Ingeniería Informática

Curso 2022/2023

Sistemas Distribuidos

Ejercicio Evaluable 3:

RPC

Grupo 82

<u>Autores:</u>

Ana Martín Magariño - 100436334

Ilse Mariana Córdova Sánchez - 100501460

Patrycja Wysocka - 100502920

ÍNDICE

1. Introducción.

En esta memoria se describe el desarrollo de un servicio de elementos ***clave-valor1-valor2-valor3*** utilizando llamadas a procedimientos remotos (RPC) y el modelo ONC RPC en lenguaje de programación C. El objetivo de este ejercicio es mantener la misma interfaz para los clientes, la cual consta de siete servicios: *init, set_value, get_value, modify_value, delete_key, exist y copy_key*.

Detallamos el diseño y especificación de la interfaz de servicio, el código del servidor concurrente encargado de gestionar las estructuras de datos, la implementación de los servicios en el lado del cliente y la creación de una biblioteca dinámica que será utilizada por las aplicaciones de usuario. Asimismo, presentamos un ejemplo de código de un cliente que utiliza las funciones anteriormente mencionadas.

2. Diseño.

ONC RPC (Remote Procedure Call) es un modelo de programación que permite que los procesos en diferentes sistemas se comuniquen y ejecuten funciones a través de la red de manera transparente.

En el modelo ONC RPC, los procedimientos remotos se definen en un archivo de especificación de RPC, que luego se utiliza para generar el código de cliente y servidor en el lenguaje de programación deseado. Los clientes invocan procedimientos remotos a través de llamadas a funciones, y los servidores responden a estas llamadas ejecutando los procedimientos correspondientes.

El modelo ONC RPC utiliza un enfoque basado en procedimientos para la comunicación entre procesos, lo que significa que las funciones se definen en el servidor y se invocan desde el cliente como si estuvieran siendo llamadas localmente. Además, ONC RPC utiliza un protocolo de transporte transparente para enviar y recibir datos, lo que permite que los procesos se comuniquen independientemente de la plataforma o sistema operativo.

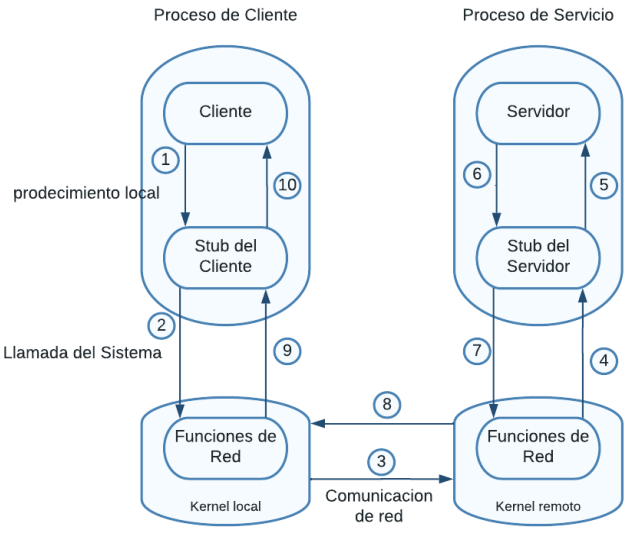

Figura 1. Modelo RPC.

1. ***claves.c*** y ***claves.h***: Estos archivos contienen las funciones y estructuras necesarias para el manejo de claves en el servidor. En claves.c se implementan implementan los servicios del lado del cliente, y en *claves.h* se declaran las estructuras y funciones necesarias.
2. ***cliente.c***: Este archivo contiene el código fuente del cliente que utiliza los servicios proporcionados por el servidor. Incluye la función principal que interactúa con el usuario y llama a las funciones de RPC para realizar solicitudes al servidor.
3. ***project_rcp.h***: Este archivo es generado por *rpcgen* y contiene las declaraciones de las funciones de RPC, las estructuras de datos necesarias y las definiciones de constantes.
4. ***project_rcp.x***: Este archivo es la especificación de la interfaz de programación remota (RPC) que se utilizará para generar el código fuente del servidor y del cliente. Incluye las definiciones de los tipos de datos y funciones que se utilizarán.
5. ***project_rcp_clnt.c***: Este archivo es generado por *rpcgen* y contiene las funciones necesarias para llamar a los servicios remotos proporcionados por el servidor.
6. ***project_rcp_xdr.c***: Este archivo es generado por *rpcgen* y contiene las funciones necesarias para convertir los datos de un formato a otro para su transmisión a través de la red.
7. ***servidor.c***: Este archivo contiene el código fuente del servidor que implementa las funciones de los servicios definidos en la interfaz RPC. Incluye la función principal que espera por solicitudes entrantes del cliente y llama a las funciones correspondientes.
8. ***servicios.c*** y ***servicios.h***: Estos archivos contienen las funciones y estructuras necesarias para implementar los servicios definidos en la interfaz RPC. En *servicios.c* se implementan las funciones que realizan las acciones necesarias y en *servicios.h* se declaran las estructuras y funciones necesarias.
9. ***servicios_help.c*** y ***servicios_help.h:*** Estos archivos contienen funciones de ayuda utilizadas por los servicios principales para realizar tareas específicas. En *servicios_help.c* se implementan las funciones y en *servicios_help.h* se declaran las estructuras y funciones necesarias.

