LABORATORY 6

-

OPERATING SYSTEMS

RUFINO GARCIA SANCHEZ

**EXERCISE 1**

* **Write the program which creates two processes: :**
  + **a parent reads data from the pipe,**
  + **a child writes to the pipe**

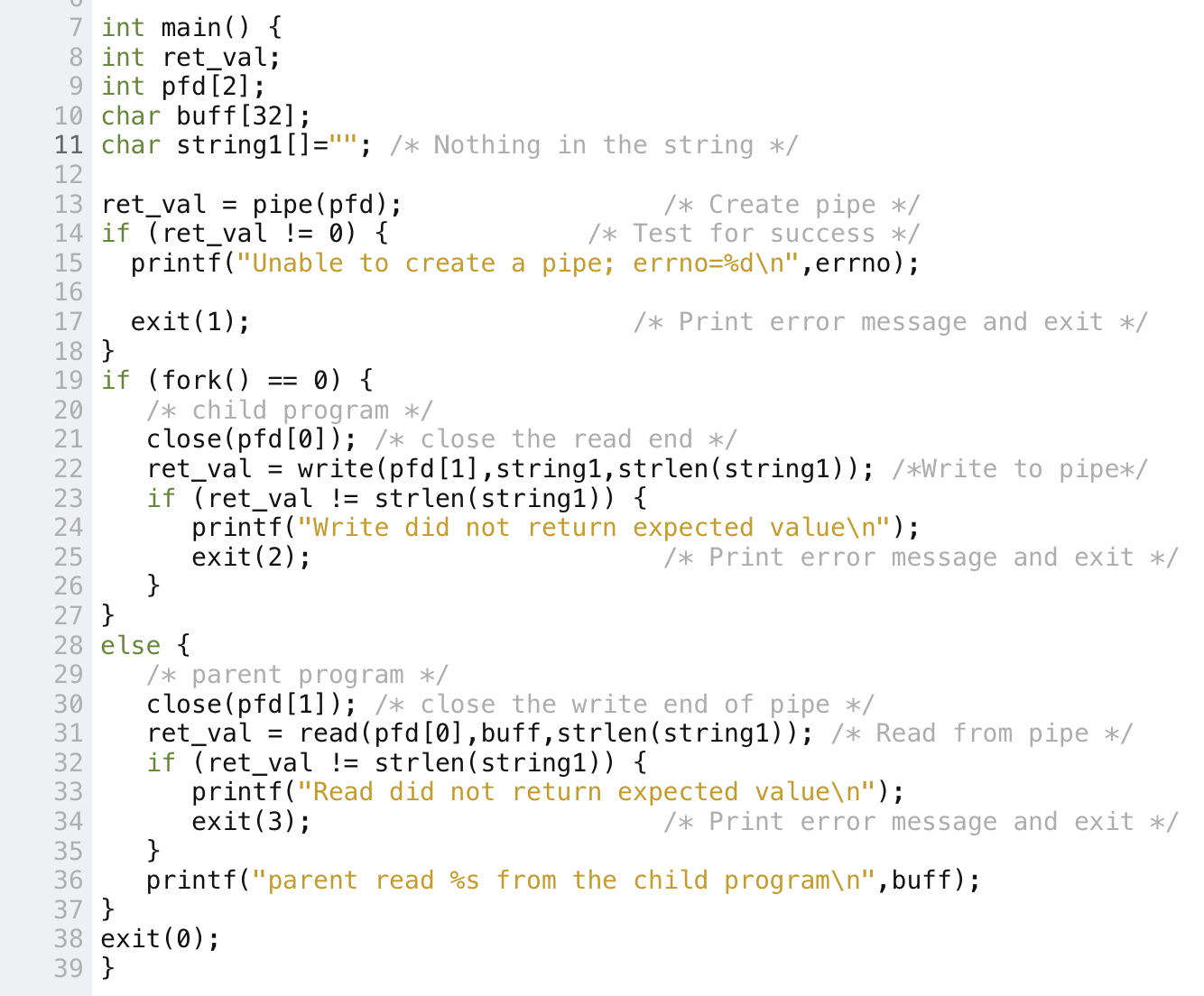
**TASK 1**

The code of the task 1 is this:

Consider the following situation (described below) and explain what happens in these cases:

