Operating Systems Project 1: Shared Memory

***Abstract*—The shared memory project serves as an introduction to the concept of shared memory and the problems that can occur if shared memory is not protected adequately. In particular, this project will demonstrate how processes are created with separate functions and using the fork() function to create child processes. This project will also assist in understanding how to create and connect shared memory segments, how the parent waits until the child processes have exited and detaching and removing shared memory.**

I. Introduction

THIS project requires the creation of four processes; each process shares a shared memory variable from a shared memory struct called 'shared_mem *total'. Each process will increase the shared memory variable by 100,000, 200,000, 300,000 and 500,000 respectively. Each child created from the fork() function will call a function 'process#()' respectively, where '#' represents the chronological order of the process being called by the child. When the child process is finished, the parent process will release the shared memory allocated to the child by using the 'wait()' function; this function allows the parent process to know when the child process has finished. Finally, by using the functions 'shmdt()' and 'shmctl()', the shared memory is detached and removed in order to prevent the program from crashing due to memory segments being in the system indefinitely, as well as preventing the degradation of system performance.

II. Execution

This project successfully compiled and executed on the Linux Red Hat Operating System-specific servers at USF. Instructions on compilation and execution are on a README file.

III. Analysis & Results

Upon several executions of the main file, it was noted that processes were running concurrently and as such caused fluctuating end values for each process. The results from the trials are as shown in the figure to the right:

From the output shown, it can be seen that each process' value fluctuates from the actual number to be increased by. The main cause of this is from multiple processes running concurrently.

IV. Conclusion

Based on our analysis, we can conclude that problems occur when shared memory is not protected adequately.

```
[dfong@osnode12 proj1]$ ./proj1

Process 1: = 99689
Process 2: = 201868
Process 3: = 261627
Process 4: = 494449

Child with ID: 16585 has just exited
Child with ID: 16586 has just exited
Child with ID: 16587 has just exited
Child with ID: 16588 has just exited

End of Simulation
[dfong@osnode12 proj1]$ ./proj1

Process 1: = 99814
Process 2: = 183327
Process 3: = 246955
Process 4: = 512875

Child with ID: 16590 has just exited
Child with ID: 16591 has just exited
Child with ID: 16592 has just exited
Child with ID: 16593 has just exited

End of Simulation
[dfong@osnode12 proj1]$ ./proj1

Process 1: = 123539
Process 2: = 198251
Process 3: = 211843
Process 4: = 520201

Child with ID: 16595 has just exited
Child with ID: 16596 has just exited
Child with ID: 16597 has just exited
Child with ID: 16598 has just exited

End of Simulation
[dfong@osnode12 proj1]$ ./proj1

Process 1: = 90879
Process 2: = 192947
Process 3: = 206589
Process 4: = 419407

Child with ID: 16600 has just exited
Child with ID: 16601 has just exited
Child with ID: 16602 has just exited
Child with ID: 16603 has just exited

End of Simulation
```