Universitat Pompeu Fabra Barcelona

Lab 01: Delivery Exercise

Understanding Binary and Text Files for Storing Data

Objective: The objective of this practice is to understand the differences between binary and text files for storing data, specifically on handling an array of integers. In this context, you can assume that all input files are well-formed files, meaning:

- **For binary files:** The file size is always a multiple of 4 bytes, since each integer is represented by 4 bytes.
- **For text files:** Integers are correctly formatted as comma-separated values without any syntax errors.

Write separate programs to perform the following actions based on the CLI arguments provided. All programs must be implemented in the C programming language.

- Program 1: **Convert** a file from binary to text or vice versa
- Program 2: **Modify** the content of an element of the array

Below are the descriptions of the programs you need to program:

Program 1: File Conversion between Formats

- **Functionality:** Converts a file from text to binary or from binary to text. Please check the examples provided in the theoretical class for better understanding.
- **Usage:**

```
file_converter -binary|-text inputfile outputfile
```

- **Explanation:**
  - -binary: Indicates that the input file is in binary format and should be converted to text.
  - -text: Indicates that the input file is in text format and should be converted to binary.

Program 2: Array / Data Modification

- **Functionality:** Modifies the content of a specific integer stored in the file, which can be either binary or text. The CLI inputs are whether the file is encoded as text, or in binary, the position of the integer to modify, and the new value. You have to do the modification inline, without using an extra file, nor storing the whole file in memory. They have to work with binary and with text files.
- **Usage:**

```
modify_int -binary/–text  inputfile position value
```

- **Hints:**

- Since integer represented as strings have variable length, you might need to change the size of the file (increase or decrease the size of the file, depending on the old and new values). You can use strlen to check the length of a string.
- Remember that to insert or remove bytes from the middle of a file, you will need to displace the content from that point to the end of the file by copying it. Beware of conflicts, since you are reading and copying in the same positions. Also remember that when you are removing bytes, you need to change the size of the file using ftruncate, which truncates the content of the file at a certain length.

Delivery Instructions:

- You do not need to perform type checks or validation on the entry files. Assume text files are always correctly formatted.
- Submit the following files using the same specified naming conventions below:
  - file_converter.c for Program 1.
  - modify_int.c for Program 2.
- Each program should include its own main function and operate independently.
- Ensure each program compiles successfully and adheres to the CLI argument structure provided. Include the instruction you used for
- Take a screenshot showing your program successfully compiled without errors.
- Submit the programs as a compressed file named `practice1_<student1_student2>.zip`, replacing `<student1>` and `<student2>` with your full names (first and last names). Along with the compressed file, include screenshots showing the program successfully compiled and implemented without errors.
- Be prepared to explain the implementation and use of each program to your peers in the next class. Ensure that all programs are written in the C programming language and follow the provided naming conventions.