Andrew Hoskins

811939260

CSCI 4730

To make sure everyone is doing their fair share in my multi-process structure, I decided to get how big the file is in bytes and then divide it by the amount of processes I am allowed to have. The amount of processes I am allowed to have is dictated by numJobs. Typically, the division would not be a nice integer so to handle the remainders in an easy fashion, we will just put it with the last process. This is implemented after we have opened, seeked, and closed the file. For implementing the communication to satisfy the IPC channel, a loop with pipe and fork will be used. This helps us ensure that no child process would be left behind without a method to communicate. Still within this loop, the child will communicate the count_t over the pipe. The word_count given was word_count(fp, 0, fsize), but I changed it to word_count(fp, offset, childSize) to include the calculation of file offset and size to read for each child for better efficiency .Finally, the parent of the child will save the information of the child and what they communicated in the struct of plist_t, and the child process will return.

My program handles crashes by using WIFEXITED(status). I chose this method because our professor hinted that this would be a “better way” in the ch3-extra slides. This method to exit the status of the child allows us to easily identify if the child terminated normally by returning a nonzero value. As per the directions and the comments given, we have to wait for all the children which is why we used wait(&status) inside the while loop that ends when our counter of total task successfully done passes the number of jobs (numJobs) needed. However, when a task is not done successfully it means that a child has crashed so to fix this we will create another process and pipe. This is handled following the fork() call where the child will use the same data values as the failed one whose info is in the plist. This will keep happening until it succeeds if the child is just unlucky. Once a process is successful, we will add one the counter and update our total lines, total words, and total characters, then send the confirmation to the next process waiting with the pipe. This will repeat until we are out of the loop.

Additional Notes: The physical examples given states to use ./wc_mul.c 4 large.txt 20 to run it however to run my files plase use ./wc_mul 4 large.txt 20 (processes, document, and crash percentage (up to 50) can all be changed)

Use make to compile if you want to recompile my code.