

Sistemas Distribuidos.

Ejercicio Evaluable 1: colas de mensajes

Se desea diseñar e implementar un servicio distribuido (utilizando colas de mensajes POSIX) que permite almacenar **tuplas <clave-valor1-valor2>**, donde la clave será un número entero (**int**) y los valores asociados a la clave serán:

- Valor1: una **cadena de caracteres** de cómo mucho 255 caracteres (excluido el código 0 que permite indicar el fin de la cadena),
- Valor 2: un **vector** de N elementos de tipo **double** (coma flotante de doble precisión). El valor 2 vendrá dado por dos elementos: un número N que determina la longitud del vector y el vector con los elementos. El valor de N será cualquier número comprendido entre 1 y 32.

Esta aplicación constará de un servidor que almacenará y gestionará las claves y de un API (*Application Programming Interface*) que permitirá a los procesos cliente del sistema hacer uso de los siguientes servicios:

- **int init()**. Esta llamada permite inicializar el servicio de elementos clave-valor1-valor2. Mediante este servicio se destruyen todas las tuplas que estuvieran almacenadas previamente. La función devuelve 0 en caso de éxito y -1 en caso de error.
- **int set_value(int key, char *value1, int N_value2, double *V_value2)**. Este servicio inserta el elemento <key, value1, value2>. El vector correspondiente al valor 2 vendrá dado por la dimensión del vector (N_Value2) y el vector en si (V_value2). El servicio devuelve 0 si se insertó con éxito y -1 en caso de error. Se considera error, intentar insertar una clave que ya existe previamente o que el valor N_value2 esté fuera de rango. En este caso se devolverá -1 y no se insertará. También se considerará error cualquier error en las comunicaciones.
- **int get_value(int key, char *value1, int *N_value2, double *V_value2)**. Este servicio permite obtener los valores asociados a la clave key. La cadena de caracteres asociada se devuelve en value1. En N_Value2 se devuelve la dimensión del vector asociado al valor 2 y en V_value2 las componentes del vector. Tanto value1 como V_value2 tienen que tener espacio reservado para poder almacenar el máximo número de elementos posible (256 en el caso de la cadena de caracteres y 32 en el caso del vector de doubles). La función devuelve 0 en caso de éxito y -1 en caso de error, por ejemplo, si no existe un elemento con dicha clave o si se produce un error de comunicaciones.
- **int modify_value(int key, char *value1, int N_value2, double *V_value2)**. Este servicio permite modificar los valores asociados a la clave key. La función devuelve 0 en caso de éxito y -1 en caso de error, por ejemplo, si no existe un elemento con dicha clave o si se produce un error en las comunicaciones. También se devolverá -1 si el valor N_value2 está fuera de rango.
- **int delete_key(int key)**. Este servicio permite borrar el elemento cuya clave es key. La función devuelve 0 en caso de éxito y -1 en caso de error. En caso de que la clave no exista también se devuelve -1.
- **int exist(int key)**. Este servicio permite determinar si existe un elemento con clave key. La función devuelve 1 en caso de que exista y 0 en caso de que no exista. En caso de error se devuelve -1. Un error puede ocurrir en este caso por un problema en las comunicaciones.

Tenga en cuenta para las llamadas anteriores que también se considera error, por ejemplo, que se produzca un error en el sistema de paso de mensajes (colas inexistentes, errores al enviar datos a una

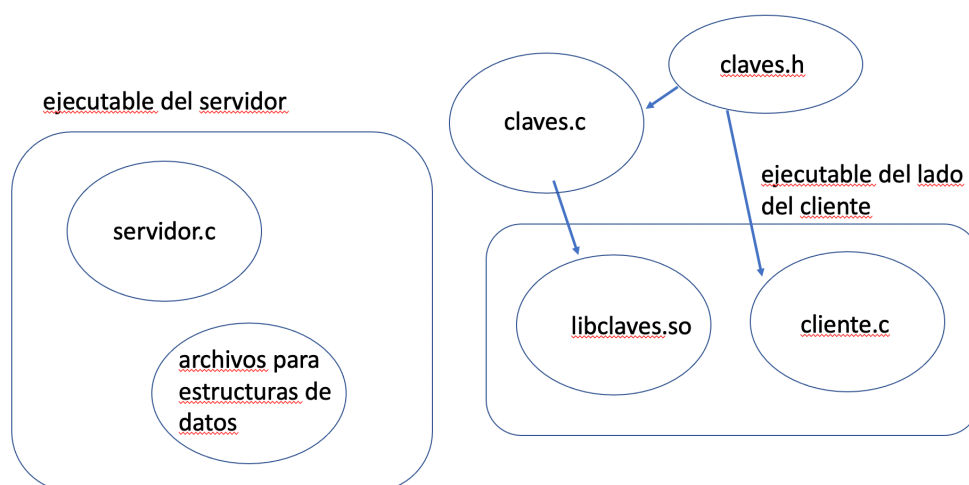
cola, errores al recibir datos de las colas, etc.). También se considerará error y se devolverá -1 sin necesidad de contactar con el servidor, en caso de que a las llamadas anteriores se pasen cadenas de más de 256 caracteres para el valor `value1`.