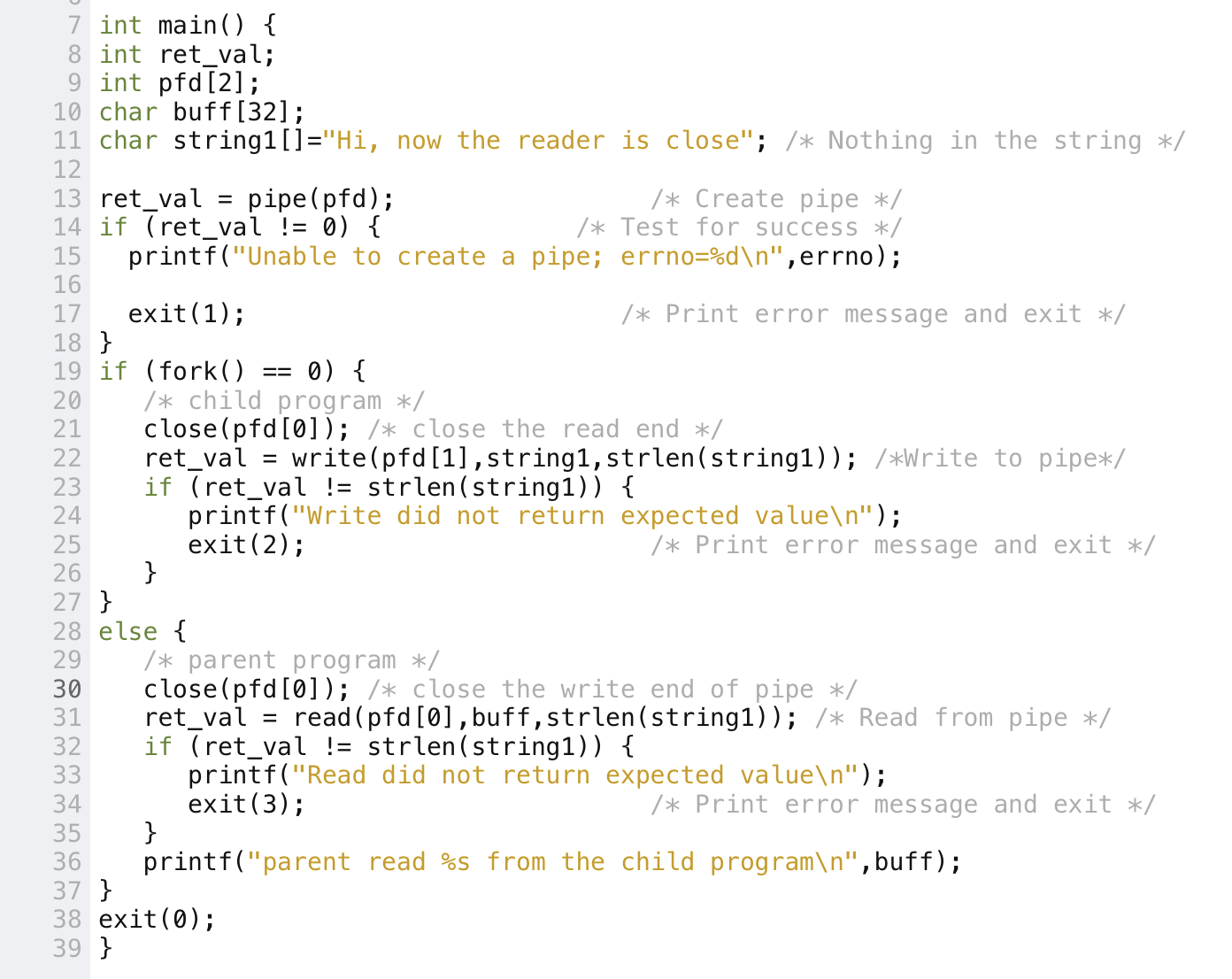
**When writing to the full pipe:**

And this happens when we execute the program:

