Name: Ethan Mok
ID: #811383355

Design of the multi-process and IPC:

In my program, I used a for loop to call the fork() function a number of times equal to nChildProc as inputted in the command line. I created a 2d array for my pipe using the dimensions p[2*nChildPrc][2]. The reason I used 2*nChildProc was because I found that it was easier to send and receive data from the parent and child using a pipe with twice the number of channels. So the parent sent data and the child read the data during the first half of the array and the child sent data and the parent read data using the second half of the array. This ensured that the parent process did not accidentally read data that was meant for the child and vice versa. This approach worked and I ended with the correct values being read by all processes.

One important job the parent process does in the multi-process is calculate the offset that the child process needs to begin to read from. That value is then sent through the pipe to the child process to let that process know where to begin to calculate from so that the entire file is read. This value is also stored in an array to keep in case a child process crashes and a new child process needs to know what offset to calculate from.

For the child process, each child opens the inputted file, calculates the number of bytes that need to be read, reads from the pipe what offset they should start reading from, and reads into count the number of characters, words, and lines. It then sends the value of count through the pipe back to the parent process.

Design of the crash handler:

My parent process uses waitpid() function to wait for each individual child. If that child crashes the program catches the signal that is sent by the crashing child process using WIFSIGNALED(status). When that signal is caught, the program then creates a new fork to re-do the work of the crashed child. This new child process uses the same code as the original child process and functions exactly the same way as the original child process. By storing all of my offsets in an array, the program can use that array to tell these new child processes to begin calculating from the same offset as the crashed child process.

I placed this code in a while loop with a boolean condition. The program will continue to create forks until a child process successfully exits and does not crash. When this happens, the boolean condition will evaluate to false and exit the while loop.