Todas las operaciones anteriores deben realizarse de forma **atómica** en el servidor.

Diseñe e implemente, utilizando exclusivamente **colas de mensajes POSIX** y el lenguaje de programación C, el sistema que implemente este servicio de elementos clave-valor1-valor2. Para ello debe:

1. Desarrollar el código del servidor (**servidor.c**) encargado de gestionar las estructuras de datos que almacenan los elementos clave-valor1-valor2-valor3. Puede elegirse la estructura de datos que se estime oportuno, siempre que no imponga un límite en el número de elementos que se pueden almacenar. El servidor desarrollado debe ser **concurrente**.
2. Desarrollar el código que implementa los servicios anteriores (**init, set_value, get_value, delete_key, modify_value y exist**). El código se desarrollará sobre el archivo con nombre **claves.c**. Este es el código que ofrece la interfaz a los clientes y se encarga de implementar los servicios anteriores (del lado del cliente) contactando con el servidor anterior. A partir de dicha implementación se deberá crear una biblioteca dinámica denominada **libclaves.so**. Esta será la biblioteca que utilizarán las aplicaciones de usuario que para usar el servicio. Debe investigar y buscar la forma de crear dicha biblioteca.
3. Desarrollar un ejemplo de código de un cliente (**cliente.c**) que utilice las funciones anteriores. El ejecutable de este programa tiene que generarse empleando la biblioteca desarrollada en el apartado anterior, es decir, el código de este cliente debe enlazarse con la biblioteca dinámica anterior. Este cliente se utilizará para probar el servicio desarrollado y deberá realizar las invocaciones al API de tuplas que considere oportuno. El código incluido en **cliente.c** solo podrá incluir llamadas a los servicios implementados y descritos anteriormente. En él no puede haber ninguna referencia a colas de mensajes.
4. Elaborar un plan de pruebas del servicio desarrollado. Este plan se probará con el código desarrollado en el apartado anterior.

La estructura del código a desarrollar se muestra en la siguiente figura:



Material a entregar: Se deberá entregar la siguiente documentación:

Fichero **ejercicio_evaluable1.tar**, que incluirá, al menos:

- El código del cliente (`cliente.c`) donde se encuentran exclusivamente el uso de las llamadas al API que permiten probar el servicio desarrollado.
- El código del servidor (`servidor.c`) y el código correspondiente a la implementación de la biblioteca del lado del cliente (`claves.c`).
- Un fichero `Makefile`, que permite compilar todos los archivos anteriores y generar la biblioteca `libclaves.so`. Este `Makefile` debe generar además dos ejecutables: el ejecutable del servidor, que implementa el servicio y el ejecutable de cliente, obtenido a partir del archivo `cliente.c` y la biblioteca `libclaves.so`.
- Una pequeña memoria en **PDF** (no más de cinco páginas, incluida la portada), indicando el diseño realizado y la forma de compilar y generar el ejecutable del cliente y del servidor.
- Dentro del fichero comprimido `ejercicio_evaluable1.tar`, también se incluirán todos aquellos archivos adicionales que necesite para el desarrollo del servicio. Por ejemplo, ficheros que gestionan las estructuras de datos elegidas, etc.

Para el almacenamiento de los elementos clave-valor1-valor2-valor3 puede hacer uso de la estructura de datos o mecanismo de almacenamiento que considere más adecuado (listas, ficheros, etc), el cual describirá en la memoria entregada. La estructura elegida no debe fijar un límite en el número de elementos que se pueden almacenar.

Se recomienda probar el servidor con varios clientes de forma concurrente.

La entrega se realizará mediante Aula Global en el entregador habilitado. La fecha límite de entrega es: 17/03/2024 (23:55 horas).

Aclaraciones adicionales:

Cuando se esté desarrollando este ejercicio es posible que en algunas ocasiones las colas de mensajes se queden en un estado inconsistente y tanto el cliente como el servidor fallen en tiempo de ejecución. En estos casos lo mejor es borrar las colas de mensajes que se hayan quedado en el sistema. Para ello pueden eliminarse las colas desde el directorio `/dev/mqueue`.