Universidad Carlos III
Curso Sistemas Distribuidos 2023-24
Ejercicio evaluable 1

Rubén de Arriba Viejo (100471975)
Óscar Hontoria Herrador (100471920)

Índice

1. Introducción

Este documento presenta los razonamientos y pasos seguidos para implementar el diseño y la implementación del servicio distribuido para el almacenamiento de tuplas. En este sistema, uno o más clientes mediante el uso de colas POSIX, podrán acceder a las tuplas almacenadas en el servidor, pudiendo realizar distintas operaciones sobre ellas. Las tuplas se componen de <clave-valor1-valor2>.

Para hacerlo se ha implementado un servidor para la gestión de las claves almacenadas, un cliente para solicitar operaciones y una biblioteca dinámica que define las operaciones.

2. Componentes

Para lograr el funcionamiento del sistema, se ha seguido la arquitectura propuesta por el enunciado, teniendo un fichero con el código correspondiente al servidor, otro fichero para simular los usuarios que se conecten al servidor y un fichero de funciones que permita acceder de una forma transparente para los usuarios al servidor. De esta forma, con un único fichero de funciones y un único servidor, se pueden conectar un número indeterminado de usuarios.

Adicionalmente, hemos generado distintos ficheros para la definición de las estructuras que se utilizan tanto en las comunicaciones como en el almacenamiento de los datos. Para el almacenamiento de datos se pensó en crear una estructura para almacenar los vectores de punteros para cada una de las variables de la tupla, pero, por simplicidad a la hora de referenciar las variables, se decidió almacenar las variables de forma separada.

Los mensajes están formados por seis campos diferentes:

- op: identificador de la operación que se debe realizar. Toma valores de 0 a 5.
- cola_cliente: cadena de caracteres que identifica la cola para enviar la respuesta al usuario. Su composición es del prefijo “Cola-” concatenado con el identificador de proceso del cliente.
- clave: identifica la clave de la tupla.
- valor_1: cadena de caracteres que identifica el valor1 de la tupla.
- n_elem: identificador de la longitud del vector que se desea almacenar.
- vector: es el vector de variables double.

Para el almacenamiento se han definido tres variables diferentes:

- int *keys: almacena las llaves guardadas de las tuplas.
- char **valores_1: almacena las cadenas de caracteres introducidas en el campo Value1.
- int* num_elements: almacena el tamaño de cada vector de valores double almacenados.
- double **vectores: almacena los vectores de valores double.

A continuación se explica cada uno de los ficheros que actúan para la correcta ejecución del programa:

- **Servidor.c**

Este archivo contiene toda la lógica de la ejecución de cada operación y se encarga del almacenamiento en memoria de las tuplas. También se encarga de recibir los datos a través de colas de mensajes POSIX y, cada vez que recibe un mensaje, se crea un hilo para que ejecute la operación y el servidor pueda seguir recibiendo operaciones.

- **cliente.c**

Este documento representa el lado del cliente y se encarga de recopilar la información del cliente, la tupla: clave, valor 1, número de elementos en el vector y el vector. Dependiendo de la operación podría solo requerir la clave de la tupla, como podría ser el caso de la operación delete_key. Una vez se ha obtenido la información se hace una llamada al la función de la operación específica que está realizando, esta es procesada por la librería dinámica. Este documento contiene en su cabecera claves.h, para que el cliente sepa de la existencia de las diferentes operaciones.

- **claves.c**

Contiene las funciones que se encarga de hacer la comunicación por colas con el servidor. Se implementa una biblioteca para hacer la lógica de comunicación entre cliente y servidor. La lógica de colas que se encarga de la comunicación del lado del cliente está presente en este documento.

- **mensaje.h**

Los tres documentos (cliente.c, servidor.c y claves.c) comparten este archivo que contiene la estructura del mensaje que tienen en común.

3. Compilación y ejecución

Para facilitar la compilación del programa se ha creado un archivo makefile
que se encarga de la compilación del servidor.c, del cliente.c y de la creación de la librería dinámica libclaves.so.

Para la ejecución del programa se debe seguir los siguientes pasos:

1. Ejecutar el servidor ./servidor
2. Ejecutar el cliente ./cliente
3. Ahora se mostrará un mensaje por la terminal del cliente a la espera de una operación válida.
4. Posteriormente se irá solicitando uno a uno los diferentes datos que requiere la operación.

5. Una vez se hayan entregado los datos el servidor los recibirá a través del cliente y comunicará por su terminal lo que ha recibido.

4. Pruebas

Para probar el programa se han ido introduciendo las operaciones seguidas de los datos en el cliente y se ha ido verificando que el resultado fuese el correcto,
debido a la implementación utilizada, donde el cliente va introduciendo por terminal las diferentes operaciones, con sus datos respectivos, se invita al usuario a probar creando el cliente y enviado varias operaciones. Ejemplo:

```
oscar@oscar-IdeaPad-5-15ITL05:~/Documents/sistemas distribuidos/proyecto/sistemas_distribuidos/ejercicio_1$ ./cliente
Self PID = 8287
Indique la operación a realizar: set_value
Peticion = set_value
Set value
Indique la clave sobre la que se desea hacer set_value(key, valor_1, num_elements, vector): key = 1
Indique el valor 1 para set_value(key, valor_1, num_elements, vector): valor_1 = mi_valor
Indique el numero de elementos de valor 2 para set_value(key, valor_1, num_elements, vector): num_elements = 2
Indique el elemento para set_value(key, valor_1, num_elements, vector): vector[0] = 1.1
Indique el elemento para set_value(key, valor_1, num_elements, vector): vector[1] = 2.2
key = 1, value1 = mi_valor, n_elem = 2 vector[0] = 1.100000 vector[1] = 2.200000
Queue_Name = /Cola-8287

Resultado = 0
Indique la operación a realizar: modify_value
Peticion = modify_value
Indique la clave sobre la que se desea hacer modify_value(key, valor_1, num_elements, vector): key = 1
Indique el valor 1 para modify_value(key, valor_1, num_elements, vector): valor_1 = nuevo_valor
Indique el numero de elementos de valor 2 para modify_value(key, valor_1, num_elements, vector): num_elements = 1
Indique el elemento para hacer modify_value(key, valor_1, num_elements, vector): vector[0] = 3.3
key = 1, value1 = nuevo_valor, n_elem = 1 vector[0] = 3.300000
Queue_Name = /Cola-8287

Resultado = 0
Indique la operación a realizar: get_value
Peticion = get_value
Get value
Indique la clave de la tupla que desea obtener: key = 1
Queue_Name = /Cola-8287
Tupla encontrada:
Clave: 1
Valor1: nuevo_valor
Valor2 longitud: 1
Valor2 vector:
3.30

Resultado = 0
Indique la operación a realizar: delete_key
Peticion = delete_key
Indique la clave sobre la que se desea hacer delete_key(key): key = 1
Queue_Name = /Cola-8287

Resultado = 0
Indique la operación a realizar: exist
Peticion = exist
Indique la clave sobre la que se desea hacer exist(key): key = 1
Queue_Name = /Cola-8287

Resultado = 0
Indique la operación a realizar:
```