HW2 – Threads.

In this task you should train your threading skills.
The goal is to implement a threadpool mechanism, and implement syncronization (as studied at class)
Just to remind you, we use multi-threading to utilise multi-cores C, and improve performance by that.
Many web-servers use the same approach to suport multiple clients in the same time.

In our task, we will use an encryption algorithm, that is not so fast.
Your goal is to parallelize it, so it runs faster in multi-core system.

The task in details:
You are given a shared (SO) library, compiled for x86, with two functions: “encode” and “decode”.
Also a simple basic_main is included to demonstrate functionality of the library.

As the encription algorithm is made by a beginer student, it will take 5ms for each char, and it’s not capable of manipulating more thatn 1k (1024) bytes of data. More than this will be ignored.

You have to implement a CMD TOOL that will use the algorithm above.

By cmd tool we mean , that your executable will get its working data from stdIn, and put the output to stdOut. A datastream (not a file) maybe used to test your solution.

As an option of stdIn reading, see the stdin_main example. Use it like this:
cat readme.txt | ./tester 2
where readme.txt is a data file (or some stream)
./tester is the compiled name of the stdin example
2 is the encription key

Your tools will get some input on sdtIn, and write the encrypted/decrypted data to stdOut.
“-e” and “-d” flags will be used for encryption and decryption accordingly.
Usage example:
coder key -e < my_original_file > encripted_file
coder key -d < my_decripter_file > my_original_file
or
pipe | coder key -e > encripted_file
cat encripted_file | coder key -d > your_original_file

Note that ‘<’, ’|’ and ‘>’ are stand alone shell command for redirecting input/output/piping
It’s expected that you use the most of the cpu available on the machine.

Keep in mind that there may be automatic test for this excersize.

Good luck !

Technical Notes:

- Submission day – up to 14/05/23 08:00 am
- Put your results in ZIP (not rar/7z/tar etc.) and name it by your ID. (-15 points if not)
- You can sibmit in pairs, name id ID_ID (-15 points if not)
- Failing to submit working Make may lower your point dramaticaly, down to Zero !
- If you are late, you can submit up to 18/05/22 11:59pm. 10 points loose for each day.
- Be aware of linux/windows files transfer. Linux is CaseSensitive, while Windows is not
- Don’t wait for the last minute, as “My VM is died”, “My cat swallowed the mouse” and other subterfuges will not be accepted
- The task should be implementing using C language
- There is no support for ARM architecture