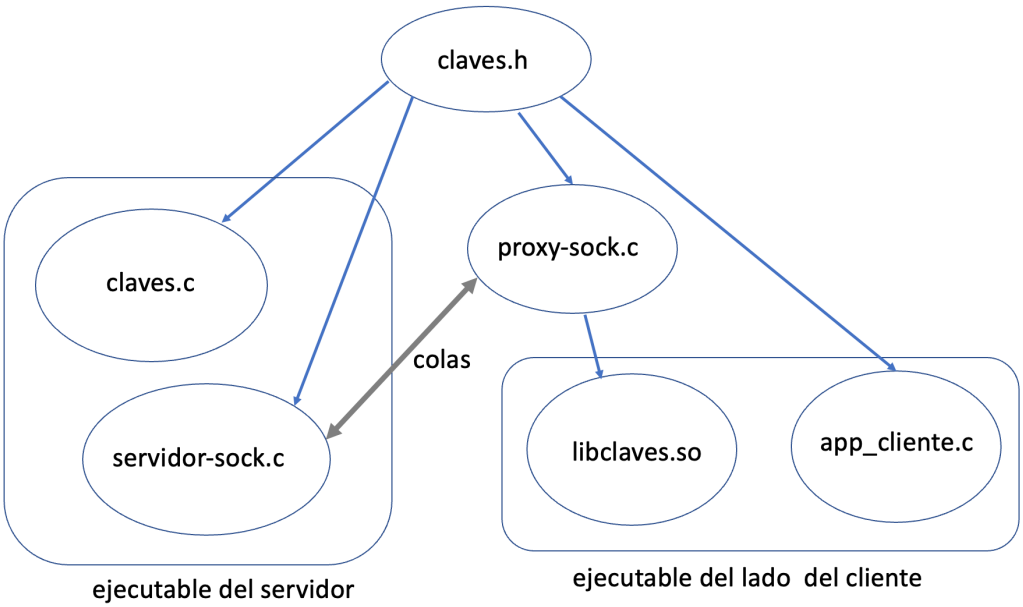
Sistemas Distribuidos.
Ejercicio Evaluable 2: Sockets TCP

Se desea diseñar e implementar el mismo servicio realizado en el ejercicio evaluable 1, pero en este caso utilizando sockets TCP.

En este ejercicio la aplicación tendrá la estructura de la siguiente figura:



En esta aplicación los ficheros `claves.h`, `claves.c` y `app-cliente.c` son los mismos que los implementados en el primer ejercicio. Para este ejercicio solo han de implementarse los archivos `proxy-sock.c` y `servidor-sock.c`, en este caso utilizando sockets TCP y el lenguaje de programación C. Como parte de este ejercicio tendrá que definirse el protocolo de aplicación utilizado entre el proxy y el servidor.

Aclaraciones adicionales:

La memoria debe indicar el diseño realizado y el protocolo de aplicación seguido en la comunicación `proxy-sock.c` y `servidor-sock.c`. Tenga en cuenta que el diseño y el protocolo definido tiene que ser independiente del lenguaje de programación utilizado finalmente. Por tanto, aunque en este ejercicio se utiliza C para el cliente y el servidor, **la solución no debe incluir el envío por un socket de estructuras de C ni cualquier otro elemento dependiente de este lenguaje de programación**, ya que esto haría que la solución no fuera independiente.

En este ejercicio la biblioteca a desarrollar tiene que conocer la dirección IP y el puerto del servidor que ofrece el servicio de tuplas. Para evitar tener una dirección físicamente programada en el código y no tener que pasar la IP y el puerto en la línea de mandatos del programa cliente, se va a considerar que tanto la dirección IP como el puerto se van a pasar al cliente utilizando dos variables de entorno:

- `IP_TUPLAS`: variable de entorno que define la IP del servidor.
- `PORT_TUPLAS`: variable de entorno que define el puerto del servidor de tuplas.

Estas variables de entorno habrán de definirse en cada terminal donde se ejecute el cliente. Para definir una variable de entorno denominada VAR1 con valor "hola" habrá que especificar en la consola:

```
$> export VAR1=hola
```

A continuación, se muestra un ejemplo de programa que es capaz de acceder al valor de dicha variable de entorno. Tenga en cuenta que la variable de entorno se devuelve como una cadena de caracteres.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
        char *var;

        var = getenv("VAR1");
        if (var == NULL){
                printf("Variable VAR1 no definida\n");
                return 0;
        }
        else
                printf("Variable VAR1 definida con valor %s\n", var);

        return 0;
}
```

Para conocer la lista de variables de entorno definidas es necesario ejecutar en la consola:

```
$> env
```

Para poder ejecutar el cliente será necesario definir las variables de entorno IP_TUPLAS y PORT_TUPLAS utilizando el comando export indicado anteriormente. Una forma alternativa de pasar los valores de estas dos variables de entorno al programa cliente es ejecutando este programa de la siguiente forma:

```
$> env IP_TUPLAS=localhost PORT_TUPLAS=8080 ./cliente
```

En cuanto al servidor, el puerto se le pasará como un argumento en la línea de mandatos de la siguiente forma:

```
./servidor <PUERTO>
```

Por ejemplo:

```
./servidor 4500
```

Especificará que el servidor aceptará peticiones de los clientes en el puerto 4500.

Material a entregar

Se deberá entregar un fichero **ejercicio_evaluable2.zip**, que incluirá, al menos:

- El código del cliente (app-cliente.c) donde se encuentran exclusivamente el uso de las llamadas a la API que permiten probar el servicio desarrollado. En caso de que se entreguen varios programas clientes serán app-cliente-1.c, app-cliente-2.c, y así sucesivamente.

- El código del servidor (`servidor-sock.c`) y el código correspondiente a la implementación de la biblioteca del lado del cliente (`proxy-sock.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 (o clientes si se entregan varios), obtenido a partir del archivo app-`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. **Debe incluir el diseño del protocolo de aplicación utilizando entre el proxy y el servidor.**
- Dentro del fichero comprimido `ejercicio_evaluable2.zip` también se incluirán todos aquellos archivos adicionales (.c o .h) que necesite para el desarrollo del servicio (como ficheros que gestionan las estructuras de datos elegidas, etc.). Por ejemplo, el archivo `claves.c` podría hacer uso de otros archivos adicionales que implementaran distintos aspectos de la aplicación.

Para el almacenamiento de los elementos `key-value1-value2-value3` 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.

Tenga en cuenta que el prototipo de las funciones (claves.h) `destroy`, `set_value`, `get_value`, `delete_key`, `modify_value` y `exist` no puede ser modificado.

Todo el código desarrollado en el ejercicio 1 sigue siendo válido en este ejercicio a excepción de los archivos `proxy-sock.c` y `servidor-sock.c`, que son los únicos que hay que implementar en este.

La entrega se realizará mediante Aula Global en el entregador habilitado. La fecha límite de entrega es: 06/04/2025 (23:55 horas).