Universidad Carlos III
Curso Sistemas Distribuidos 2023-24
Ejercicio evaluable 2

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 sockets, 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 en 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.

Los mensajes son diferentes dependiendo de la operación que se desea realizar, pero todos siguen una estructura similar. Esta estructura son las distintas variables separadas por comas en un vector de variables char, la estructura quedaría: “op,key,n_elementos,vector,valor1”. Op representa el número de operación que se va a realizar, key es la clave de la tupla, n_elementos es el tamaño del vector de valores double, vector son los elementos double separados por comas entre ellos (“vector[0],vector[1]...vector[n_elementos]”) y valor1 es la variable valor1 del vector. De esta forma, se puede pasar la información sin problemas de little-endian big-endian y viceversa, porque al ser char solo ocupan un byte.

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 sockets y, cada vez que recibe un mensaje, se crea un hilo para que ejecute la operación y el servidor pueda seguir recibiendo operaciones. El tamaño del buffer de recepción es de 1807

char porque es el máximo tamaño que puede tener un mensaje (teniendo en cuenta los valores máximos de cada variable).

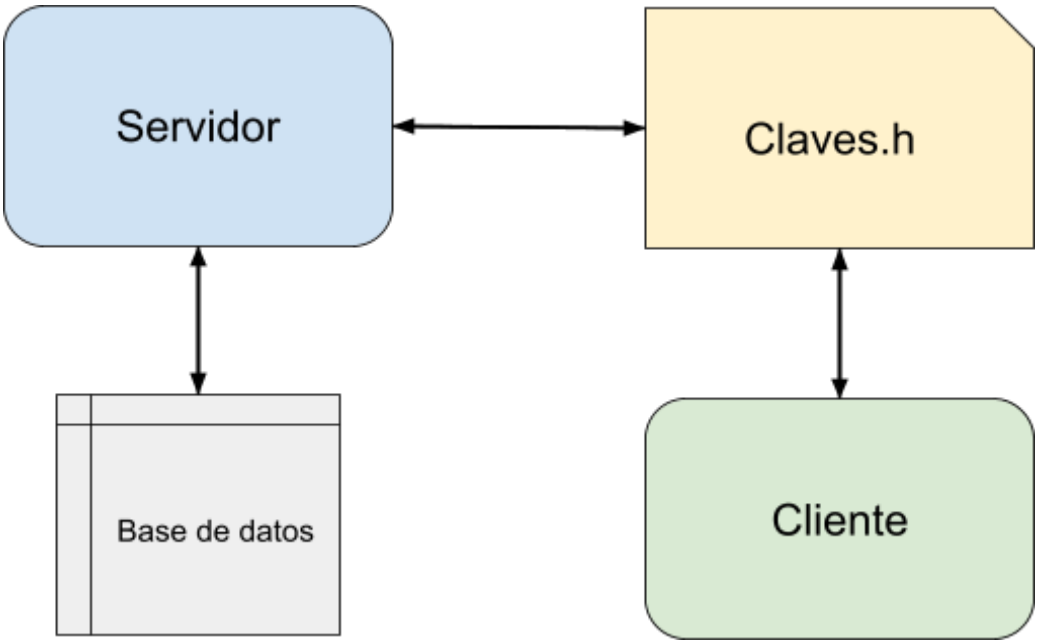
- **cliente.c**

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

De esta forma, la estructura del ejercicio quedaría:



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 make

2. Ejecutar el servidor ./servidor 4200

3. Ejecutar el cliente env IP_TUPLAS=127.0.0.1 PORT_TUPLAS=4200 ./cliente

4. Ahora se mostrará un mensaje por la terminal del cliente a la espera de una operación válida.

5. Posteriormente se irá solicitando uno a uno los diferentes datos que requiere la operación.

6. 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. Ejemplos:

```
oscar@oscar-IdeaPad-5-15ITL05:~/Documents/sistemas distribuidos/proyecto/sistemas_distribuidos/ejercicio_2$ env I
P_TUPLAS=127.0.0.1 PORT_TUPLAS=4200 ./cliente
Indique la operación a realizar: 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 = 1
Indique el numero de elementos de valor 2 para set_value(key, valor_1, num_elements, vector): num_elements = 1
Indique el elemento para set_value(key, valor_1, num_elements, vector): vector[0] = 1
Resultado = 0
Indique la operación a realizar: get_value
Get value
Indique la clave de la tupla que desea obtener: key = 1
Tupla encontrada:
Clave: 1
Valor1: 1
Valor2 longitud: 1
Valor2 vector:
[1.00 ]
Resultado = 0
Indique la operación a realizar: 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 = hola
Indique el numero de elementos de valor 2 para modify_value(key, valor_1, num_elements, vector): num_elements = 2
Indique el elemento para hacer modify_value(key, valor_1, num_elements, vector): vector[0] = 3
Indique el elemento para hacer modify_value(key, valor_1, num_elements, vector): vector[1] = 4
Resultado = 0
Indique la operación a realizar: get_value
Get value
Indique la clave de la tupla que desea obtener: key = 1
Tupla encontrada:
Clave: 1
Valor1: hola
Valor2 longitud: 2
Valor2 vector:
[3.00 , 4.00 ]
Resultado = 0
Indique la operación a realizar: exist
Indique la clave sobre la que se desea hacer exist(key): key = 1
Resultado = 1
Indique la operación a realizar: delete_key
Indique la clave sobre la que se desea hacer delete_key(key): key = 1
Resultado = 0
Indique la operación a realizar: exist
Indique la clave sobre la que se desea hacer exist(key): key = 1
Resultado = 0
```

