Report on exercise #3

Matteo Corain S256654

Laboratory #1 – System and device programming – A.Y. 2018-19

The solution makes use of three global variables `next`, `this` and `last` (defined as `int`, so that they can contain also the value associated to EOF). The `main()` function (run by the main thread), after having checked the number of input parameters and the correctness of the file opening operation, performs a `for` loop, which at every step increments a variable `i` (used to keep track of the number of characters processed) and terminates when the end of the file has been reached (the value of the `last` global variable becomes EOF); at each iteration, the loop:

- Creates a processing thread (if at least one character has been read) by means of the system call `pthread_create()`, storing the thread identifier in `tid_p` and using as the thread function `process_this()`, whose argument is set to `NULL` (we don't really need to pass anything to the function, since variable `this` is global);
- Creates an output thread (if at least two characters have been read), by means of the system call `pthread_create()`, storing the thread identifier in `tid_o` and using as the thread function `write_last()`, whose argument is set to `NULL` (we don't really need to pass anything to the function, since variable `last` is global);
- Reads the next character from the input file, by using the library function `fgetc()`;
- Joins the threads with identifiers `tid_p` and `tid_o`;
- Shifts the values of the global variables, by moving the value of `this` in `last` and the value of `next` in `this`.

Finally, the main thread closes the input file and shows a termination message (the actual number of processed characters is `i-2`, since the EOF value needs to be shifted twice before being copied in variable `last`). The two thread functions, whose return value is `void*` and which take as an argument a `void*` value just to comply with the requirements of the `Pthread` library (values returned and passed are always set to `NULL`), perform two simple actions:

- `process_this()` makes the character in the global variable `this` uppercase, by means of the library function `toupper()`;
- `write_last()` outputs the value of the global variable `last` on the console, by means of the library function `fputc()`.