Assignment 5: Process Creation

CS 3424 - Systems Programming — Sam Silvestro - UTSA

Assignment 5: Process Creation and File Operations (75 Points)

1 Objectives:

Practice with system calls to: a) perform file operations; and b) create and use processes.

2 Description:

Overview: You will write a program (file named *childwords.c*) to count out the number of words contained within multiple text files. Specifically, the program will first determine the number of files to be processed. Then, the program will create multiple processes, with each process being responsible counting the words within one particular input file. The format to run your program with input parameters is as follows:

`./childwords file_1` *`file_2 [...] file_n`*

Details: First, the program needs to determine the number of files to be processed. This can be done using the `argc` parameter of the main function. Then, the `argv` parameter can be used to retrieve the name of each file. After that, the `fork()` system call will be used to create multiple processes (one for each file). For each child process, it should simply invoke a function to count the number of words inside a specific file and print out the result as shown below [note that no interprocess communication is required, as child processes will report (i.e., print out) their results individually]:

`Child process for` *`file_x`*`: word count is` *`num`*

However, child processes should be executed and run concurrently, not in series/sequentially. This means that on a multicore system such as the "`fox`" machines, child processes have the potential to be in execution simultaneously (i.e. in parallel).

The main process should wait until all child processes finish and report their results. Then, the main process will report at the end of the program by printing out the following message (or one applicably similar to it):

`n-x` `of n files counted successfully.`

For word counting, you will simply use any combination of space and tab characters as delimiters. Anything not separated by a combination of these characters will be counted as a single word. For instance, in the example, "`The first program is hello-world`", your program will report five words (where "`hello-world`" is counted as a single word). You could compare your results against those of the `wc` utility that is available on Linux machines. Your program must not crash when parsing text documents that

only contain words consisting of 100 characters or fewer!

3 Submission Requirements

The assignment needs to be submitted on Blackboard.

Source code: This assignment should have only one source file (*childwords.c*) and at most one optional header file. Finally, please include a `Makefile` in order to facilitate easy compilation and testing. You program will be graded based on functional correctness.

Cheating/collusion will be detected and punished according to university guidelines.

Submission: Please submit all of the above files zipped into a single archive named **a5-*abc123*.zip** and upload the file on Blackboard, where *abc123* represents your myUTSA ID.