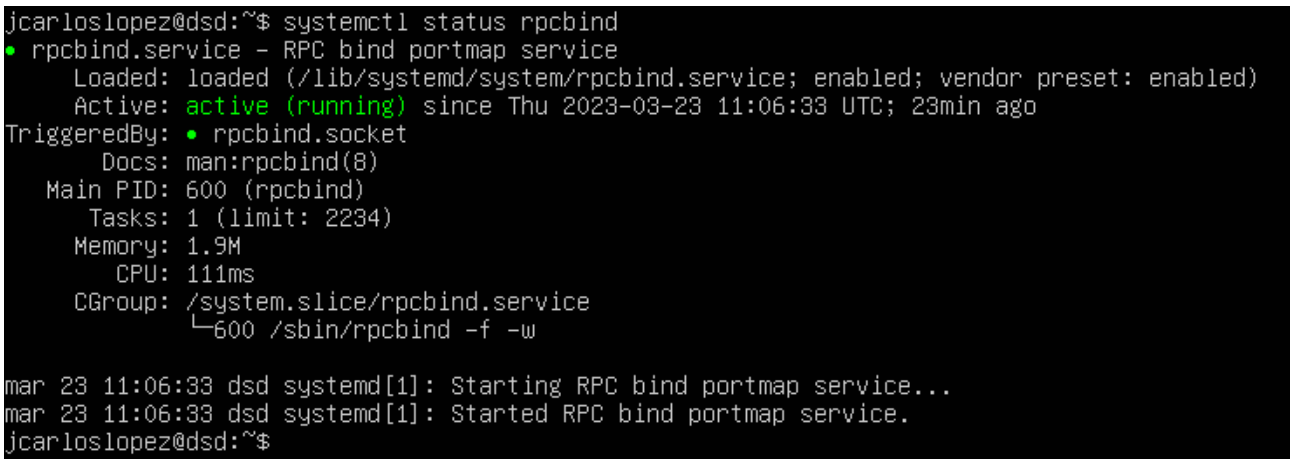
    El cliente será encargado de gestionar los errores que el servidor le envía.

Las operaciones disponibles para la calculadora son:

- Suma: símbolo +
- Resta: símbolo -
- Multiplicación:*
- División: /

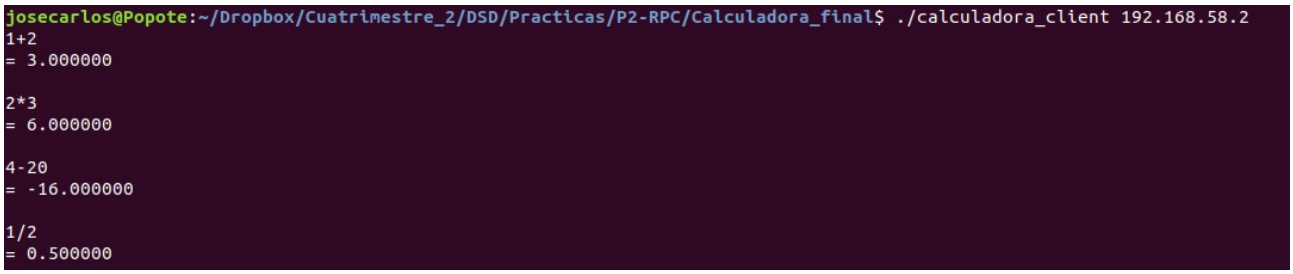
Ejemplo de uso

En el servidor se debe estar ejecutando un proceso demonio llamado **rpcbind** (escucha en el puerto 111). Después ejecutamos el programa servidor.

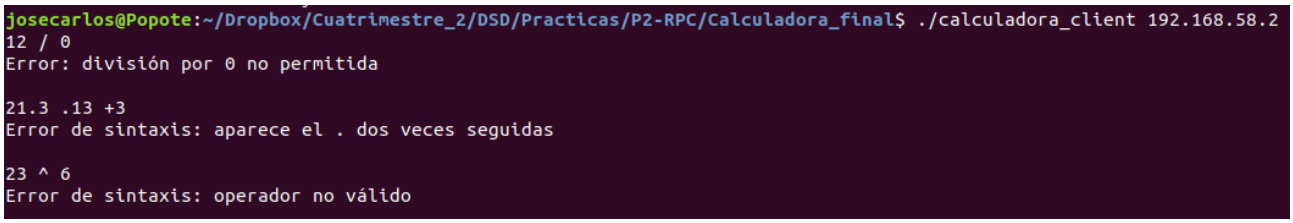


Estado del demonio rpcbind y ejecución del programa servidor. (En este caso estoy usando una máquina virtual como servidor)

En el cliente solo debemos ejecutar su correspondiente programa. Este recibe un parámetro por línea de comandos: la dirección del servidor, ya puede ser nombre de dominio o IP. Si no se usara una máquina virtual y fuera todo en el propio anfitrión pondríamos localhost por ejemplo.



Ejecución de las operaciones básicas de la calculadora. Una operación con dos operandos



Gestión de errores: división por 0 y algunos errores sintácticos

El cliente, en un bucle infinito está esperando expresiones matemáticas. Para terminar el programa hay que enviar la señal EOF , es decir, **Ctrl + D**.

