Departamento de Ingeniería Telemática
Escuela Superior de Ingenieros
Universidad de Sevilla

PRÁCTICA 2.
RPC. Servidor con y sin estado.

1. OBJETIVO

El objetivo es distinguir entre un servidor con estado y un servidor sin estado mediante la creación de un servidor de ficheros simple.

2. DESCRIPCIÓN

Un sistema de ficheros distribuido permite acceder ficheros remotos de la misma forma que se acceden los ficheros locales.

El fichero fscon.x dado a continuación contiene la descripción de un servidor de ficheros simple que permite de forma remota leer/escribir en un fichero, abrir y cerrar un fichero utilizando RPCs.

```
const BUF_SIZE=1024;
const MAX_PATH=256;

typedef opaque BUF<BUF_SIZE>;

struct LEER_result {
  int cod_error;
  BUF datos;
  };

program FSCON {
   version CON {

      LEER_result leer (int fd, int nbytes)= 1;


      int escribir (int fd, int nbytes, BUF buf)= 2;


      int abrir (string nom<MAX_PATH>)= 3;


      int cerrar (int fd)= 4;


   } = 1;
} = 999999991;
```

fscon.x

⇒ 1. Estudie detenidamente la descripción de este servidor. Esta especificación es una especificación sencilla, que se podría completar con otras operaciones sobre ficheros, como por ejemplo crear o borrar, obtener información acerca del fichero, etc.

⇒ 2. Arranque el ordenador con Ubuntu. Abra un terminal y cree un directorio cuyo nombre sea su login en la plataforma de enseñanza virtual (este directorio es el que se entregará en la tarea correspondiente una vez finalizada la práctica). Vaya a este directorio y siga con los siguientes pasos.

⇒ 3. Descargue el fichero fsRPC.zip de la plataforma de enseñanza virtual. Al descomprimir este fichero se obtendrá un directorio fsRPC con dos directorios fscon y fssin. En el directorio fscon encontrará la especificación del servidor de ficheros, parte de la implementación del servidor y un pequeño ejemplo de cliente.

⇒ 4. Genere con rpcgen –N –C –a fscon.x todos los ficheros necesarios.

⇒ 5. Comprenda y compile el servidor:

```
gcc –W –Wall fscon_svc.c fscon_servidor.c fscon_xdr.c –o servidorfscon
```

⇒ 6. Comprenda y compile el cliente:

```
gcc –W –Wall fscon_clnt.c fscon_cliente.c fscon_xdr.c –o clientefscon
```

⇒ 7. Cree un fichero: echo texto > fichero

⇒ 8. Ejecute el servidor

⇒ 9. Ejecute el cliente: clientefscon localhost fichero

⇒ 10. Usando de base fscon_cliente.c cree el fichero fscon_escribe.c que utilice el servidor para escribir un texto en un fichero. Por simplicidad, utilice una variable dentro del programa que contenga el texto a escribir en el fichero. Los parámetros que se deben dar al programa son, en este orden, la máquina del servidor y el fichero a escribir. El servidor usa la función write para escribir en el fichero. Compile y pruebe.

Este servidor se dice que es un **servidor con estado**. Un servidor con estado es aquel que guarda algún tipo de información acerca de las operaciones que realizan los clientes. En este caso, cuando un cliente abre un fichero en el servidor, el servidor de ficheros mantiene el fichero abierto hasta que el cliente lo cierra. El servidor devuelve el descriptor de fichero para el fichero abierto por el cliente. En llamadas posteriores a leer o escribir en el fichero, el cliente debe proporcionar este descriptor al servidor para poder realizar la lectura o escritura.

En este caso, si el cliente no tuviera respuesta para una escritura transcurrido cierto tiempo, el cliente podría intentar repetir la solicitud. Pero el cliente no sabe exactamente qué es lo que ha ocurrido con su petición. Podría haber ocurrido una de las tres situaciones siguientes:

1. La solicitud se perdió.
2. El servidor recibió la solicitud, la atendió, pero se ha perdido la respuesta.
3. El servidor no recibe peticiones.

En caso de que lo que haya ocurrido sea que se ha perdido la respuesta del servidor, cuando el cliente vuelva a intentar la escritura, se obtendrá un resultado diferente. Esto es debido a que el servidor en la primera escritura adelanta el puntero de escritura al fichero y al volver a recibir la petición de escritura, la escritura se vuelve a producir a continuación de la que ya se había realizado.

Este tipo de operaciones se dice que no son **idempotentes** ya que al repetir la operación el resultado final no es el mismo. Las operaciones que son idempotentes, obtienen el mismo resultado en caso de repetir la operación.

Para lograr operaciones idempotentes se realiza otra especificación del servidor de la siguiente forma:

```
const BUF_SIZE=1024;
const MAX_PATH=256;

typedef opaque BUF<BUF_SIZE>;

struct LEER_result {
  int cod_error;
  BUF datos;
  };

program FSSIN {
   version SIN {

      LEER_result leer (string nom<MAX_PATH>, int offs, int nbytes)= 1;


      int escribir (string nom<MAX_PATH>, int offs, int nbytes, BUF buf)= 2;


   } = 2;
} = 999999992;
```

fssin.x

Este servidor será un **servidor sin estado** ya que no guardará información de ningún tipo acerca de las operaciones que realizan los clientes. En este caso, para cada operación de lectura o escritura de los clientes, el servidor debe abrir el fichero correspondiente, posicionarse en el lugar adecuado (con lseek), luego leer o escribir y por último cerrar el fichero. Observe que para la operación de lectura o escritura hay que dar el nombre del fichero y también el offset dentro del fichero.

⇒ 11. Implemente el servidor. Para la lectura y la escritura, el servidor debe, antes de leer o escribir en el fichero, abrirlo y posicionarse en el lugar adecuado del fichero. Una vez realizada la operación de lectura o escritura, el servidor debe cerrar el fichero. La función para posicionarse en el fichero es lseek (hay que indicar, el descriptor del fichero, el offset y a continuación SEEK_SET para indicar que es a partir del comienzo del fichero).

⇒ 12. Implemente un cliente en el fichero fssin_cliente.c que escriba la frase "ESTO ES UNA PRUEBA" en el fichero de nombre fsin (que previamente debe estar creado).

⇒ 13. Pruebe a ejecutar varias veces la misma escritura en el cliente. Podrá comprobar que el resultado final de la ejecución es la misma.

⇒ 14. En la tarea habilitada para la práctica, entregue un fichero comprimido (.zip) con nombre su login que sea el directorio de nombre su login comprimido (que contiene el directorio fsRPC con todo su trabajo).