William Garrett Corder

- **Multiprocessing**
  - Within the first for loop wc.c forks to start a number of processes equal to nChildProcesses called looping_process. Each of these processes creates a new fork() in children[] to run word_count using offset[i] and bytes_to_count[i] such that each process has a designated portion of the file to search.
  - looping_process felt necessary to include such that all sections of the file could be searched simultaneously.

- **IPC**
  - Each child process has a pipe it correlates to stored in fd[][].
  - When the process in children[i] successfully finishes running word_count and collects data in the count_t count it writes this data to fd[i][1].
  - Once all child processes have finished running another for loop iterates through the pipes and reads the count data from fd[i][0].

- **Error Handling**
  - The outer process called looping_process handles the errors using the variable status. Each time the child process of looping_process concludes status is updated.
  - When status = 0 it means that the child process exited safely and has written its information to the appropriate pipe.
  - If status does not equal 0 it means that the child process did not exit safely so it restarts the child process to run again.

Code Overview

- wc.c uses 4 arrays based on the variable, nChildProcesses, given in the command line argument. Each value in the arrays corresponds to one of the child processes intended to run word_count().
    - The function fill_offset_arrays() calculates the bytes each process will need to search as well as the offset value they will need to start from then places them into the arrays offsets[] and bytes_to_count[].

- The bulk of main() happens in the first for loop which iterates the variable i from 0 to nChildProcesses - 1.
    - First it opens the pipe fd[i] to transfer data from the process in children[i] to the parent process via IPC
    - It creates a "looping_process" for each iteration of the for loop. Each looping_process is intended to run word_count on the subprocess children[i] until the function is successful. The purpose of doing this in a subprocess is to search each section of the file concurrently.
        - Opens the file pointed to by command line arguments, each process needs the file to be opened for it to search through.
        - Initializes the variable status and enters a while loop until status = 0 meaning the process reached exit(0) and exited safely.
        - Creates another fork() called children[i] inside of looping_process to run word_count while the parent of children[i] waits for an exit status.
            - children[i] runs word_count() and either crashes or succeeds. If it succeeds it writes the count data to fd[i][1] to be retrieved by the parent of looping_process then exits.
        - When status = 0 looping_process exits and the file is closed.

- Once the for loop ends the main function waits for all children processes to conclude. Then it enters another for loop to iterate through all of the pipes in fd[][] correlating to the children processes and collect the count data stored in the pipes.
- Once all the totals are added the results are printed