1. Introducción

El objetivo de este laboratorio es familiarizar al alumno/a con las llamadas a procedimientos remotos que ofrece ONC-RPC. Para ello se proponen los dos siguientes ejercicios:

Ejercicio 1. Implemente un servicio `ObtenerTiempo` que permita obtener los siguientes datos de la hora y fecha actual de un servidor: hora, minutos, segundos, día, mes y año. Para ello se deberá definir el archivo `obtenerTiempo.x` con la interfaz que permita acceder a esta información. Los datos anteriores se devolverán como un entero y el servidor podrá acceder a ellos utilizando la siguiente función:

```
#include <time.h>

struct tm *localtime(const time_t *timer);
```

La estructura `tm` tiene los siguientes campos:

```
struct tm {
   int tm_sec;         /* seconds,  range 0 to 59          */
   int tm_min;         /* minutes, range 0 to 59           */
   int tm_hour;        /* hours, range 0 to 23             */
   int tm_mday;        /* day of the month, range 1 to 31  */
   int tm_mon;         /* month, range 0 to 11             */
   int tm_year;        /* The number of years since 1900   */
   int tm_wday;        /* day of the week, range 0 to 6    */
   int tm_yday;        /* day in the year, range 0 to 365  */
   int tm_isdst;       /* daylight saving time             */
};
```

El siguiente programa muestra un ejemplo de uso:

```
#include <stdio.h>
#include <time.h>
int main ()
{
   time_t rawtime;
   struct tm *info;
   time( &rawtime );
   info = localtime( &rawtime );
   printf("Seconds: %d", info.tm_sec);
   return(0);
}
```

Ejercicio 2. El objetivo de este segundo ejercicio es desarrollar, utilizando RPC, un cliente y un servidor con la misma funcionalidad del servidor desarrollado en el laboratorio 3 de sockets. La funcionalidad del cliente es la siguiente:

1. El cliente recibe como argumento en la línea de mandatos la dirección del servidor. Para probarlo con un servidor RPC previamente arrancado, puede invocar al cliente de esta forma:

   `./cliente  localhost`

2. A continuación, el cliente ejecuta un bucle infinito. En cada iteración lee de la entrada estándar una cadena y ejecuta un procedimiento denominado `Eco`. Esta función permite enviar la cadena al servidor. El servidor responderá con la misma cadena, pero en este caso con todos los caracteres en mayúscula. Cuando el usuario teclee "EXIT", el cliente finalizará su ejecución. En este caso, no enviará la cadena "EXIT" al servidor. La función `Eco` por tanto, acepta una cadena como entrada y devuelve una cadena como salida.

Para desarrollar este ejercicio en primer lugar tendrá que definir la interfaz `.x` del servicio que permita obtener la funcionalidad deseada y a continuación compilar esta interfaz con `rpcgen` para obtener los *stubs* del cliente y de servidor. Por último, habrá de implementarse el servidor y el cliente con la funcionalidad deseada.