SDP 2016/17 - Lab 1 - Davide Gallitelli S241521

E01 - C program for file generation

The first task of this laboratory asks to implement a C program which creates a total of four files, fv1 and fv2 as both binary and text files, containing a user-specified number of random integer numbers.

In order to receive from input these values, the approach chosen is to read them from command line as arguments of the program. This means that *argv[1]* and *argv[2]* will have the characters related to the values in input. The *atoi* function is used to return the integer value from the ASCII read by the console. If the number of arguments is different from 3 (./file n1 n2), the program prints an error and exits with a -1 return value.

```
if (argc != 3){
    fprintf(stderr, "Error: wrong # of arguments");
    return -1;
}

srand(time(NULL));
// Take from command line two integer numbers - as parameters
int n1 = atoi(argv[1]);
int n2 = atoi(argv[2]);
```

In order to allocate an array with a dimension defined by a variable, a dynamic array has to be allocated by means of the *malloc* function:

```
int *v1 = malloc(n1*sizeof(int));
int *v2 = malloc(n2*sizeof(int));
```

According to the given specification, the values accepted for the array elements are:

- file v1: even integers between 10 and 100 --> generate even integers from 0 to 90 then add 10
- file v2: odd integers between 21 and 101 --> generate odd integers from 0 to 80 then add 21 To sort the arrays just generated, the *qsort* function of the C *stdlib*.

The *open()* system call allows to both open and create, with specified permissions, files, returning an integer associated to the file descriptor. Our goal is to print to a text file the characters which relate to the integer from the array. To achieve this, we iterate over the array, pushing the values to a string by means of the *sprintf()* call. This calls return the actual number of chars written, which is why it is used through the *j* variable to determine the size of the buffer for the later *write()* call. The *write()* call finalizes the writing of the buffer into the text file, having as parameter the file descriptor, the buffer itself (the string previously generated with the *sprintf()*) and the size of the string, which is *j* times the size of a character in bytes. This value will never exceed *n1*4 + 4* (or *n2*4 + 4*), because the worst case possible is having n1 (or n2) times the value "100", which needs 3

characters as well as one character for a space to separate the values. The last 4 characters are added for CR.

In order to write a binary file, it is sufficient to call a *write()* function on the binary file, writing the generating array (v1 or v2), which will have size of n1 (or n2) times the size of an integer.