Memoria de P2

RPC-GEN

Shunya Zhan

27 de marzo del 2025

Introducción

En esta práctica, hemos desarrollado una aplicación distribuida utilizando la herramienta RPCGEN, que facilita la creación de aplicaciones basadas en el modelo de llamada a procedimiento remoto (RPC). El objetivo principal de esta práctica es implementar una calculadora distribuida que permita realizar operaciones aritméticas básicas (suma, resta, multiplicación, división, potencia y raíz cuadrada) y llevar un registro del número de llamadas realizadas al servidor.

RPCGEN es una herramienta que genera automáticamente el código necesario para la comunicación remota en ONC-RPC. Simplifica la creación de aplicaciones RPC al evitar la necesidad de escribir manualmente los ficheros de cabecera y las estructuras de comunicación. A partir de un archivo de definición de interfaz con extensión ".x", RPCGEN genera el código en C tanto para el cliente como para el servidor.

En esta memoria, se detallarán los siguientes aspectos:

- La estructura y funcionamiento del cliente, incluyendo cómo se empaquetan y envían las solicitudes al servidor.
- La estructura y funcionamiento del servidor, incluyendo cómo se reciben y procesan las solicitudes de los clientes.
- El análisis del fichero de cabecera, que contiene las definiciones comunes al cliente y al servidor.
- La implementación de las funciones RPC en el servidor y su correspondiente invocación desde el cliente.

Desarrollo

calculadora.x

Este es el archivo fuente principal que define la interfaz RPC. Contiene:

- Definición de estructuras: Como datos, que contiene los dos números (dato1 y dato2) que se usarán en las operaciones.
- Definición del programa RPC: CALCULADORA con un identificador único (0x20000001).
- Definición de versiones: VERSIÓN con un número de versión (1).
- Definición de procedimientos: Cada operación (suma, resta, multiplicación, etc.) tiene un identificador único dentro de la versión.

Este archivo es la base del sistema RPC. Define las funciones que estarán disponibles para el cliente y el servidor.

calculadora.h

Este archivo es generado automáticamente por rpcgen a partir de calculadora.x. Contiene:

- Definiciones de estructuras: Como struct datos.
- Prototipos de funciones: Declaraciones de las funciones RPC para el cliente (suma_1, resta_1, etc.) y el servidor (suma_1_svc, resta_1_svc, etc.).
- Definiciones de constantes: Como el identificador del programa (CALCULADORA) y los números de procedimiento (suma = 1, resta = 2, etc.).

Este archivo es compartido entre el cliente y el servidor. Permite que ambos conozcan las estructuras y funciones definidas en el sistema RPC.

calculadora_xdr.c

Este archivo es generado automáticamente por rpcgen. Contiene las funciones XDR (External Data Representation) para serializar y deserializar las estructuras definidas en calculadora.x. Estas funciones son necesarias para transmitir datos entre el cliente y el servidor.

Garantiza que los datos enviados entre el cliente y el servidor sean interpretados correctamente, independientemente de la arquitectura de las máquinas.

calculadeora_clnt.c

Este archivo es generado automáticamente por rpcgen. Contiene las funciones que el cliente utiliza para realizar llamadas RPC al servidor.

Proporciona las funciones que el cliente llama para interactuar con el servidor. Estas funciones encapsulan la lógica de comunicación RPC.

calculadora_svc.c

Este archivo es generado automáticamente por rpcgen. Contiene el código que el servidor utiliza para recibir y procesar las solicitudes RPC del cliente.

Gestiona las solicitudes entrantes del cliente, llama a las funciones implementadas en el servidor (suma_1_svc, resta_1_svc, etc.) y envía los resultados de vuelta al cliente.

calculadora_server.c

Este archivo contiene la implementación de las funciones del servidor. Aquí es donde defines la lógica de cada operación matemática (suma, resta, multiplicación, etc.) que el cliente puede solicitar.

El archivo tiene las siguientes partes principales:

1. Inclusión de cabeceras:
    - Se incluye calculadora.h, que contiene las definiciones generadas por rpcgen.
    - Se incluye <math.h> para realizar operaciones matemáticas avanzadas como potencia y raíz cuadrada.
2. Variable global:
    - NUM_LLAMADAS: Lleva un conteo del número de llamadas realizadas al servidor.
3. Implementación de funciones RPC:
    - Cada función RPC corresponde a una operación matemática definida en calculadora.x. Estas funciones son llamadas por el servidor cuando recibe una solicitud del cliente.
4. Funciones principales
    - suma_1_svc:

