UNIVERSIDAD DE MÁLAGA

Programación de Sistemas y Concurrencia

Dpto. de Lenguajes y Ciencias de la Computación

Control Junio 2018

APELLIDOS_____NOMBRE_____

DNI _____ ORDENADOR _____GRUPO /TITULACIÓN_____

Bloque Programación C

Vamos a diseñar un esquema de cifrado muy sencillo, consistente en una secuencia alternada de operaciones, denominadas respectivamente **SumaBox** y **XORBox**:

- Una **SumaBox** es una operación que toma un valor de entrada tipo **unsigned char**, y devuelve el resultado de su suma con un determinado número, sin importar si se produce *overflow*, es decir, si el resultado es aritméticamente incorrecto (por ejemplo, al sumar 255 + 1 cuando un **unsigned char** tiene 8 bits). En los ordenadores que vamos a usar, esto se puede conseguir simplemente con el operador de suma del lenguaje C.
- Una **XORBox** es una operación que toma un valor de entrada tipo **unsigned char**, y devuelve como resultado el valor donde exactamente uno de los bits ha cambiado de valor (si era 0 cambia a 1, y si era 1 cambia a 0). Por ejemplo, si para el valor 0x0F cambiamos el valor del último bit (posición 7), obtendremos como resultado el valor 0x8F. Sólo vamos a considerar dos posibles XORBoxes: una cambiará el valor del primer bit (posición 0), y la otra cambiará el valor del último bit (posición 7).

Importante: vamos a considerar que el tipo numérico **unsigned char** tiene 8 bits, numerados de 0 a 7, por lo que puede tomar los valores de 0x00 a 0xFF.

Vamos a representar un esquema de cifrado como una lista de nodos en memoria dinámica, donde:

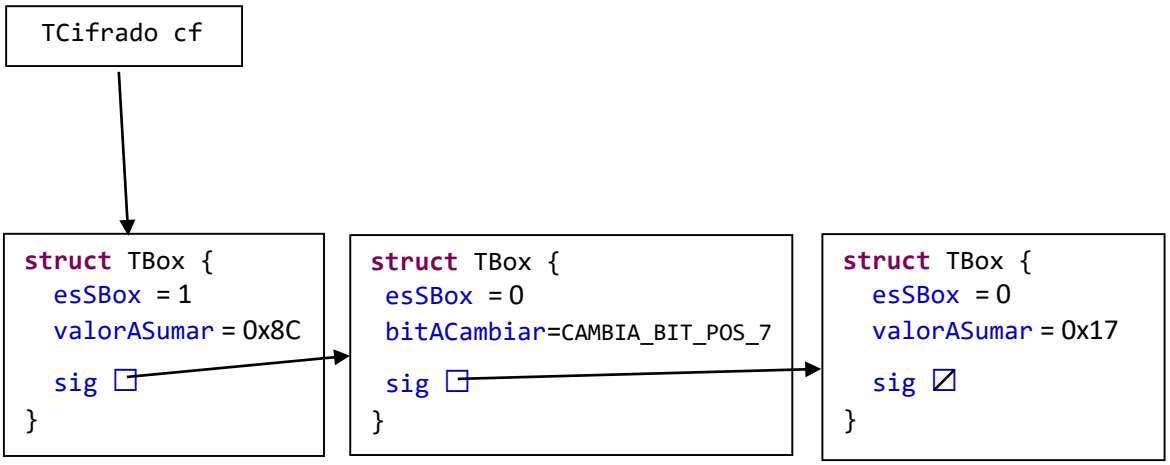
- El primer nodo siempre es de tipo SumaBox.
- El sucesor de un nodo de tipo SumaBox siempre será un nodo de tipo XORBox, y viceversa.
- Puede haber cualquier cantidad de nodos (incluso ningún nodo, es decir, 0).

Lo cual queda plasmado en los siguientes tipos:

```
typedef struct TBox *TCifrado;

#define CAMBIA_BIT_POS_0 0
#define CAMBIA_BIT_POS_7 1
struct TBox {
  unsigned char esSBox; //verdadero si es una SumaBox, falso si es una XORBox
  unsigned char bitACambiar; //valor para indicar si en una XORBox se
                             //cambia el valor del primer o del último bit,
                             //segun las constantes indicadas arriba
  unsigned char valorASumar; //en una SumaBox: valor a sumar
  struct TBox *sig;
};
```

Vemos a continuación un ejemplo gráfico con un esquema de cifrado de 3 nodos:



La operación de aplicar una esquema de cifrado a un determinado valor X será el resultado de ir aplicando cada una de las operaciones que contiene, en orden, de forma que el valor de entrada para el primer nodo es X, y el valor de entrada para los demás nodos es el resultado del nodo anterior. El resultado de la aplicación será el resultado del último nodo. Si el esquema de cifrado tiene 0 nodos, el resultado será el mismo valor inicial X.

