# Systems Programming and Concurrency

# Lab nº2. Files and Bits Management

A block cipher algorithm Works with blocks of 64 bits and a key of 128 bits following the next procedure (assuming our architecture uses 4 bytes to store a long value; if not, use the corresponding 4 bytes primitive type):

For each ciphered block of 64 bits (`unsigned long v[2]`), let's say *k* is the key (an array of `unsigned long` with 4 items and *delta* is the constant 0x9e3779b9:

Initialize the variable *sum* with the value 0xC6EF3720
Perform 32 times the next:

Subtract to `v[1]`: (`v[0]` left shifted 4 bits + `k[2]`)
XOR
(`v[0]` + `sum`)
XOR
((`v[0]` right shifted 5 bits) + `k[3]`)

Subtract to `v[0]`: (`v[1]` left shifted 4 bits + `k[0]`)
XOR
(`v[1]` + `sum`)
XOR
((`v[1]` right shifted 5 bits) + `k[1]`)

Subtract to `sum` the value of `delta`.

Once finished the 32 iterations, `v` contains the deciphered value of 64 bits.

Develop a C project that loads in dynamic memory the content of a ciphered file, deciphers its content following the above procedure and stores the result into an output file (remind to free the memory). The names of both files are arguments of the program indicated in the command line.

The length of the input file may not be multiple of 8, but this deciphering algorithm works with block of 8 bytes (64 bits); thus, the next must be considered:

- The first four bytes of the input file (`unsigned long`) contain the length of this file. These bytes must not be stored in dynamic memory.
- When the input file was cyphered, previously was extended in order to achieve a length multiple of 8. So, these values are stored in the input cyphered file and must be deciphered, but must not be stored in the deciphered output file.

For example, if the original file is 30 bytes long, to be cyphered is extended to 32 bytes by adding two bytes (whose values are not important). We add a header of 4 bytes with the length of the original file (30), so the final real length of the input file is 36 bytes:

4+30+2. The deciphering procedure works on 32 bytes, but stores only 30 into the output file.

To proceed with deciphering the next function must be defined:

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

where `v` is an array of 2 `unsigned long` to be deciphered and `k` (the key) is an array of 4 `unsigned long` with the values: `{128, 129, 130, 131}`.

Hint: Use the next C standard functions: `fopen`, `fread`, `fwrite`, `fclose`, `malloc`, `free` and `memcpy`. Use the C operator ^ when you need to apply a XOR.