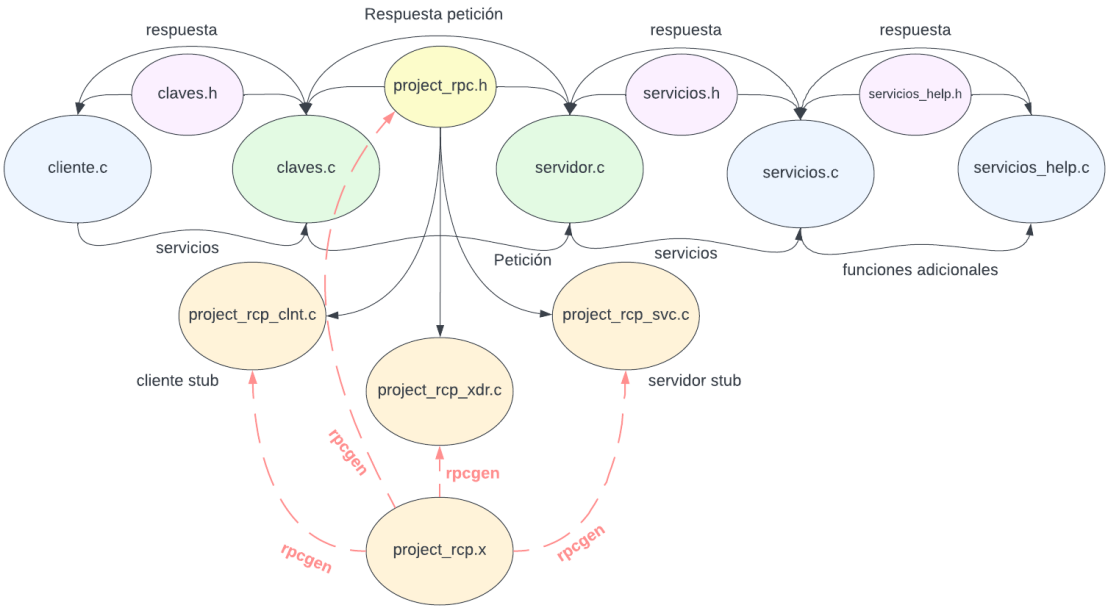

Figura 2. Estructura de diseño de código.

3. Estructura de archivos utilizado en la compilación del servidor, de la biblioteca libclaves.so y del cliente.

El servidor se apoya de *servicios.h* y de *project_rpc.h* para la implementación de su funcionalidad. Por otro lado, el cliente utiliza *claves.h* y *project_rpc.h*. Además, para compilar y generar el ejecutable del cliente, este se vincula a la biblioteca compartida *libclaves.so*. Es importante destacar que al mismo tiempo, *project_rpc.h* incluye a *project_rpc_clnt.c*, *project_rpc_xdr.c* y *project_rpc_svc.c*.

En el *Makefile* del proyecto, podemos apreciar las siguientes secciones relacionadas al cliente:

SOURCES_CLNT.c =
SOURCES_CLNT.h =
SOURCES_SVC.c =
SOURCES_SVC.h = *SOURCES.x = project_rpc.x*

TARGETS_CLNT.c = project_rpc_clnt.c client.c project_rpc_xdr.c claves.c

OBJECTS_CLNT = $(SOURCES_CLNT.c:%.c=%.o)$(TARGETS_CLNT.c:%.c=%.o)

$(CLIENT) : $(OBJECTS_CLNT)
$(LINK.c) -o $(CLIENT) $(OBJECTS_CLNT) $(LDLIBS)

Como podemos apreciar, el cliente utiliza tanto *targets_clnt* como *objects_clnt* para su compilación. Al mismo tiempo, *targets_clnt* está compuesto por *project_rpc_clnt.c*, *client.c, project_rpc_xdr.c* y *claves.c*.

