Práctica 3.
Programación de sistemas y concurrencia

Un algoritmo de encriptado permite modificar el contenido de un fichero de forma que sea difícil se entender, y permite, sólo a quien conozca los parámetros y el algoritmo, desencriptar y ver el contenido original del fichero.

En esta práctica, vamos a usar el un algoritmo de que opera con bloques de 64 bits (8 bytes) y una clave de 128 bit para desencriptar ficheros. En concreto, vamos a realizar un programa en C que:

1. cargue en memoria dinámica el contenido de un **fichero encriptado** (little-endian).
2. realice su desencriptado siguiendo el procedimiento que se va a describir al final.
3. lo almacene en un fichero de salida (recordar liberar la memoria dinámica al final).

Tanto el nombre del fichero de entrada como el del fichero de salida son indicados como argumentos del programa en la línea de comandos.

Como el tamaño del fichero original no tiene por qué ser múltiplo de 8 y el algoritmo de desencriptado trabaja con bloques de 8 bytes (64 bits) hay que tener en cuenta las siguientes indicaciones:

- Al comienzo del fichero encriptado se encuentra almacenado (`unsigned`) el tamaño del fichero original desencriptado **en bytes** (*pista: sizeof(char)*, 1 byte).
- Al hacer el **encriptado**, si el tamaño del fichero original no era múltiplo de 8, se tiene al final un **bloque incompleto** para aplicar el cifrado, por lo que artificialmente se completa hasta tener 8 bytes. Estos valores de relleno están almacenados también en el fichero encriptado, y deben también ser desencriptados, pero **no deben escribirse en el fichero de salida.**

Por ejemplo, si el fichero original ocupa 30 bytes, al hacer el encriptado se tuvieron que utilizar 32 bytes, esto es, se pusieron 2 valores de relleno. La longitud total del fichero encriptado es 4 (almacenamiento de la longitud del fichero encriptado) + 30 + 2 (posiciones de relleno), esto es 36 bytes. Al hacer el desencriptado, se hará de 32 bytes, pero sólo se escriben 30 en el fichero de salida.
Para realizar el desencriptado se recomienda definir una función con la siguiente cabecera:

```
void decrypt(unsigned * v, unsigned * k);
```

donde `v` es el array de 2 `unsigned` que se va a desencriptar y k es la clave consistente en un array de 4 `unsigned` con los siguientes valores: `{128, 129, 130, 131}`.
Funciones de C que pueden resultar útiles para la práctica (no es necesario utilizar todas): `fopen`, `fread`, `fwrite`, `fclose`, `malloc`, `free` y `memcpy`.

Pasos para desencriptar. Para cada bloque de 64 bits (`unsigned v[2]`), sea la clave *k* (`unsigned k[4]`), y *delta* una constante igual a 0x9e3779b9:

Inicializar *sum* a 0xC6EF3720

Repetir 32 veces:

Restar a `v[1]` la aplicación del operador XOR (^) a:
(`v[0]` desplazado a la izquierda 4 bits + `k[2]`)
(`v[0]` + `sum`)
(`v[0]` desplazado a la derecha 5 bits + `k[3]`)

Restar a `v[0]` la aplicación del operador XOR (^) a:
(`v[1]` desplazado a la izquierda 4 bits + `k[0]`
(`v[1]`+ `sum`)
(`v[1]` desplazado a la derecha 5 bits + `k[1]`)

Restar a `sum` el valor de `delta`.

Al final de las 32 iteraciones, **se tendrá en `v` el valor desencriptado de los 64 bits**. Recuerda que un array puede ser visto como un puntero al primer elemento.