CISC324: Operating Systems

-------Lab 1-------

January 13th, 2019

By: Ryan Protheroe | 20069587

Exercise 1.

I. Program eXer_1.c does not perform the correct computations for the given scenario. Currently, it is only displaying the total from the first half ($0\ to\ \frac{n}{2}$) of computations for a given n value. The child process for computing the latter of the sum of the sequence is not returning a value to the parent, there is no exchange of information.

II. To remedy this issue, exit() and wait() system calls were used to relay information from the child process to the parent process. The exit() system call in the child process returns a signal caught by wait() in the parent process. Wait() is passed a pointer to the exit() value of the child process. The value returned can be divided by 255 to yield exact (correct) values for the latter sum of the sequence. This however contracts us with a limitation on the solution since the exit status cannot exceed 255, its limited to 8 bits. Values of n greater than 25 will not yield correct computations given the current solution.

Exercise 2.

I. **Executing eXer_2.c**

Test #	Final Value in nums.txt
1	6416
2	5000
3	5000
4	5170
5	5045
6	5071

Figure 1: Final values from the execution of eXer_2.c

As we can see from *figure 1*, eXer_2.c is not producing correct results. Since the final values vary, we can assume this is the result of overlapping when concurrent processes are reading and writing from the same file. The three children are incrementing the same value the majority of the time, and not incrementing the result of another child. To put it simply, the children are not waiting until another child finishes, they're just all going at it simultaneously.

II. Program was modified so that the final value is always 15000 in nums.txt.