****

**Universidad Autónoma de Guadalajara**

Biomedical Engineering

Peripherals and Interfaces

Jesús Arnoldo Zerecero Nuñez, 2885993

Edwin Francisco Zamora Oros, 2874382

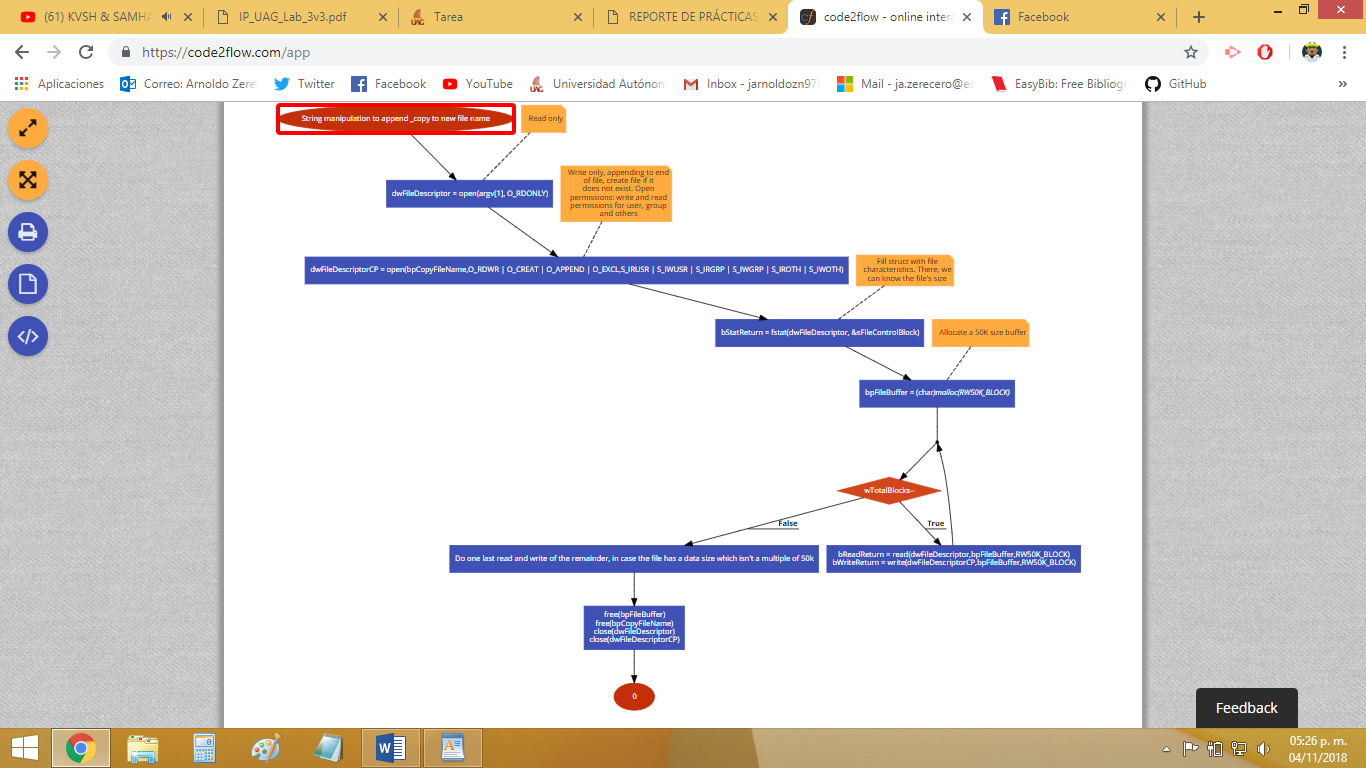
*“Lab 3: Cp Bash Command”*

5/11/18

**Introduction:**

In this third lab practice, the objective was to develop a simplified version of the “cp” bash command. The function created must generate an exact copy of the source file received as a parameter, no matter what kind of file it is. The generated file must be created on the same path as the source file, and must have the same name as the source file, with “\_copy” added to the file name string. Ex: Newfile -> Newfile\_copy, Newfile.txt - > Newfile\_copy.txt.

The application was developed using the open, close, stat, read and write APIs from die.net, creating the file with open(), knowing its size with stat() and copying its contents using read() on the source file and write() on the destination copy file.

**Flowchart:**

**Development (issues):**

* **Figuring out and applying the correct method to generate the copy file name, based on the original file name.**
  + As the original file name is passed as a parameter (argv[1]), the copy file name needed to be generated by manipulating and concatenating the original file name string.
  + If the file name had an extension, the “\_copy” needed to be placed in the middle of the file name string (before the extension’s ‘.’), which means a slightly more complex copy name generation code needed to be implemented.
  + We first needed to understand how argv[] behaves as a string.

**Conclusion:**

**Arnoldo:**

This third practice was a relatively easy one, as we had little to no trouble developing it. Fortunately, everything we needed to know about the file APIs was found in die.net, so everything went smoothly, most of the time. The most time consuming element of this practice was finding and trying different corner cases, in order to find out-of-plain-sight flaws in our code. In the end, the code seems extense, as there are a lot of strategically placed conditionals which ensure nothing goes wrong.

**Edwin:**

The use of files for this practice was fundamental, so we used the functions found in die.net, which are the most common for file management. We had some problems because when copying files that were too large we needed to divide them into small blocks and send them in parts, but the packages had to contain the whole file, so we needed to divide it into a multiplo of the actual file size. In the end we were able to solve it and send small packages and copy the entire file.