Ejercicio Evaluable 3 - Comunicación por RPC

Adrián Martínez Cruz 100451213

César López Navarro 100451326

Grupo 86

1. Introducción

Este proyecto presenta la implementación de un sistema distribuido que emplea comunicación entre un servidor y un cliente mediante Llamadas a Procedimientos Remotos (RPC) utilizando el protocolo TCP. Se utiliza la librería rpc/rpc.h del lenguaje C para facilitar esta comunicación. Para la interacción del cliente con el servidor, se han creado interfaces del tipo client_*(), las cuales automatizan el proceso de comunicación entre ambos. El servidor envía los resultados de las operaciones ejecutadas al cliente.

2. Protocolo de comunicación

Nuestro protocolo de comunicación se basa en el uso de la librería RPC para gestionar todas las interacciones entre el cliente y el servidor. Aquí está cómo funciona:

1. El cliente se conecta al servidor especificando su dirección IP.
2. Se llama al procedimiento remoto correspondiente, pasando al servidor los parámetros proporcionados por el cliente.
3. Se verifica la variable 'retval' para asegurar que la ejecución del procedimiento remoto fue exitosa.
4. El cliente recibe el resultado final de la operación.

Todo este proceso se encuentra implementado en el archivo de cabecera claves-rpc.h. Esto significa que este archivo se encarga de toda la comunicación entre el cliente y el servidor, sin necesidad de que ninguno de los dos implemente algún aspecto del protocolo de comunicación.

3. Diseño

Programa

Las llamadas a procedimientos remotos requieren de una estructura que facilite la comunicación entre máquinas:

- Por un lado, tenemos el programa del servidor y el programa del cliente:

        - El servidor aloja la implementación de los servicios junto con un bucle infinito que permite su ejecución constante para recibir y procesar operaciones. Los stubs, al comunicarse con su implementación, obtienen el código necesario para procesar uno de sus servicios.
        - El cliente se limita a realizar solicitudes de diferentes servicios y esperar posteriormente una respuesta con los resultados solicitados.
    - Por otro lado, están los stubs tanto del servidor como del cliente. Actúan como intermediarios y permiten la comunicación entre las partes. En este proyecto, los stubs son claves-rpc_svc.c para el servidor y claves-rpc_clnt.c para el cliente.

Es importante destacar que el archivo claves-rpc.x contiene las funciones descritas en el enunciado del ejercicio. Estas funciones tienen como parámetros de entrada un struct llamado args_struct que contiene key, value1, N_value2, V_value2 y value3; que se utiliza también para la salida.

Las demás decisiones de diseño tomadas anteriormente se mantienen en este proyecto. Además, al tener que incluir rpcgen en el Makefile, para que no se eliminen los archivos de código proxy-rpc.c y servidor-rpc.c, estos no se han incluido en los targets de rpcgen.

Errores

Para manejar los errores durante el procesamiento, se utilizarán las funciones de errores proporcionadas por la librería rpc/rpc.h:

- En caso de fallo al crear el cliente, se llamará a clnt_pcreateerror().
- Si la comunicación entre el cliente y el servidor no funciona correctamente, principalmente debido a la librería, se utilizará clnt_perror().

Además de estas funciones, el archivo de cabecera ya incluye su propio tratamiento de errores.

4. Pruebas

Para validar este proyecto, se han creado cuatro archivos de tipo cliente que enviarán múltiples solicitudes al servidor simultáneamente. Dado que en proyectos anteriores de Colas de Mensajes y Sockets TCP ya se ha verificado la validez de las funciones ejecutadas por el servidor al recibir las solicitudes, en este proyecto se enfocará principalmente en el uso de RPC en el programa.

5. Compilar y generar los ejecutables

Para la compilación y generación de ejecutables y librerías se ha utilizado un Makefile. Será necesario ejecutar en terminal en el directorio *ejercicio_evaluable_3* dos comandos en dos terminales.

En la primera:

```
./run-server.sh
```

Y en la segunda:

```
./run-clients.sh
```

El primer script corresponde a la compilación del programa y el ejecutable del servidor .

El segundo script ejecuta otros scripts de clientes que dan valores a IP_TUPLAS al mismo tiempo, o casi al mismo tiempo, para poder validar la concurrencia de la aplicación.