Christopher Brant
ECE 3220
Due 3/5/18

ECE 3220 Intro to Operating Systems Project 1

Throughout the writing of this project, there were a few difficulties in writing and utilizing the pipe() and fork() aspects of this program. Overall, this project did not take too much time, and was fairly straightforward for the most part, however some problems arose during the writing of the source code for this program. Some issues that arose during writing or debugging were with the random number generator, the pipe() function, and the fork() function.

Once I had finished writing all the code for this program, the average amount of numbers the producer sent to the consumer was far below the expected value which tends to be ~12000 numbers sent, and I was originally sending ~7000 to ~8000 numbers. After revisiting my source code, I realized I was "modding" by 4, but not adding 1 so that I was getting a number between 1-4 and not between 0-3. Once I fixed that I also fixed the random number being sent, as it was sending 0-999999, having the same problem. So that was fixed as well. Correcting this issue resulted in negligible differences in final averages, but much higher amounts of numbers being sent, ~12000 normally, which is what was expected.

Moving to problems with pipe(), there were very few of these overall, except with my initial setup. When initially setting up the pipe, I forgot to create a two-value integer array for the pipe ends to be saved in. Yet I caught myself before compiling and did not have any compile or run time issues with pipe. All ends of the pipe are eventually closed as well, as the producer immediately closes the read end, the consumer immediately closes the write end, and when both of their loops end, they close their own write and read end, respectively. The pipe is also not deemed closed until all ends of the pipe are closed in both processes/threads. The last hurdle that was jumped by my code for pipe() was remembering to check if the pipe() returns an error, and in that case the program will exit with an error message.

Lastly, errors or difficulties with fork() were few and far between for myself, as ECE students have already worked in fork() extensively. However, similarly to pipe(), I had to go back and add an error checking conditional in the case that my fork() function failed to create a child process. The last quasi-issue with fork() was that when using the wait() function I forgot to just use NULL as the argument and wait for the one child process to return instead of giving the specific child process PID to it, as doing so gave a compile error.

Overall, this project was not excessively difficult, and there were a relatively few amount of issues in writing it. The process from start of writing to end of debugging took no longer than an hour, so in my mind this is the perfect length of project. This was a good project in my opinion, let's have more like this.