L3: Message logger

Required compilation flags: **-Wall -fsanitize=address,undefined**

The purpose of this task is to implement a simple program that logs received packages. Every packet in the first two bytes encodes the length of the total message. Data received should be saved to the file with the time received (without initial length). Apart from writing down data, our program can process data in 3 ways:

1. Converting lowercase letters to uppercase letters and vice versa ("101AlaMaKota" → "aLAmAkOTA")

2. Replacing all spaces with underscores and vice versa ("152_ Ala_Ma Kota " → " _Ala Ma_Kota_")

3. Combination of the above two ("153_ Ala_Ma Kota " → " _aLA mA_kOTA_")

The method of data modification should be encoded in the first byte of the message (character 1, 2 or 3). The remaining bytes are the login message. Packets with the wrong first byte should be rejected with the error **17Wrong filter mode** sent back to the client (the first two characters also have encode the message length). Packets that do not comply with the protocol should not result in a shutdown program.

The logger should listen on the selected port for TCP connections. It should be able to maintain at most 5 connected clients. When you try to connect a 6th client, the connection should stay rejected with error **23Too much active clients** sent back to the client. The program parameters are:

• path to the log file,

• listening address,

• listening port.

Clients can be simulated using telnet processes. Example: 121Ala has a cat → [12:34] aLA has a cat

Stages:

1. 4 p. The program should listen for data on the TCP socket and save all received data to file. At this stage, service for one customer is enough.

2. 4 p. Support for multiple client connections works (without tracking the number of connections). If the connection to any number of clients is lost, the program must continue to run.

3. 3 p. We add filter functions. The first byte should be skipped when writing to the file. We add a timestamp before each log (hint: ctime() function).

4. 3 p. We are adding functions for rejecting too large packets.

5. 3 p. We add functions to track the number of clients and reject the 6th active connection.