Este es un ejemplo donde se hacen las operaciones: set_value, get_value, modify_value, get_value, exist, delete_key y exist. En la imagen se ve un ejemplo de funcionamiento

cuando las operaciones se realizan con las pre condiciones adecuadas: la clave usada en set_values no tenía una tupla anteriormente.

Partiendo de que se ha creado una tupla para la clave 1, se van a intentar hacer operaciones que no deberían funcionar.

```
oscar@oscar-IdeaPad-5-15ITL05:~/Documents/sistemas distribuidos/proyecto/sistemas_distribuidos/ejercicio_2$ env I
P_TUPLAS=127.0.0.1 PORT_TUPLAS=4200 ./cliente
Indique la operación a realizar: 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 = 1
Indique el numero de elementos de valor 2 para set_value(key, valor_1, num_elements, vector): num_elements = 1
Indique el elemento para set_value(key, valor_1, num_elements, vector): vector[0] = 1
Resultado = 0
Indique la operación a realizar: 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 = 1
Indique el numero de elementos de valor 2 para set_value(key, valor_1, num_elements, vector): num_elements = 1
Indique el elemento para set_value(key, valor_1, num_elements, vector): vector[0] = 1
Resultado = -1
Indique la operación a realizar: modify_value
Indique la clave sobre la que se desea hacer modify_value(key, valor_1, num_elements, vector): key = 2
Indique el valor 1 para modify_value(key, valor_1, num_elements, vector): valor_1 = 1
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] = 1
Resultado = -1
Indique la operación a realizar: get_value
Get value
Indique la clave de la tupla que desea obtener: key = 2
[Error] No se pudo obtener la tupla para la clave 2 ,compruebe que existe
Resultado = -1
Indique la operación a realizar: delete_key
Indique la clave sobre la que se desea hacer delete_key(key): key = 2
Resultado = -1
Indique la operación a realizar:
```

Vemos como se intenta volver hacer set_value sobre la clave que ya tiene una tupla y se intenta usar las operaciones modify_value, get_value y delete_key sombre una clave que no tiene una tupla.

Ahora vamos a mostrar otros errores posibles, desde parámetros de la funciones erróneos, puerto erróneo o ip errónea.

```
oscar@oscar-IdeaPad-5-15ITL05:~/Documents/sistemas distribuidos/proyecto/sistemas_distribuidos/ejercicio_2$ env I
P_TUPLAS=127.0.0.1 PORT_TUPLAS=4200 ./cliente
Indique la operación a realizar: get_key
[ERROR] Operación no reconocida
Resultado = -1
Indique la operación a realizar: set_value
Indique la clave sobre la que se desea hacer set_value(key, valor_1, num_elements, vector): key = w
[ERROR] El valor de key debe ser un int
Resultado = -1
Indique la operación a realizar: 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 = w
Indique el numero de elementos de valor 2 para set_value(key, valor_1, num_elements, vector): num_elements = 0
Resultado = -1
```

```
oscar@oscar-IdeaPad-5-15ITL05:~/Documents/sistemas distribuidos/proyecto/sistemas_distribuidos/ejercicio_2$ env I
P_TUPLAS=127.0.0.1 PORT_TUPLAS=8200 ./cliente
Indique la operación a realizar: exist
Indique la clave sobre la que se desea hacer exist(key): key = 1
Error en connect
Resultado = -1
Indique la operación a realizar:
```

```
oscar@oscar-IdeaPad-5-15ITL05:~/Documents/sistemas distribuidos/proyecto/sistemas_distribuidos/ejercicio_2$ env I
P_TUPLAS=127 PORT_TUPLAS=4200 ./cliente
Indique la operación a realizar: exist
Indique la clave sobre la que se desea hacer exist(key): key = 1
Error al convertir la dirección IP
Dirección IP del servidor recivida: 127
 (si localhost no funciona prueba con 127.0.0.1)
Resultado = -1
Indique la operación a realizar:
```

El primer ejemplo es uno donde se intenta llamar a una operación inexistente o dar un parámetro no válido. El segundo uno donde el puerto es incorrecto y el tercero uno donde la ip es incorrecta.