Por ejemplo, si aplicamos el esquema de cifrado que hemos representado antes gráficamente al valor 0x64, obtenemos como resultado final el valor 0x87:

Valor de entrada	Operación	Resultado
0x64(==100)	SUMAR 0x8C(==140)	0xF0(==240)
0xF0(==240)	CAMBIA VALOR DE BIT EN POSICION 7	0x70(==112)
0x70(==112)	SUMAR 0x17(==23)	0x87(==135)

El ejercicio consiste en implementar un módulo en C con las siguientes operaciones:

```
/* (0.5 puntos) función necesaria para crear un esquema de cifrado vacío, sin nodos*/
void crearEsquemaDeCifrado (TCifrado *cf);

/* (3 puntos) función que pone un nodo al final de un esquema de cifrado, si es posible. Se debe devolver en el último parámetro un valor lógico que sea verdadero si ha sido posible realizar la operación. No se debe suponer que el valor de box.sig es válido.*/
void insertarBox (TCifrado * cf, struct TBox box, unsigned char *ok);

/* (1.5 puntos) función que dado un nodo y un valor, devuelve el resultado de aplicar dicho nodo a dicho valor. Deberás de tener en cuenta si el nodo es una SumaBox o una XORBox. En el último caso, necesitarás usar operadores lógicos a nivel de bit, como &, |, ^ o bien ~, así como probablemente usar constantes numéricas. */
unsigned char aplicarBox (struct TBox box, unsigned char valor);

/* (1.5 puntos) función que toma un esquema de cifrado y un valor, y devuelve el resultado de aplicar dicho esquema de cifrado a dicho valor, según el método descrito anteriormente.*/
unsigned char aplicarEsquemaDeCifrado(TCifrado cf, unsigned char valor);
```

```
/* (2.5 puntos) función que toma un nombre de fichero, en el que se escribirán
en modo binario los datos correspondientes al esquema de cifrado que se pasa como
parámetro, de modo que el al final el fichero únicamente contenga dichos datos.
Si no se puede abrir el fichero, se debe de mostrar un mensaje de error por
pantalla.*/
void escribirAFichero(char *nm, TCifrado cf);

/* (1.0 puntos) función que destruye un esquema de cifrado y libera la memoria
que ocupa*/
void destruirEsquemaDeCifrado (TCifrado *cf);
```

INSTRUCCIONES: descarga el fichero cifrado_esqueleto.zip del campus virtual. En Eclipse, crea un nuevo proyecto de C que se llame *Cifrado*, y copia el contenido del zip en dicho proyecto.

Anexo. Los prototipos de las funciones de lectura y escritura binaria en ficheros de la biblioteca <stdio.h> son los siguientes (se dan por conocidos los prototipos de las funciones de <stdlib.h> que necesites, como `free` o `malloc`):

`FILE *fopen(const char *path, const char *mode);`
Abre el fichero especificado en el modo indicado ("rb" para lectura binaria y "wb" para escritura binaria). Devuelve un puntero al manejador del fichero en caso de éxito y NULL en caso de error.

`unsigned fread(void *ptr, unsigned size, unsigned nmemb, FILE *stream);`
Lee `nmemb` elementos de datos, cada uno de tamaño `size`, desde el fichero `stream`, y los almacena en la dirección apuntada por ptr. Devuelve el número de elementos leídos.

`unsigned fwrite(const void *ptr, unsigned size, unsigned nmemb, FILE *stream);`
Escribe `nmemb` elementos de datos, cada uno de tamaño `size`, al fichero `stream`, obteniéndolos desde la dirección apuntada por `ptr`. Devuelve el número de elementos escritos.

`int fclose(FILE *fp);`
Guarda el contenido del buffer y cierra el fichero especificado. Devuelve 0 en caso de éxito y -1 en caso de error.

Bloque Concurrencia

Ejercicio Semáforos binarios

Supón que átomos de hidrógeno y oxígeno están dando vueltas en el espacio, intentando agruparse para formar moléculas de agua. Para ello es necesario que dos átomos de hidrógeno y uno de oxígeno se sincronicen. Supongamos que cada átomo de hidrógeno y oxígeno está simulado por una hebra. La gestión de la sincronización de los átomos tiene lugar en un objeto gestor de la clase GestorAgua. Cada átomo de hidrógeno llama al método hListo cuando quiere formar parte de una molécula. Del mismo modo los átomos de oxígeno llaman a oListo cuando quieren combinarse con otros dos hidrógenos para formar agua. Las hebras deben esperar en estos métodos hasta que sea posible formar la molécula. Implementa este sistema utilizando semáforos binarios para sincronizar las hebras.

Ejercicio Monitores (Métodos sincronizados/Locks)

Un grupo de NUM_ESTUDIANTES están estudiando para un examen. Los **estudiantes** sólo pueden estudiar mientras están comiendo pizza que se encuentra en la **mesa** de estudio. Cuando un **estudiante** quiere un trozo de pizza lo coge, si la **mesa** no está vacía. El **primer estudiante** que quiere comer pizza y se encuentra la mesa vacía llama al **pizzero**, y espera que traiga una nueva pizza. Cuando el pizzero llega, el estudiante que le ha llamado le paga y se come un trozo. Si un **estudiante** quiere un trozo de pizza y la mesa está vacía, pero otro estudiante ha llamado ya al pizzero, espera hasta que haya un trozo de pizza para él (come después del estudiante que ha llamado al **pizzero**). Por otro lado, el **pizzero** está siempre esperando un nuevo pedido, cuando lo llaman, prepara la pizza, la lleva al domicilio de los estudiantes, la entrega, espera a que le paguen y se vuelve a la pizzería a esperar otro pedido. Implementa este sistema utilizando métodos sincronizados o locks.