Por otro lado, en el Makefile también podemos observar el siguiente código para el servidor:

SOURCES_CLNT.c =
SOURCES_CLNT.h =
SOURCES_SVC.c =
SOURCES_SVC.h = *SOURCES.x = project_rpc.x*

TARGETS_SVC.c = project_rpc_svc.c server.c project_rpc_xdr.c servicios_help.c servicios.c

OBJECTS_SVC = $(SOURCES_SVC.c:%.c=%.o) $(TARGETS_SVC.c:%.c=%.o)

$(SERVER) : $(OBJECTS_SVC)
$(LINK.c) -o $(SERVER) $(OBJECTS_SVC) $(LDLIBS)

Al igual que el cliente, el servidor también utiliza *targets* y *objects* para su compilación. Además, *targets_svc.c* utiliza a *project_rpc_svc.c, server.c project_rpc_xdr.c, servicios_help.c* y *servicios.c*.

4. Compilación y ejecutable del cliente y del servidor.

El archivo Makefile es utilizado para compilar y enlazar un cliente y un servidor para un proyecto de llamada a procedimiento remoto (RPC). El comentario en la primera línea del archivo indica que este es un Makefile generado por *rpcgen*, una herramienta que se utiliza para generar código C para la comunicación a través de procedimientos remotos (RPC).

El comando *rpcgen* se utiliza para generar código C a partir de un archivo de especificación de RPC (en este caso, *project_rpc.x*) y que el código de RPC generado tendrá los siguientes archivos fuente:

- *project_rpc_clnt.c*: código de RPC del lado del cliente
- *project_rpc_svc.c*: código de RPC del lado del servidor
- *project_rpc_xdr.c*: código de serialización XDR
- *cliente.c*: función principal para el programa del cliente
- *servidor.c*: función principal para el programa del servidor

El Makefile también incluye varias variables que se utilizan en todo el archivo. Por ejemplo, la variable *SOURCES_CLNT.c* especifica los archivos fuente utilizados para construir el ejecutable del cliente, y la variable *LDLIBS* especifica las bibliotecas que se deben vincular con el ejecutable.

La regla "all" depende de las reglas del objetivo "*CLIENT*" y "*SERVER*". Estas reglas compilan y enlazan los archivos fuente y objeto relevantes para generar los ejecutables "*client*" y "*server*". *"client"* se compila utilizando los objetos de los clientes, mientras que la regla *"server"* utiliza los objetos del servidor. Finalmente, *"clean"* elimina todos los archivos objeto, los ejecutables y cualquier archivo generado.

En general, el Makefile se utiliza para automatizar el proceso de construcción de un programa que utiliza llamadas de procedimiento remoto (RPC) para comunicarse entre procesos de cliente y servidor.

5. Pruebas del servicio desarrollado.

Para el código desarrollado se realizaron pruebas para comprobar el correcto funcionamiento de las funciones programadas. Estas pruebas fueron las mismas que realizamos en la Práctica 1. Cabe mencionar que para todas las pruebas de funcionalidad obtuvimos los resultados esperados. En esta ocasión, no se realizaron pruebas enfocadas a la conexión TCP / UDP ya que RPC se encarga de generar el código para esta tarea. Sin embargo, para el reporte de esta práctica decidimos enfocarnos en las pruebas relacionadas a RPC.

#Test	Tipo	Output esperado	Output obtenido
1	Usar el comando rpcinfo.	Recibir el reporte del estatus del servidor.	(El output esperado se anexa en la figura 3.)
2	Tanto cliente como servidor tienen el mismo número de versión.	La conexión se realiza exitosamente y las funciones se ejecutan de manera correcta.	La conexión se realiza exitosamente y las funciones se ejecutan de manera correcta.
3	Cliente y servidor tienen números de versión diferentes.	Error en la compilación del proyecto.	(El output esperado se anexa en la figura 4.)

Tabla 1. Pruebas del servicio desarrollado.

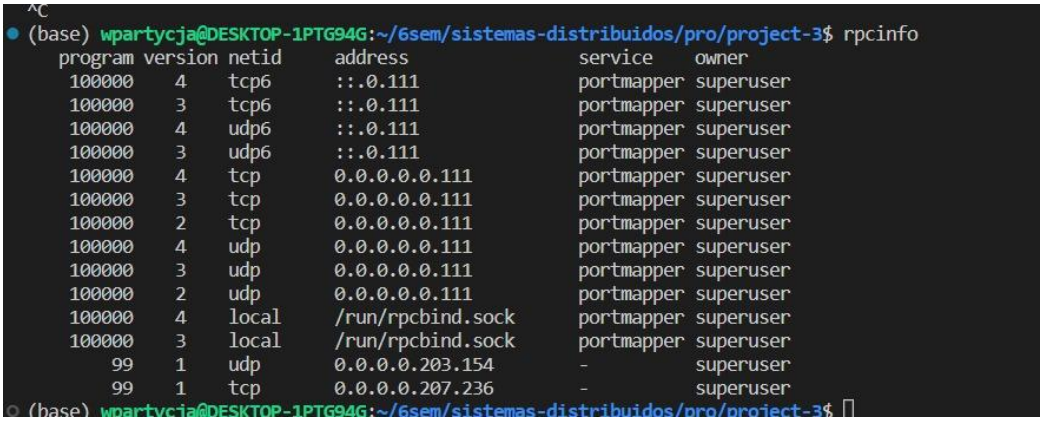

Figura 3. Output de test #1.

