Report on exercise #1

Matteo Corain S256654

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

The first version of the proposed solution makes use of the C standard library functions to write data on the output files. The `main()` function of the program performs the following actions:

- Initializes the random seed based on the current time via the `srand()` function, so that it is ensured that at every execution of the program the seed used to generate random numbers is different;
- Checks the number of input parameters and converts them to integers via the `atoi()` library function, storing their values in variables `n1` and `n2`;
- Calls the function `handle_vector()` twice, to manage the operations on vectors `v1` and `v2`.

This function, which takes arguments the number of the vector to manage (`0` or `1`), the pointer to the vector and its length, performs the following actions:

- Allocates the vector with the given length in the heap by means of a `malloc()` operation, and checks that the allocation has been successful;
- Fills the array with values obtained by calling the `rand_interval()` function, which scales the results of the `rand()` function in the given interval (whose limits are stored in the num-th row of the `limits` matrix); in particular, it performs a modulo operation by $\frac{max-min}{interval} + 1$, multiples the result by the interval (we use 2 in both cases, so that it is assured to select only either even or odd numbers), and finally sums to that the minimum value of the interval;
- Sorts the array by using the library `qsort()` function, whose comparator function is implemented by the `int_comparator()` function, which casts and dereferences the pointers to the two values and returns the difference between the two, so that the return value is negative when the second is bigger than the first and positive when the first is bigger than the second;
- Opens in write mode the respective text file (whose filename is stored in the first cell of the num-th row of the `filenames` matrix), checks the correctness of the operation, writes one by one the contents of the vector by means of the `fprintf()` library function, and closes the file;
- Opens in binary write mode the respective binary file (whose filename is stored in the second cell of the num-th row of the `filenames` matrix), checks the correctness of the operation, writes the contents of the vector by means of the `fwrite()` library function, and closes the file;
- Frees the allocated dynamic memory for the vector.

The second version of the proposed solution makes use of the Unix system calls to write data on the output files. The following modifications have been made:

- Instead of a file pointer (`FILE*`) variable, an integer file descriptor has been used, initialized by means of the `open()` system call (specifying the flags for opening in write mode, creating the file if not existing and truncating it if already existing, in bitwise OR);
- The system call `write()` has been used to write the output binary files;
- The wrapper function `write_ascii()` has been implemented to write numbers in textual format to the output text files; this function selects one after the other the decimal digits of the number (by means of a modulo operation by 10 and a division by 10), transforms them to ASCII codes by summing the ASCII code of 0, and finally outputs the obtained digits in reverse order via `write()` operations;
- The system call `close()` has been used to release the resources associated with the file descriptor.