Report on exercise #3

Matteo Corain S256654

Laboratory #8 – System and device programming – A.Y. 2018-19

The proposed solution for the exercise makes use of a `Menu()` function, whose argument is a `HANDLE` related to a file previously opened in the main function via `CreateFile()` after having checked its return value. This file, which represents the students' database, is opened with the following arguments:

- The name of the file is the one given by the user as the first command line parameter (`argv[1]`);
- The access mode is `GENERIC_READ | GENERIC_WRITE` (open both for reading and writing);
- The share mode is set to `0` (no sharing);
- The security attributes are set to `NULL` (default security);
- The creation disposition is set to `OPEN_EXISTING`;
- The file attributes are set to `FILE_ATTRIBUTE_NORMAL`;
- The template file is set to `NULL` (no attributes copied).

The `Menu()` function performs an infinite loop, asking the user to input a command; then, based on the first letter of the command, it takes the following actions:

- If the command is "R", it reads the index via a `_stscanf()` operation, then it calls the `ReadStudent()` function to read the student's information into the `student` variable (structure of type `STUDENT`) and finally it outputs that information on the standard output via `_tprintf()`;
- If the command is "W", it reads from the standard input to the student structure the student's information via `_fgetts()` and `_stscanf()`, then it calls the `WriteStudent()` function to write the contents of this structure to the file;
- If the command is "E", it breaks the loop making the program terminate;
- If the command is anything else, it prints an error message.

The only differences among the three versions of the program are found in the `ReadStudent()` and `WriteStudent()` functions:

- In the first case, the file pointer is repositioned by means of the `SetPointerEx()` system call, to which a `LARGE_INTEGER` set to `(n-1)*sizeof(STUDENT)` is passed (first student to be read/written from the beginning of the file) in order to move the file pointer from the beginning of the file, checking the consistency of the passed value with the returned one; after that, a `ReadFile()` or `WriteFile()` operation is performed, checking the number of read bytes against the size of a `STUDENT` structure;
- In the second case, the file pointer is repositioned contextually to the read/write operations by means of an `OVERLAPPED` structure, whose two fields `Offset` and `OffsetHigh` are set to the respective fields of a `LARGE_INTEGER` variable, containing value `(n-1)*sizeof(STUDENT)`;
- In the last case, the file pointer is still repositioned through an `OVERLAPPED` structure, but partial locking of the file is used for performing read and write operations; this is achieved by a call to `LockFileEx()` executed before the operation and a call to `UnlockFileEx()` after the operation.