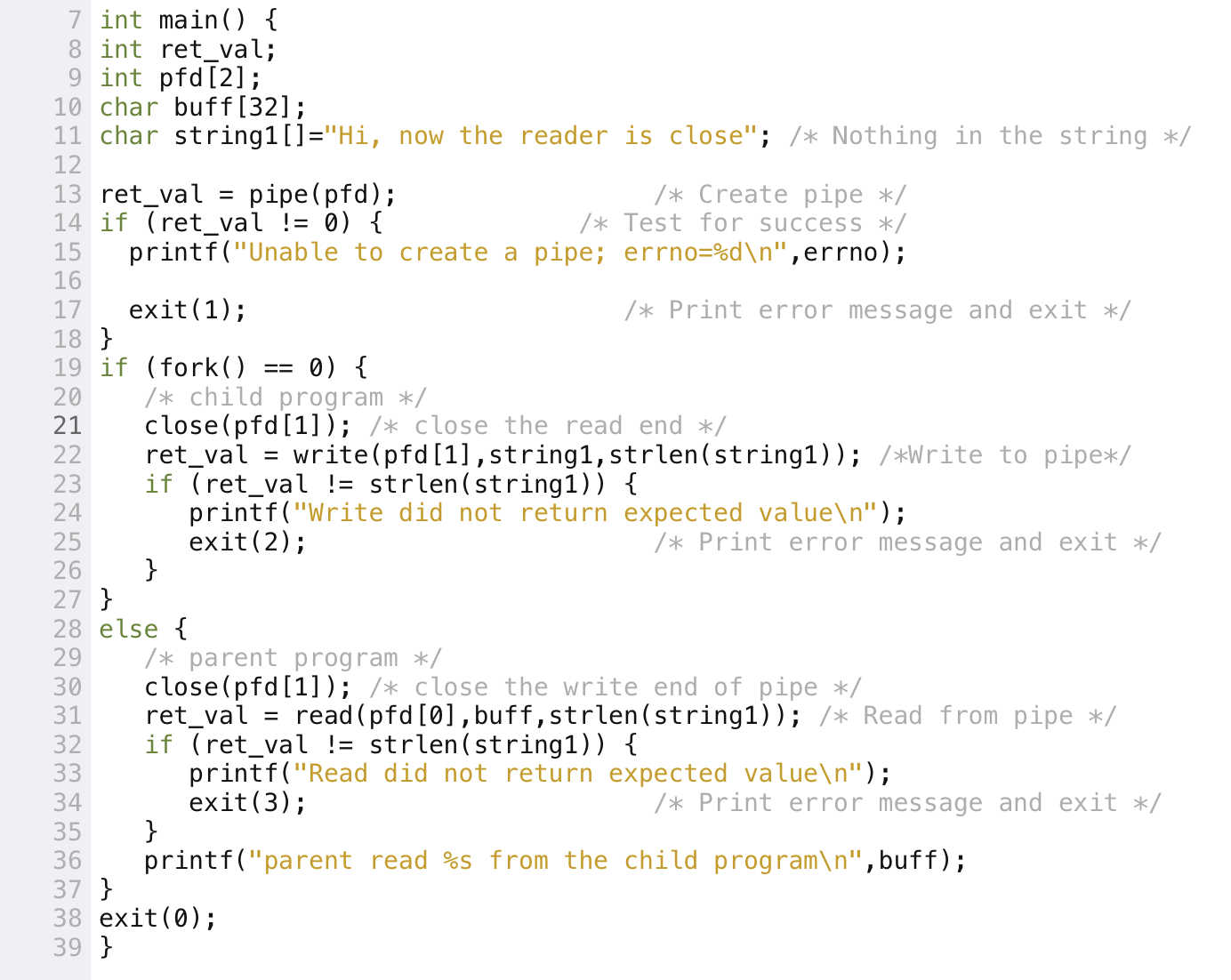
**When reading from the empty pipe:**

And this happens when we run and execute the code:



**When writing to the pipe when the reader had been closed:**

And this is what happens:

**When reading  from the pipe with writer had been closed:**

This is what happens when we run and execute the code:

**TASK 2**

**The code of this program is in the next image:**



In this case I have obtained exactly the same results as the first task. In this part of the exercise I have reached the same goals as the first part so I will introduce my conclusion to achieve the principal and final goal of the exercise that is work with pipes.

I have obtained and I have reached the conclusions that I’m going to explain in the next paragraphs:

First of all I have to introduce that all this information comes from the lectures and also from the man of Linux that is based in unix.

**When writing to the full pipe:**

If a process attempts to write to a full pipe (see below), then write(2) blocks untilsufficient data has been read from the pipe to allow the write to complete.

**When reading from the empty pipe:**

If a process attempts to read from an empty pipe, then read(2) will block until data is available.

**When writing to the pipe when the reader had been closed:**

If all file descriptors referring to the write end of a pipe have been closed, then an attempt to read(2) from the pipe will see end-of-File.

**When reading from the pipe with writer had been closed:**

If all file descriptors referring to the read end of a pipe have been closed, then a write(2) will cause a SIGPIPE signal to be generated for the calling process.