        - Realiza la suma de dos números (dato1 y dato2) enviados por el cliente.
        - Incrementa el contador NUM_LLAMADAS
    - resta_1_svc:
        - Realiza la resta de dos números.
    - multiplicacion_1_svc:
        - Realiza la multiplicación de dos números.
    - division_1_svc:
        - Realiza la división de dos números.
        - Maneja el caso de división por cero devolviendo 0.
    - potencia_1_svc:
        - Calcula la potencia de dato1 elevado a dato2.
    - raiz_1_svc:
        - Calcula la raíz cuadrada de un número.
        - Maneja el caso de números negativos devolviendo -1.
    - n_llamadas_1_svc:
        - Devuelve el número total de llamadas realizadas al servidor.
    - menu_1_svc:
        - Devuelve un menú con las opciones disponibles para el cliente

calculadora_cliente.c

Este archivo contiene la lógica del cliente. Aquí es donde defines cómo el cliente interactúa con el servidor, por ejemplo, mostrando un menú y llamando a las funciones RPC.

El archivo tiene las siguientes partes principales:

1. Inclusión de cabeceras:
    - Se incluye calculadora.h, que contiene las definiciones generadas por rpcgen.
2. Función principal (main):
    - Configura el cliente y llama a la función calculadora_1 para interactuar con el servidor.
3. Función calculadora_1:
    - Contiene la lógica principal del cliente.
    - Muestra un menú, solicita datos al usuario y llama a las funciones RPC correspondientes.
4. Lógica del cliente
    a. Mostrar el menú: Llama a la función menu_1 para obtener el menú desde el servidor.

b. Solicitar una operación: El cliente solicita al usuario que seleccione una operación y los datos necesarios.
c. Llamar a las funciones RPC: Dependiendo de la opción seleccionada, el cliente llama a la función RPC correspondiente.
d. Mostrar el resultado: El cliente muestra el resultado devuelto por el servidor.
e. Salir del programa: Si el usuario selecciona la opción 0, el cliente finaliza.

Flujo de ejecución

Cliente:

El cliente llama a una función RPC (por ejemplo, suma_1).

La función en calculadora_clnt.c serializa los datos y los envía al servidor.

Servidor:

El servidor recibe la solicitud en calculadora_svc.c.

Llama a la función correspondiente en calculadora_server.c (por ejemplo, suma_1_svc).

La función realiza la operación y devuelve el resultado.

Cliente:

El cliente recibe el resultado del servidor.

La función en calculadora_clnt.c deserializa los datos y los devuelve al cliente.

El cliente muestra el resultado al usuario.

Memoria

Modificar fichero calculadora_server.c y calculadora_client.c. Y el Makefile.calculadora cada vez que haga make -f Makefile.calculadora se regenera el archivo (para el make, hay que ir al docker y situar en servidor e instalar el make con el comando “apt install make”. He tenido problema

```
E: Package 'make' has no installation candidate
```

y tuve que hacer el “apt update” y “apt upgrade", porque no se encontraba el archivo. También necesita instalar el “apt install libnsl-dev”) y hay que añadir esta dos lineas de nuevo:

CFLAGS += -g -I/usr/include/tirpc

LDLIBS += -lnsl -ltirpc -lm

Al ejecutar calculadora.x se genera automáticamente estos ficheros como explicado previamente:

calculadora_svc.c calculadora.h calculadora_clnt.c calculadora_xdr.c

calculadora.x

```
rpcuserserver@15e5bddb0ad8:~/p2$ rpcgen -a calculadora.x
rpcuserserver@15e5bddb0ad8:~/p2$ ls
Makefile.calculadora  calculadora_client.c  calculadora_svc.c
calculadora.h         calculadora_clnt.c    calculadora_xdr.c
calculadora.x         calculadora_server.c
```

Problema encontrado de la práctica:

A la hora de mostrar el menú al cliente, no se muestra por completo el menú, sino que solo se muestra una letra “M”. No se me ha encontrado la solución ya que parece que no me ha fallado a la hora de llamar al ‘menu_1’. Y para solucionar esto, he decidido hacer una printf con el menú en el lado cliente:

(printf("Seleccione una opción 1-Suma;2-Resta;3-Mul;4-Div;5-Pow;6-Raiz;7-NumLlamada (0 para salir): ");)

```
rpcuserclient@937a38ebb92c:~/p2$ ./cliente 172.17.0.2
M
Seleccione una opción 1-Suma;2-Resta;3-Mul;4-Div;5-Pow;6-Raiz;7-NumLlamada (0 para salir): 0
Saliendo...
```

Compilación y ejecución

Compilación: Abre una terminal y navega al directorio donde se encuentran los archivos fuente. Luego, ejecuta el siguiente comando para compilar:

