Sistemas distribuidos

Grado en Ingeniería Informática

Ejercicio Evaluable 1: colas de mensaje

Darío Caballero Polo

(100451112, grupo 81, 100451112@alumnos.uc3m.es)

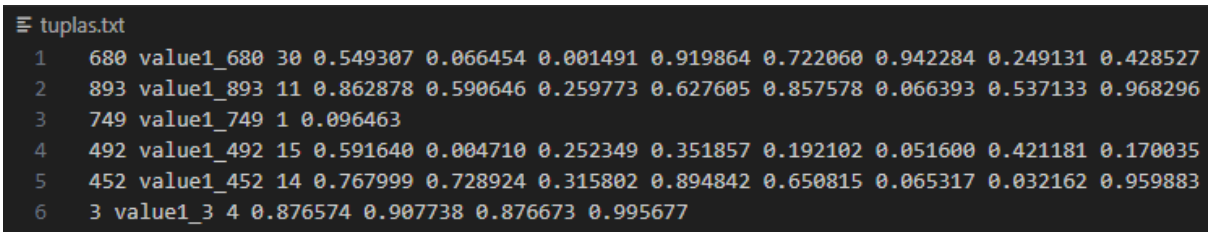
Raúl Miguel Carrero Martín

(100451286, grupo 81, 100451286@alumnos.uc3m.es)

Tabla de contenidos

1.- Introducción

Se pide crear un programa de tipo cliente/servidor mediante colas de mensajes POSIX para almacenar tuplas conteniendo una clave (*key*), un mensaje de texto (*value1*), un número entero conteniendo la cantidad de elementos que habrá en el siguiente elemento de la tupla (*N_value2*) y un array de doubles. Estas tuplas se almacenan en un archivo (*tuplas.txt*) con cada elemento separado por un espacio:



Cabe destacar que las líneas 1, 2, 4 y 5 de la imagen han sido truncadas para facilitar la lectura.

2.- Servidor

En el lado del servidor tenemos un archivo clave llamado *"servidor.c*". En este se llevará a cabo el procesamiento de las peticiones enviadas por los clientes. Cuando comienza la ejecución del servidor, este entra en un bucle infinito que le permite seguir recibiendo y procesando mensajes hasta que recibe la señal de interrupción (CTRL+C). Cuando se recibe una nueva petición, se crea un hilo de ejecución nuevo para procesarla. Para evitar condiciones de carrera en la manipulación del archivo *tuplas.txt*, el procesamiento de las peticiones se realiza en exclusión mutua. Sin embargo, la apertura de la cola de cada cliente y el envío de la respuesta sí que se corre de forma concurrente. Cada petición se procesa leyendo el número de la operación indicando en la petición y llamando a la función correspondiente que la lleve a cabo.

Para mantener cierta estructura de diseño, se ha decidido crear un archivo donde se encuentran las funciones de procesamiento de peticiones por parte del servidor, llamado *"funciones_servidor.c*". En el archivo *"funciones_servidor.h*" se definen las librerías pertinentes, así como la interfaz de cada función. Este código también se compila en una librería dinámica, "*libserverclaves.so*", con la cual se enlaza el servidor.

A continuación se explican brevemente las funciones del servidor:

1. ***init()***
   Se intenta abrir el archivo *tuplas.txt*. Si este no existe, se crea uno nuevo. En cambio, si el archivo ya existía, se elimina y se vuelve a crear.
2. ***set_value()***
   Se comprueba si la clave existe mediante la función *exist()*. Si no existe, se abre el archivo de tuplas y se escribe al final del mismo.

3. ***get_value()***
   Se abre el archivo de tuplas y se lee línea por línea en busca de la clave especificada. Cuando se encuentra, se copian los datos en los parámetros. Si no se encuentra, los parámetros adquieren un valor de 0 o NULL dependiendo su tipo.
4. ***modify_value()***
   Se elimina la tupla especificada mediante *delete_key()*. Tras ello, se inserta la nueva tupla con *set_value()*.
5. ***delete_key()***
   Se comprueba que existe la clave con *exist()*. En caso de que exista, se recorre el archivo copiando cada línea a un archivo temporal *temp_file.txt* a excepción de la línea que incluye la tupla que se desea eliminar. Al terminar de copiar todas las tuplas, se elimina el antiguo archivo de tuplas y el temporal se renombra a *tuplas.txt*.
6. ***exist()***
   Se recorre el archivo en busca de la clave especificada. Si se encuentra, se devuelve 0. En caso contrario se devuelve 1.

La implementación de estas funciones es altamente ineficiente y prioriza la reutilización de las mismas entre sí antes que el rendimiento, pues no es el enfoque principal del ejercicio.

3.- Cliente

Los archivos relacionados con el lado del cliente son:

Archivo *"claves.h*" contiene las definiciones de las funciones y las librerías necesarias. El archivo que implementa las funciones *"claves.c*" en este archivo se rellenan las peticiones y se envían al servidor mediante su cola y se espera hasta que se reciba una respuesta. La estructura base de cada función es:

1. Manejo de errores en los parámetros.
2. Apertura de las colas del cliente y del servidor.
3. Creación de las estructuras de *Request y Response* y cumplimentación de la petición introduciendo el número de operación, la clave, cualquier parámetro relevante para la operación a realizar y el nombre de la cola del cliente.
4. Envío de la petición mediante una función llamada *send_message*. En esta, se trata de enviar la petición a la cola del servidor. Si falla porque la cola del servidor está completa (10 mensajes), se intenta de nuevo tras un tiempo de espera que crece exponencialmente hasta llegar a *MAX_RETRIES*, que por defecto es 5. En ese momento, si la cola del servidor sigue llena, se lanza un error.
5. Recepción de la respuesta.
6. Cierre de las colas y borrado del archivo de cola del cliente.
7. Devolución del código de estado (error o éxito) de la respuesta del servidor.

Para la comprobación de las funciones y de la práctica se han generado varios clientes. Por un lado se ha creado el archivo "*cliente_tests.c*". Este cliente se encarga de probar todas las posibles respuestas de cada función, de forma similar a unos test unitarios. Así se comprueban todos los posibles errores y el funcionamiento correcto de cada función. Los tests no se especificarán en el documento, pues se sobrepasarían las 5 páginas permitidas. Sin embargo, se puede ejecutar el archivo y ver en consola los detalles de cada test. En resumen, se ha probado que la ejecución de cualquier función antes de realizar *init()* resulta en error, que después de realizarlo todas las funciones no dan error y su comportamiento es el esperado (incluyendo la comprobación de los valores de la tupla recuperada con *get_value()*) y que todo ello se mantiene cierto con más de una tupla a la vez.

El otro archivo cliente es "*cliente_concurrente.c*". Este se puede ejecutar las veces que se desee en distintas terminales, ya que funcionará como clientes diferentes. Estos clientes entran en un bucle infinito enviando continuamente peticiones. El objetivo de estos clientes es probar la ejecución concurrente.

Por último, cabe notar que todos los test y pruebas realizados en nuestras computadoras son exitosos. Sin embargo, corriendo el código en la máquina Debian del Servicio Picasso, la creación del archivo *tuplas.txt* falla. Por ello, el servidor no puede realizar ninguna operación y muestra continuamente el error de que no puede abrir el archivo. Creemos que esto se debe a un problema de permisos, por lo que si se encuentra este error, rogamos que nos contacten para demostrar la exitosa ejecución del código.