Sistemas Distribuidos

Grado de Ingeniería en Informática
Curso 2023-2024

Ejercicio Evaluable 3

Integrantes del Grupo:
María Isabel Hernández Barrio (100472315)
Blanca Ruiz González(100472361)

En este informe, se presenta el diseño, implementación y evaluación de un sistema distribuido de elementos clave-valor utilizando RPC. El sistema permite el almacenamiento y uso de pares clave-valor, donde el valor asociado a cada clave puede ser una cadena de caracteres y/o un vector de números. Consta de un servidor que gestiona los datos y clientes que interactúan con el servidor a través de llamadas RPC.

El sistema se compone de las siguientes partes:

- claves_rpc.x:

El contenido de claves.x incluye la definición de la estructura respuesta y el programa CLAVES_PROG. Dentro de CLAVES_PROG definimos las operaciones disponibles para los clientes, como inicialización, establecimiento de valores, obtención de valores, modificación de valores, eliminación de claves y verificación de la existencia de una clave en el sistema.

Estas definiciones están escritas en un lenguaje que es utilizado por herramientas de generación de código RPC. Usamos el comando “rpcgen” para generar el código necesario.

“rpcgen” nos genera los siguientes archivos:

- claves_rpc.h:

Contiene las declaraciones de las estructuras de datos y las operaciones definidas en claves_rpc.x, así como las firmas de las funciones que serán implementadas en el servidor y utilizadas por los clientes.

- claves_rpc_svc.c:

Contiene las implementaciones de las funciones del servidor que manejan las operaciones definidas en claves_rpc.x.

- claves_rpc_xdr.c:

Contiene las funciones de codificación y decodificación de datos necesarias para la comunicación entre el cliente y el servidor utilizando el protocolo RPC.

- claves_rpc_clnt.c:

Contiene las implementaciones de las funciones del cliente necesarias para llamar a las operaciones remotas definidas en el archivo claves_rpc.x

- Makefile:

Este archivo es un archivo Make utilizado para compilar y construir el servidor y el cliente del sistema distribuido basado en RPC generado a partir del archivo claves_rpc.x. Incluye la creación de la biblioteca libclaves.so.

Un vez rpcgen nos ha generado todos estos archivos, hemos diseñado nuestros archivos claves.c, servidor.c y cliente.c:

- claves.c:

El archivo claves.c implementa las funciones del cliente para comunicarse con el servidor. Cada función en claves.c se encarga de llamar a una operación remota específica en el servidor. Estas son el init, set_value, modify_value, delete_key y exist. Estas funciones manejan la creación del cliente RPC, realizan la llamada remota al servidor (mediante las funciones que el rpcgen había generado) y gestionan posibles errores de comunicación.

- servidor.c:

El servidor proporciona las funciones init, set_value, modify_value, delete_key y exist. Cada operación RPC se implementa como una función de servicio (_svc) que toma los parámetros necesarios y devuelve un resultado booleano indicando el éxito de la operación. Además, se define una función claves_rpc_prog_1_freeresult para liberar la memoria utilizada por los resultados de las operaciones RPC. Las funciones que utilizamos para gestionar los ficheros de datos son las mismas que en el ejercicio anterior (guardarDatosFichero, leerDatosFichero, etc), solo que hemos tenido que hacer algunas modificaciones para manejar los datos que se guardan en los ficheros: ahora se guardan en forma de struct “datos”, de manera que tenemos que transferir los datos que recibimos por parámetros a este tipo de structs al escribir o meter los datos del struct datos en el struct respuesta al leer datos de un fichero.

- cliente.c:

Hemos utilizado el mismo archivo que para los ejercicios anteriores. Implementa un cliente para interactuar con el servidor RPC definido en “claves_rpc.x”. El cliente

realiza diversas pruebas de las operaciones disponibles en el servidor, incluyendo la inicialización del sistema, establecimiento, obtención, modificación y eliminación de valores, así como la verificación de la existencia de claves. Cada prueba se realiza llamando a las funciones correspondientes del cliente definidas en “claves.h”, como “init”, “set_value”, “get_value”, “modify_value”, “delete_key”, y “exist”. Además, se realizan comprobaciones adicionales para verificar el comportamiento del sistema en casos como eliminar una clave que no existe o establecer un valor con un número de elementos fuera de rango. Se muestran mensajes de estado para indicar el éxito o fallo de cada operación. Una vez completadas las pruebas, se liberan los recursos y se finaliza el programa.

Ejecución: después de hacer ejecutar el Makefile con “make”, ejecutamos el servidor con ./servidor y al cliente hay que pasarle la dirección IP del servidor como variable de entorno, de forma que hacemos “env IP_TUPLAS=localhost ./cliente”.

Pruebas:

```
a0472361@guernika:~/ejercicio$ env IP_TUPLAS=localhost ./cliente
Init OK.
Set_value OK.
Valor1 debe ser Hola: Hola
N_valor2 debe ser 3: 3
V_valor2 debe ser 1.0, 2.0, 3.0
  V_valor2[0]: 1.000000
  V_valor2[1]: 2.000000
  V_valor2[2]: 3.000000
Get_value OK.
Valor1 debe ser Adios: Adios
N_valor2 debe ser 3: 3
V_valor2 debe ser 4.0, 5.0, 6.0
  V_valor2[0]: 4.000000
  V_valor2[1]: 5.000000
  V_valor2[2]: 6.000000
Resultado de Exist debe ser 1: 1
Modify Value y Exist OK.
Resultado de Exist debe ser 0: 0
Delete Key OK.
Error en el delete_key 2 ( OK. ERROR ESPERADO)
Set_value OK.
Resultado de Exist debe ser 0: 0
Init 2 TEST OK.
N_value2 out of range
: Transport endpoint is not connected
Error en el set_value ( OK. ERROR ESPERADO)
Fin de los tests, todo OK
```

El mensaje que sale de N_value2 out of range: Transport endpoint is not connected, es un error esperado (el que se indica justo a continuación) que se obtiene al hacer un set_value con un valor que no está entre 1 y 32, como el 0.