**Sistemas Distribuidos.**

**Grado en Ingeniería Informática. Curso 2019/2020**

**Ejercicio Evaluable 1: colas de mensajes**

Se desea diseñar e implementar un modelo de vector distribuido. Sobre un vector distribuido se definen los siguientes servicios:

- `int` **`init`** `(char *nombre, int N)`. Este servicio permite inicializar un vector distribuido de `N` números enteros. La función devuelve 1 cuando el vector se ha creado por primera vez. En caso de que el vector ya esté creado con el mismo número de componentes, la función devuelve 0. La función devuelve -1 en caso de error, por ejemplo, que se intente crear un vector que existe con un número de componentes distinto.
- `int` **`set`**`(char *nombre, int i, int valor)`. Este servicio inserta el valor en la posición `i` del vector nombre. La función devuelve 0 en caso de éxito y -1 en caso de error, por ejemplo, que se intente insertar un elemento en una posición invalida, o el vector no exista.
- `int` **`get`**`(char *nombre, int i, int *valor)`. Este servicio permite recuperar el valor del elemento `i` del vector nombre. La función devuelve 0 en caso de éxito y -1 en caso de error, por ejemplo, que se intente recuperar un elemento en una posición invalida, o el vector no exista.
- `int` **`destroy`**`(chat *nombre)`. Este servicio permite borrar un vector previamente creado. La función devuelve 1 en caso de éxito y -1 en caso de error.

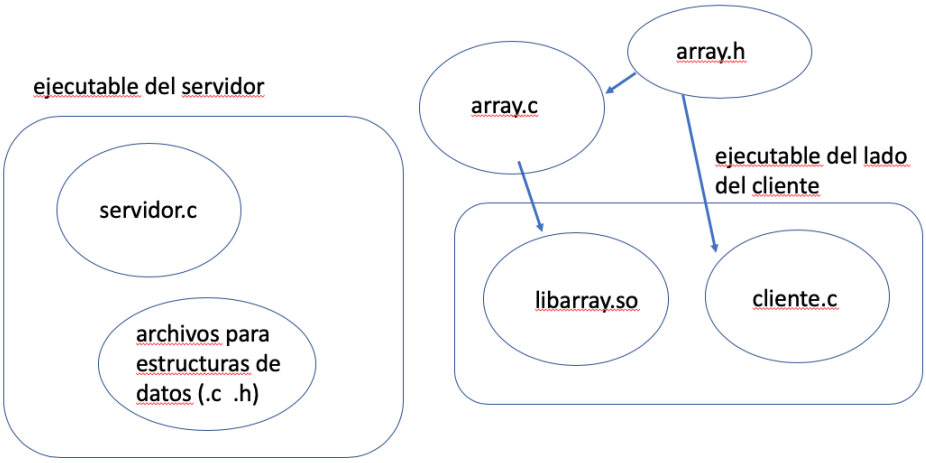
Diseñe e implemente, utilizando **colas de mensajes POSIX** y el lenguaje de programación C, el sistema que implemente este servicio de vectores distribuidos. Para ello debe:

1. Desarrollar el código del servidor (**`servidor.c`**) encargado de gestionar los vectores distribuidos. Puede elegirse la estructura de datos que se estime oportuno para la gestión de los arrays, siempre que **no** se imponga un límite en el número de arrays a gestionar y en el tamaño de cada array. Dado que el servidor desarrollado debe ser concurrente, es importante que se gestione correctamente los accesos concurrentes sobre la estructura de datos utilizada para la gestión de los arrays. Se recomienda que la gestión de las estructuras de datos utilizadas se programe en otro archivo **`.c`** diferente.
2. Desarrollar el código que implementa los servicios anteriores (**`init`**, **`set`**, **`get`** y **`destroy`**). El código se desarrollará sobre el archivo con nombre **`array.c`.** Este es el código que ofrece la interfaz a los clientes y se encarga de implementar los servicios anteriores contactando con el servidor anterior. A partir de dicha implementación se deberá crear una biblioteca **dinámica** denominada `libarray.so`. Esta será la biblioteca que utilizarán las aplicaciones para usar el servicio. Debe investigar y buscar la forma de crear dicha biblioteca.
3. Desarrollar el 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 anterior. Este cliente deberá realizar llamadas como, por ejemplo:

```
init(“vector1”, 100);
init(“vector2”, 200);
set(“vector1”, 0, 40);
set(“vector1”, 120, 30);
init(“vector1”, 200);
destroy(“vector1”);
destroy(“vector”);
```

Comprobando, eso si, los errores que devuelven cada una de las llamadas.

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 el código del cliente, el código del servidor y el código del archivo que implementa los servicios sobre el vector. Además, se debe incluir una pequeña memoria en PDF (no más de tres, cuatro páginas), indicando el diseño realizado y la forma de compilar y generar el ejecutable del cliente y del servidor. El servidor desarrollado debe ser **concurrente**. También debe incluirse un fichero `Makefile`, que permite compilar todos los archivos anteriores y generar la biblioteca `libarray.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 `libarray.so`. Dentro del fichero comprimido `ejercicio_evaluable1.tar`, también se incluirán todos aquellos archivos adicionales que necesite para el desarrollo del servicio.

Para crear un archivo comprimido con formato **`tar`** siga los siguientes pasos. Suponiendo que existe un directorio denominado **`ejercicio_evaluable1`** donde están todos los archivos a entregar, incluido la memoria en formato PDF, debe ejecutarse el siguiente mandato en la consola:

```
tar cvf ejercicio_evaluable1.tar ejercicio_evaluable1
```

Para crear el fichero Makefile puede utilizarse de base y modificarse el archivo que se proporcionó en el ejercicio de laboratorio 1, cuyo material de apoyo está en Aula Global.

**La entrega se realizará mediante Aula Global. La fecha límite de entrega es: 8/03/2020. El Ejercicio puede realizarse en grupos de hasta tres alumnos.**