Distributed Systems
Bachelor's Degree in Computer Science and Engineering. 2019/2020
Exercise 1: POSIX message queues

We want to design and implement a distributed vector system. For a distributed vector, we define the following services:

- int **init** (char * name, int N). This service allows to initialize a distributed vector of N integers. The function returns 1 when the vector is firstly created. If the vector is already created with the same number of elements, the function returns 0. The function returns -1 on error, e.g., creating a vector that exists with different size.
- int **set** (char * name, int i, int value). This service inserts the value at position *i* of the vector *name*. The function returns 0 on success and -1 on failure, for example, when inserting an element in an invalid position or in a vector that does not exist.
- int **get** (char * name, int i, int * value). This service allows to retrieve the value of the *i* element of the vector *name*. The function returns 0 on success and -1 on failure, for example, you try to retrieve an element in a invalid position or the vector does not exist.
- int **destroy** (chat * name). This service allows to delete a previously created vector. The function returns 1 on success and -1 on error.

State: design and implement using POSIX message queues, the system that implements this service:

1. Implement the server code (*server.c*) that allows to manage distributed vectors. The server must be multithread.

2. Implements the above services (init, set, get, and destroy). The code will run on the file *array.c*. This is the code that provides the client's interface and is responsible of the above services by contacting the server.

3. Implement a client code (*client.c*) that uses the above functions. The client must make at least the following sequence of calls:

```
Init ("vector1", 100);

Init ("vector2", 200);

Set ("vector1", 0, 40);

Set ("vector1", 120, 30);

Init ("vector1", 200);

Destroy ("vector1");

Destroy ("vector");
```

Submission: You must submit the following documentation:

exercise1.tgz file, including the client, server, and array.c codes that implement the services. Also you should include a small report in PDF (no more than three pages) indicating the design and how to compile and build both client and server executables.

Delivery will be made by Aula Global. The submission deadline is: 06.03.2020.