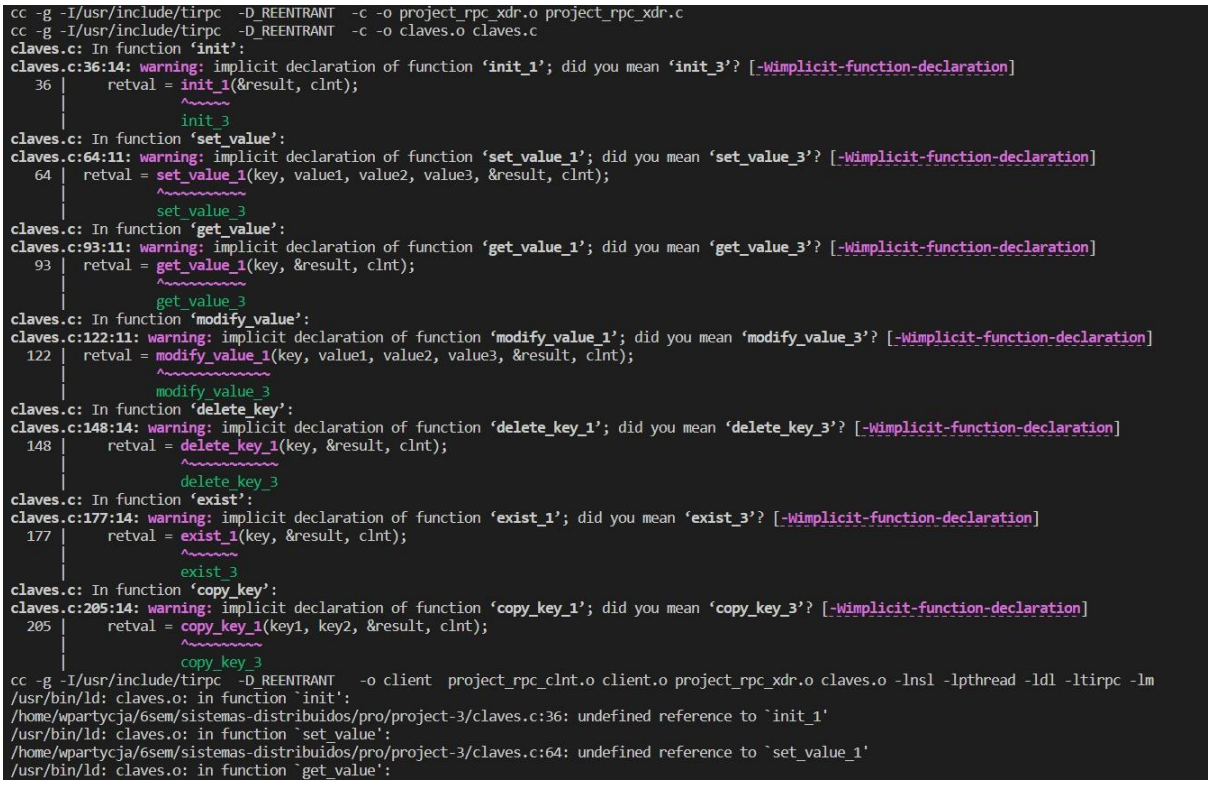

Figura 4. Output de test #3.

6. Conclusión.

El desarrollo de esta práctica nos resultó compleja al principio, esto debido a que no estábamos familiarizadas con todos los cambios que debíamos hacer en el código para lograr que funcionara el programa de manera adecuada. Las mayores dificultades con las que nos encontramos al realizar el proyecto fueron las siguientes: el comprender cuáles ficheros debíamos descartar y editar después de generarlos, hacer cambios para que nuestras funciones regresaran un valor booleano y alocar memoria para ciertas variables que no necesitaban de esto en prácticas pasadas.

A pesar de los contratiempos con los que nos encontramos para realizar la práctica, consideramos que hemos desarrollado un código funcional que cumple con los requisitos establecidos por el enunciado del laboratorio. Además, este trabajo también nos permitió concretar los conceptos de RPC estudiados en clase para así poder implementarlos. .