SYSTEMS ARCHITECTURE

LAST NAME: .....
NAME: .....

LABORATORY SESSION 7: Processes

PROBLEM 1 (3.5 points) Exam 11.11.2022

Previous considerations:
1: The ordered steps suggested bellow for the problem's resolution are mere indications, they are NOT requirements as there can be many ways of solving the problem. (Though every code written in ***C*** language must be correctly sequentially ordered, of course).
2: Declare and use the variables you estimate necessary. Give then proper initial values to avoid errors.
3: Making unnecessary initializations is not penalized, however NOT doing necessary initializations DOES reduce the grade.
4: In the appendix at the end of the problem (see below) there is a list of standard library function declarations aimed to help in resolving this problem. Each item in the list consists of the first lines of information retrieved with the standard Unix *man* command. You can use this information to know which #*include* files are needed before using the functions. *Not all functions in the list are necessary*.

Part 1 (qualifies 2 points over 10)
You must build a function called ***get_number_of_bytes,*** that yields the correct number of chars (bytes) contained in a text file whose name is given to the function as a parameter. For this you shall:

- (0,3 points) Write the first implementation line of the function, provided that it must return in integer; that the name of the function must be *get_number_of_bytes* and that it receives as its only parameter a standard C language text string (ended with '\0') containing the name of the file to process.
- (0,2 points) Write the code needed to open the file in *read* mode, whose name is in the string given as parameter to the function.
- (0,1 points) Manage the possible error that can be returned when the opening fails. Return a negative value in this case.
- (1 point) Implement a loop that counts the number of characters while the opened file is being readed.
- (0,2 points) Release the resources associated to the use of the file (i.e., shall close it).
- (0,2 points) Return the byte count as result if there were no errors. Otherwise, a negative integer if any error occurred.

Part 2 (qualifies 7 points over 10)
Write the ***main*** function of a multiprocessing program that receives as input arguments the names of an undetermined amount of text files containing plain text. This program also …

- Shall print the number of characters (bytes) contained in each input file.
- Shall create a child process per file name given. This child process will process the file whose name is provided.
- The parent process will release all the assigned resources related to child process management. This means that there cannot remain *zombie* children processes after parent's execution.

To write this ***main*** function you must:

- (0,1 puntos) Write the first implementation line of ***main***, which is the standard one used along this course.
- (0,1 puntos) Ensure the program returns successfully, not doing anything whether it receives no parameters.
- (4 puntos) Write the ***for*** loop that creates one child process per input file name introduced in the command line and delivered through main's arguments.
- (0,3 puntos) Notify, printing in console, the failure when any child process creation cannot be performed. This fact will NOT stop the program, nor any other part of it.

Moreover:

- (0,1 puntos) And any child process will get the number of bytes of a file by calling the function *get_bytes_number*. This function will be described in the first part of the problem

- (0,2 puntos) Any given child process shall print, in the console, the number of bytes in the file computed by *get_number_of_bytes*. The process must notify, through the console, whether the computation returned an error value.
- (0,1 puntos) Each of the children shall end after printing its results. At the same time, it shall return a constant suited to whether the process work was successful or not.
- (2 puntos) Just before the end, the parent process shall wait for the conclusion of all their children. It shall free all resources obtained (transparently) for their execution.
- (0,1 puntos) The parent process will exit the main function returning a value meaning success, in all cases.

Part 3 (qualifies 1 point over 10)
Modify the ***main*** function above, in such a way that there cannot be more than three child processes running at a given time. That means that the parent process must *wait* when one third process is launched, till one of the three ends its execution.
***Suggestion*:** count the number of correctly launched child processes. *wait* when the count reaches three and decrease the counter when the wait finishes.

Appendix: Some standard C library functions documentation retrieved with the ***man*** Unix command.

PROBLEM 2
Given the previous Problem 1, lets do a variation.

Part 1 (qualifies 2 points over 10)
You must construct a function, which we will call*:* ***get_number_of_lines***, that returns the correct number of lines contained in a text file whose name is passed as a parameter. To do this you must:

- (0.3 points) Write the first line of implementation of the function, knowing that it returns a standard integer; that the name of the function is *get_number_of_lines* and that it accepts as its only parameter a standard C language text string (terminated with the null character: '\0'), containing the name of the file to be processed.
- (0.2 points) Write the code to open the file, in *read* mode, whose name is the one contained in the string passed as parameter to the function.
- (0,1 points) Manege the possible opening error by returning a negative value in case of error.
- (1 point) Implement a loop that counts lines of text as the file opened in the previous paragraph is read.
- (0.2 points) Release the resources associated with the use of the file (i.e., shall close it).
- (0.2 points) Return as result the value obtained from counting the lines, if this process has been successful, or on the contrary a negative integer, in case of error.

Part 2 (qualifies 7 points over 10)
Write the ***main*** function of a multiprocessing program that receives as input arguments the names of an undetermined amount of text files containing plain text. This program also …

- Shall print the number of lines of text contained in each input file.
- Any given child process will know the number of lines of its file by calling the function *get_number_of_lines* function. This function will be described in the first part of the problem.
- Any given child process shall print, in the console, the number of lines of the file, calculated by *get_number_of_lines*. The process must notify, through the console, whether the computation returned an error value.

Part 3 (qualifies 1 point over 10)
Modify the ***main*** function above, in such a way that there cannot be more than three child processes running at a given time (as problem 1)