Concatenación de operaciones

Además de realizar las operaciones básicas binarias, se pueden escribir operaciones complejas, es decir, concatenación de operaciones. Esto permite utilizar operadores diferentes respetando la precedencia de estos, además del uso de paréntesis.

Por tanto se permite la expresión (2+1)*3. Se pueden usar tanto números enteros como reales, por ejemplo 1.23 + 12 / 6 .

```
josecarlos@Popote:~/Dropbox/Cuatrimestre_2/DSD/Practicas/P2-RPC/Calculadora_final$ ./calculadora_client 192.168.58.2
(2+1)*3
= 9.000000

1.23 + 12 / 6
= 3.230000
```

Formato de los números reales

El programa acepta como números reales aquellos que cumplan con el formato de las funciones **scanf** o **atof**. Es decir, son válidos:

- 1.1
- .1 == 0.1
- 1. == 1.0 == 1

```
josecarlos@Popote:~/Dropbox/Cuatrimestre_2/DSD/Practicas/P2-RPC/Calculadora_final$ ./calculadora_client 192.168.58.2
1.1
= 1.100000

1.
= 1.000000

.1
= 0.100000
```

Cómo se ha realizado

¿Cómo se analiza y evalúa una expresión matemática? Las personas usamos una notación para escribir y expresar las operaciones. Esta se llama **notación infija**, ya que el operador se encuentra en medio de los operandos (1+2). Existen más notaciones, como la **polaca** o **prefija** (+1 2) y la **polaca inversa, sufija** o **postfija** (1 2 +).

Nuestro programa cliente recibe la expresión infija y se la envía al servidor. Este se encarga de pasar de esta notación a la postfija. Una vez con la notación postfija aplicamos un algoritmo (más sencillo) para evaluar una expresión matemática, donde ya no se tienen en cuenta paréntesis y la precedencia de operadores viene determinada por el orden en la notación postfija.

Para el algoritmo de pasar de notación infija a postfija y el algoritmo que evalúa una expresión postfija, me he basado en la página https://runestone.academy/ns/books/published/pythoned/BasicDS/ExpresionesInfijasPrefijasYSufijas.html . Viene una implementación en python para cada algoritmo. En este caso la hemos traducido a C y además para que funcione con números reales.

Más cerca de la implementación

Lo primero es definir el fichero interfaz de RPC (.x) que define los tipos de datos, el programa, su versión y los procedimientos que encapsula. Como se ve en el fichero calculadora.x, se ha definido un programa con un solo procedimiento. Este se encarga de evaluar cualquier expresión infija (como se ha descrito antes), por lo que no es necesario publicar procedimientos para operaciones binarias básicas.

Se realiza un control de errores y por ello se devuelve un dato complejo (union de RPC). Si la variable error == 0, no hay error y por tanto se devuelve un valor numérico en resultado. Si error = 1, se devuelve un mensaje descriptivo del error.

Una vez creado este fichero, con la herramienta **rpcgen** generamos las plantillas con código para crear nuestros programas, además de otros archivos necesarios. Para ello ejecutamos:

- rpcgen -NCa calculadora.x

Para implementar los algoritmos de paso de notación infija a postfija y la evaluación de una expresión infija se ha tenido que crear la estructura de datos, **pila**. Son necesarias tanto una pila de elementos char (u operadores como + - * / …) como una pila de números. Se proporcionan los siguientes ficheros:

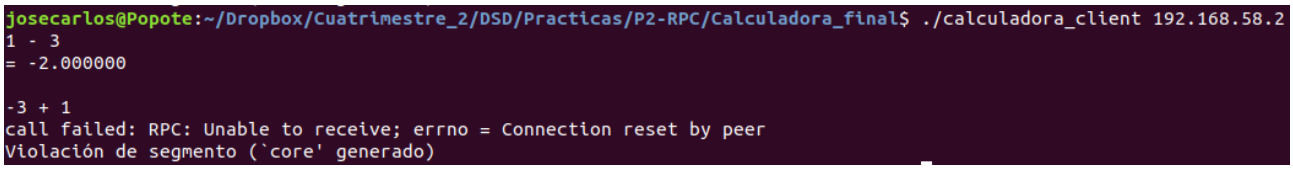
- Para la pila de caracteres u operandos
  - pila_char.h
  - pila_char.c
- Para la pila de números
  - pila_numeros.h
  - pila_números.c

En resumen se ha implementado una estructura que representa a una pila y un conjunto de operaciones como son top, pop, push, etc. Se hace uso de memoria dinámica. El programa servidor es quien utiliza estas estructuras así que son necesarios estos archivos a la hora de su compilación. El makefile que se entrega ya viene modificado para que la compilación sea correcta.

3 Algunos fallos…

Los siguientes errores producen un error/excepción que abortan la ejecución del programa cliente y servidor:

- No se ha tenido en cuenta ciertos errores sintácticos, como por ejemplo 5 ++ 3 (repetición de operadores).
- No se han gestionado los números negativos. Las restas funcionan bien pero el uso de números negativos no



4 Ejemplos para probar en la calculadora

- 1 + 2 *3/ 6
- (1 + 2) * (3/6)
- 1/2 + 1/2
- 1 / (2+1) / 2

5 Referencias

1. Algoritmo para pasar de una notación infija a postfija y evaluación de está última: https://runestone.academy/ns/books/published/pythoned/BasicDS/ExpresionesInfijasPrefijasYSufijas.html