Servidor de ficheros utilizando Web Services

1.- Descripción del servicio

En este ejercicio se va a implementar un servidor de ficheros sencillo utilizando **Web Services con gSoap**. Básicamente, el servidor contendrá **4 servicios** con las siguientes funcionalidades: obtener el tamaño de un fichero, crear un fichero, leer datos de un fichero y escribir datos en un fichero. El objetivo consiste en que la parte cliente pueda realizar copias de ficheros, tanto desde el cliente al servidor, como desde el servidor al cliente, invocando estos procedimientos de forma remota.

El siguiente fichero, llamado `wsFileServer.h`, contiene la definición del servicio con los tipos de datos y dos de los servicios del servidor.

```
//gsoap wsFilens service name: wsFileServer
//gsoap wsFilens service style: rpc
//gsoap wsFilens service encoding: encoded
//gsoap wsFilens service namespace: urn:wsFilens

/** Length of data buffer */
#define MAX_DATA_SIZE 4094

/** Length for tString */
#define NAME_LENGTH 128

/** Typedef for the file name */
typedef char *xsd__string;

/** Struct that contains a file name */
typedef struct tFileName{
        int __size;
        xsd__string __ptr;
}wsFilens__tFileName;

/** Binary data */
typedef struct tData{
        int __size;
        unsigned char *__ptr;
}wsFilens__tData;

int wsFilens__getFileSize (wsFilens__tFileName fileName, int *result);
int wsFilens__createFile (wsFilens__tFileName fileName, int *result);
```

En este fichero se definen dos constantes, `MAX_DATA_SIZE` y `STRING_LENGTH`, las cuales indican la longitud máxima del buffer empleado para leer/escribir datos y la longitud máxima de los nombres de los ficheros, respectivamente. El tipo `tString` se utiliza para representar el nombre de los ficheros. La estructura `wsFilens__tFileName` se emplea para representar el nombre de un fichero. Es importante remarcar que, para poder enviar arrays dinámicos a través de la red, es necesario contar con el campo `__size`, el cual representa el número de elementos a enviar dentro del array. En este caso, el campo `__ptr` hace referencia al puntero que contiene los datos. Esta estructura se emplea como parámetro de entrada en los dos procedimientos remotos, `getFileSize` y `createFile`. El primer procedimiento devuelve el tamaño del fichero indicado en el parámetro `fileName`, mientras que el segundo crea un fichero con el nombre indicado en el parámetro `fileName`. En ambos procedimientos, si ocurre algún error, se devuelve el valor -1. A diferencia de los RPCs, el valor devuelto se indica siempre en el último parámetro. Para estos dos servicios, ese parámetro es `result`.

Además de estos servicios, **se deberán crear dos servicios adicionales**: uno para leer datos de un fichero y otro para escribir datos en un fichero. Para desarrollar estos

servicios, de forma similar a como se hizo con los dos anteriores, será necesario definir los parámetros de entrada y salida.

Es importante remarcar que en los Web Services de gSoap, el valor devuelto de cada servicio hace referencia al estado de la ejecución del mismo. Por ello, se indicará con `SOAP_OK` la ejecución exitosa.

Para representar un buffer se utilizará la estructura `wsFilens__tData`, donde `__data` representa un array dinámico de tipo `unsigned char` y `__size` indica el número de caracteres almacenados en el array `__data`.

Para la lectura de datos, en los parámetros de entrada se debe indicar el nombre del fichero, el offset y la cantidad de bytes a leer. Como salida, obtendremos una estructura `wsFilens__tData`. De forma similar, para la escritura de datos tendremos como parámetros de entrada el nombre del fichero, el offset y una estructura `wsFilens__tData` que contiene los datos a escribir. Como salida, se devolverá el valor 0 si no ha ocurrido ningún error y -1 en caso contrario.

Con la información descrita anteriormente, se pide:

a) Definir los servicios remotos para leer y escribir datos.
b) Implementar los servicios remotos en el servidor.
c) Implementar un cliente que realice las siguientes operaciones.
    a. Envíe ficheros desde el cliente al servidor.
    b. Copie ficheros desde el servidor al cliente.

Para desarrollar este ejercicio se proporciona parte del código en los siguientes ficheros:

- `wsFileServer.h`, contiene constantes, tipos de datos y la definición de los servicios pedidos.
- `client.c`, contiene la implementación de las funciones pedidas.
- `Server.c`, contiene el código del servidor, incluyendo la implementación de los servicios y de las funciones auxiliares.

2.- Implementación del servidor

En el servidor se deberán incluir los servicios anteriormente descritos.

La operaciones que debe realizar el servicio para leer datos de un fichero se describen a continuación:

1. Abrir fichero para lectura.
2. Posicionarse en el fichero utilizando `lseek`.
3. Calcular tamaño del fichero.
4. Reservar memoria en la estructura `wsFilens__tData`.
5. Mientras no se hayan leído todos los datos pedidos:
   5.1. Leer datos (máximo `MAX_DATA_SIZE` bytes)
6. Cerrar fichero.
7. Devolver resultado.

Si se produce cualquier error, por ejemplo al abrir el fichero, se devolverá la estructura `t_data` con el valor -1 en el campo `__size`.

De forma similar, para escribir datos en un fichero se deberán realizar las siguientes operaciones:

1. Abrir fichero para escritura.
2. Posicionarse en el fichero utilizando `lseek`.
3. Mientras no se hayan escrito todos los datos:
   3.1. Escribir datos (máximo `MAX_DATA_SIZE` bytes)
4. Cerrar fichero.
5. Devolver resultado.

Si se produce cualquier error, por ejemplo al escribir datos, se devuelve el valor -1.

3.- Implementación del cliente.

El programa cliente debe poder realizar dos operaciones:

a) Copiar un fichero, ubicado en la parte cliente, en el servidor (enviar fichero).
b) Copiar un fichero, ubicado en la parte servidor, en el cliente (recibir fichero).

La ejecución del programa cliente se realizará de la siguiente forma:

```
./client serverIP:port sendFile|receiveFile sourceFile destinationFile
```

donde `serverIP` es la dirección IP donde se ejecuta el servidor, `port` es el puerto donde escucha peticiones el servidor, `sendFile` es la opción que indica que se realizará la copia desde el cliente al servidor, `receiveFile` es la opción que indica que se realizará la copia desde el servidor al cliente, `sourceFile` es el nombre del fichero origen y `destinationFile` es el nombre del fichero destino.

En la parte cliente, para enviar un fichero al servidor se deberán realizar las siguientes operaciones:

1. Reserva memoria en la estructura `wsFilens__tFileName`.
   a. Copia el nombre del fichero destino en esta estructura.
2. Crear fichero destino en el servidor.
3. Obtener el tamaño del fichero origen en el cliente.
4. Reserva memoria en la estructura `wsFilens__tData`.
5. Mientras no se hayan enviado todos los bytes del fichero origen.
   a. Leer datos del fichero origen (cliente).
   b. Escribir datos en el fichero destino (servidor).
6. Cerrar fichero origen.

En la parte cliente, para recibir un fichero del servidor se deberán realizar las siguientes operaciones:

1. Crear fichero destino en el cliente.
2. Reserva memoria en la estructura `wsFilens__tFileName`.
   a. Copia el nombre del fichero origen en esta estructura.
3. Obtener el tamaño del fichero origen en el servidor.
4. Mientras no se hayan recibido todos los bytes del fichero origen.
   a. Leer datos del fichero origen (servidor).
   b. Escribir datos en el fichero destino (cliente).
5. Cerrar fichero destino.