**Pipe**

library used - #include<unistd.h>

You can think at the pipe as a stack, shared by all the processes, after a fork call, where the read is equivalent with popping an element out of the stack, and writing with pushing an element in the stack, but contrary to the stack, the pipe will have the items stored in the original form.

function call: pipe(fd[2]);

the call returns 0 if successful, -1 otherwise

fd is an array of 2 elements, which will point to 2 file descriptors:

* first for reading
* second for writing

We can use the functions:

* write(an file descriptor, element to be written, the size of the element in bits) to write data to the pipe
* read(an file descriptor, element to be read, the size of the element)

Important: RIGHT AFTER YOU END STOP USING A FILE DESCRIPTOR, IT'S RECOMMENDED TO CLOSE THAT FILE