Sistemas Distribuidos

Colas de mensajes

Nathalia Moniz Cordova

Alberto García de la Torre Fernández del Moral

17 de marzo del 2024

Diseño del programa

El programa desarrollado se trata de un sistema de comunicación de procesos mediante colas de mensajes. En dicho sistema tenemos principalmente un cliente y un servidor en el cual el cliente es el que realiza las llamadas a ciertas funcionalidades que serán recibidas y ejecutadas por el cliente el cual mandará la respuesta pertinente al cliente. A continuación pasaremos a describir brevemente el diseño de los distintos componentes de nuestro programa.

Claves.c

En este archivo se encuentran las implementaciones de cada una de las funciones que se encargan de comunicarse con el servidor a través de las colas de mensajes y que a su vez permitirán la ejecución de las operaciones de inicializar el servicio (**init**), establecer valores para claves específicas (**set_value**), obtener valores asociados con claves (**get_value**), modificar valores (**modify_value**), eliminar claves (**delete_key**) y verificar la existencia de una clave específica (**exists**).

En cada función se crean y abren las colas tanto del cliente como del servidor para enviar las solicitudes y recibir respuestas, cada solicitud se guarda en una estructura **message** que contiene el tipo de operación a realizar (cada operación tiene un número asociado), la clave, los valores asociados y el nombre de la cola del cliente y dependiendo de la operación a realizar se rellena la estructura con los valores correspondientes. Por ejemplo, en la función *delete_key* solo rellenamos el mensaje con el tipo de operación y la claves, puesto que es todo lo que necesita la función para llevarse a cabo. Cada una de estas peticiones es enviada a la cola del servidor mediante la función *mq_send* y la respuesta de las peticiones es recibida a través de la cola del cliente con la función *mq_receive*, posteriormente se cierran las colas y se devuelve la respuesta para que el cliente sepa si se realizó con éxito o no, además se devuelve un -1 en caso de fallo en las funciones de cola.

Servidor.c

Esta parte del programa se encarga de procesar las peticiones del cliente y enviarle las respuestas. El servidor utiliza hilos para manejar las solicitudes de los clientes de manera concurrente, cada solicitud se recibe a través de la cola del servidor y un hilo se encarga de procesarla. El hilo extrae el mensaje de la cola, lo copia localmente y realiza la operación correspondiente en la estructura de datos, que en este caso hemos decidido usar una lista enlazada por temas de flexibilidad, ya que nos permiten almacenar y manipular datos de manera dinámica, y eficiencia en la inserción y eliminación de datos. Finalmente, después de llamar a las funciones de la lista, se guarda el resultado en una variable que se enviará al cliente mediante su cola.

Cliente.c

Por último, contamos con la parte del cliente que simplemente se encargará de llamar a las funciones, ya que todo el manejo de colas se hace en **claves.c** y en **servidor.c**. Esta sección del cliente está compuesta por una única función principal que toma como argumentos mediante la terminal el tipo de operación y los valores que recibe cada una, a excepción de los valores del vector que en este caso son *doubles* generados aleatoriamente e insertados en el vector en función del número de elementos que desea insertar el cliente. Hemos decidido diseñarlo de esta manera, ya que da más libertad al cliente sobre los valores que quiere insertar y a la hora de hacer pruebas nos otorga más variedad de valores. Dependiendo del tipo de operación se llamará a la función correspondiente y se almacenará su devolución en una variable para al final imprimir un mensaje de error en el caso de que se dé.

Compilación y ejecución del programa

Estando en la carpeta **ejercicio_evaluable1** hay que introducir en un terminal el siguiente comando:

```
Unset
make
```

Cuya salida debe ser similar a la siguiente:

```
Unset
gcc -Wall  -c cliente.c
gcc -Wall  -fPIC -c claves.c -o claves.o
gcc -Wall  -shared -o libclaves.so claves.o
gcc -Wall  -o cliente cliente.o -L. -lclaves
gcc -Wall  -c servidor.c
gcc -Wall  -o servidor servidor.o -L. -lclaves
```

Para la ejecución del programa se tiene que iniciar en primer lugar el servidor con el siguiente comando:

```
Unset
./servidor
```

Y para el cliente primero debemos decirle al sistema operativo dónde buscar las bibliotecas compartidas (archivos de bibliotecas dinámicas) necesarias para ejecutar el programa con el siguiente comando:

```
Unset
export LD_LIBRARY_PATH=/home/username/Ejercicio_evaluable1:$LD_LIBRARY_PATH
```

Y finalmente ejecutar el cliente con los argumentos que se desee:

```
Unset
./cliente 'op' 'key' 'value1' 'N_value2'
```

Pruebas

Las pruebas realizadas consisten en 13 tests que se encuentran dentro de la carpeta “tests”. Cada una de las pruebas comprueba una especificación distinta del enunciado. Hay tres tipos de test: correctos (test 1-5), errores (test 6-11), y de concurrencia (test 12 y 13). A continuación describiré el objetivo de cada test:

- **test1:** comprueba el funcionamiento de la función set_value.
- **test2:** comprueba el funcionamiento de la función get_value.
- **test3:** comprueba el funcionamiento de la función modify_value.
- **test4:** comprueba el funcionamiento de la función delete_key.
- **test5:** comprueba el funcionamiento de la función exist.
- **test6:** muestra un error si no se realiza init al inicio.
- **test7:** muestra un error si se intenta introducir un valor con una llave idéntica a otro existente, o si el tamaño del vector no es válido.
- **test8:** muestra un error si el valor de la llave buscada no se corresponde a ninguno de la lista.
- **test9:** muestra un error si el valor de la llave a modificar no se corresponde a ninguno de la lista.
- **test10:** muestra un error si el valor de la llave a eliminar no se corresponde a ninguno de la lista.
- **test11:** la función exist devuelve el valor 0 en caso de no encontrar un valor en la lista que se corresponde con la llave introducida.
- **test12:** comprueba que es posible ejecutar dos procesos de cliente simultáneamente.
- **test13:** comprueba que es posible insertar una gran cantidad de valores en la lista.

Ejecución de las pruebas

Al ejecutar las pruebas, se debe introducir el nombre del test que deseas realizar. A continuación hay un ejemplo del comando para ejecutar el test1.sh:

```
Unset
./test1.sh
```

Importante: al ejecutar los tests, debes estar situado dentro de la carpeta “tests” en el terminal. De lo contrario, los tests no funcionarán correctamente.