- gcc -o servidor -I /usr/include/tirpc calculadora_server.c calculadora_svc.c calculadora_xdr.c -l tirpc -lm
- gcc -o cliente -I /usr/include/tirpc calculadora_client.c calculadora_clnt.c calculadora_xdr.c -l tirpc

Ejecución: En la terminal, ejecuta el servidor con el siguiente comando:

- ./cliente <IP_del_servidor>
- ./servidor

Prueba y resultado

Servidor:

```
rpcuserserver@15e5bddb0ad8:~/p2$ gcc -o servidor -I /usr/include/tirpc calculadora_server.c calculadora_svc.c calculador
a_xdr.c -l tirpc -lm
rpcuserserver@15e5bddb0ad8:~/p2$ ./servidor
```

Cliente:

```
rpcuserclient@937a38ebb92c:~/p2$ gcc -o cliente -I /usr/include/tirpc calculadora_client.c calculadora_clnt.c calculadora_xdr.c -l
tirpc
rpcuserclient@937a38ebb92c:~/p2$ ./cliente 172.17.0.2
M
Seleccione una opción 1-Suma;2-Resta;3-Mul;4-Div;5-Pow;6-Raiz;7-NumLlamada (0 para salir): 1
Ingrese dos números para SUMA(separados por espacio): 2 2
Resultado: 4.000000
M
Seleccione una opción 1-Suma;2-Resta;3-Mul;4-Div;5-Pow;6-Raiz;7-NumLlamada (0 para salir): 2
Ingrese dos números para RESTA(separados por espacio): 3 1
Resultado: 2.000000
M
Seleccione una opción 1-Suma;2-Resta;3-Mul;4-Div;5-Pow;6-Raiz;7-NumLlamada (0 para salir): 3
Ingrese dos números para MULTIPLICACION(separados por espacio): 2 3
Resultado: 6.000000
M
Seleccione una opción 1-Suma;2-Resta;3-Mul;4-Div;5-Pow;6-Raiz;7-NumLlamada (0 para salir): 4
Ingrese dos números para DIVISION(separados por espacio): 4 2
Resultado: 2.000000
M
Seleccione una opción 1-Suma;2-Resta;3-Mul;4-Div;5-Pow;6-Raiz;7-NumLlamada (0 para salir): 5
Ingrese la base y el exponente para POTENCIA (separados por espacio): 2 3
Resultado: 8.000000
M
Seleccione una opción 1-Suma;2-Resta;3-Mul;4-Div;5-Pow;6-Raiz;7-NumLlamada (0 para salir): 6
Ingrese un número para RAIZ: 9
Resultado: 3.000000
```

Conclusión

En esta práctica, hemos desarrollado una aplicación distribuida utilizando la herramienta RPCGEN, que nos ha permitido implementar una calculadora distribuida con diversas operaciones aritméticas. A lo largo del desarrollo, hemos aprendido a:

- **Definir la Interfaz RPC**: Utilizando el archivo calculadora.x, hemos especificado las estructuras de datos y los procedimientos que el servidor implementará y que los clientes podrán invocar.
- **Generar Código Automáticamente:** Con RPCGEN, hemos generado automáticamente los ficheros de cabecera, los stubs del cliente y del servidor, y las funciones de serialización (XDR), simplificando así el proceso de desarrollo.
- **Implementar Funciones RPC en el Servidor**: En el archivo calculadora_server.c, hemos implementado las funciones que realizan las operaciones aritméticas, manejan las solicitudes de los clientes y devuelven los resultados.
- **Desarrollar el Cliente**: En el archivo calculadora_cliente.c, hemos desarrollado la lógica del cliente para interactuar con el servidor, enviar solicitudes, recibir respuestas y manejar errores.
- **Comprender el Funcionamiento de RPC**: Hemos adquirido una comprensión profunda del modelo de llamada a procedimiento remoto (RPC), incluyendo cómo se empaquetan y desempaquetan los datos, cómo se manejan las comunicaciones entre el cliente y el servidor, y cómo se gestionan los errores.

En resumen, esta práctica nos ha proporcionado una valiosa experiencia en el desarrollo de aplicaciones distribuidas utilizando RPCGEN, y nos ha permitido aplicar conceptos teóricos en un contexto práctico. Hemos aprendido a definir interfaces, generar código automáticamente, implementar funciones en el servidor, desarrollar clientes y comprender el funcionamiento de RPC, lo que nos prepara para abordar proyectos más complejos en el futuro.

Anexo

- Código fuente
- Campus Virtual
- Copilot