Project Write-Up Samuel Adams and Tom Morin

Project 1\_1

2a. We attempted using wait commands to synchronize both the Child and Parent process. When the parent would write to the child, it would wait until the Child was done to write.

2b. We chose bi-directional pipes, as it was the most sensible option when transferring data between processes. Data could be transferred between both ends of the pipes without obstruction.

2c. Shared memory could most likely be used in this program, but it would not be advisable to use it for this situation. It is not known how large the file that is being read would be, so the proper space allocation for the shared memory would be obfuscated, therefore it is not recommended to use this approach.

Project 1\_2

2a. We implemented threads, which were a lot more accessible and robust than pipes or other IPCs. The threads were quite easy to implement. A parent thread sent data to a child thread, that ran a runner thread process that counted the number of alphabetical characters.

2b. We shared data between the threads through global variables as well as exiting the thread at a given time.

Part 3:

I think that thread based would be more efficient given the size of the problem at hand. Threads have pretty low overhead and generally use less IPC, unlike process based. Though, for bigger problem sizes such as running entire processes, it is probably better to use the process based approach. It would be a very situational decision. Perhaps Thread based would be better for smaller programs while process based would be better for industrial sized programs.