CECS 326-01 Operating Systems

William Luu

Assignment 4

Due Date: 11/19/2024

Submission Date: 11/19/2024

Program Description

The program is an enhanced version of Assignment 3's program, where the master program creates a shared memory segment and slave processes that access the shared memory segment. By implementing a semaphore in the slave processes, it eliminates the possibility of a race condition, and enables mutual exclusion between the slave processes.

master.c

The master process first takes in the command line arguments of the number of child processes to create, and the name of the shared memory segment. By using the POSIX implementation of shared memory, the master process creates a shared memory segment using shm_open(). It then configures the size by using ftruncate(), which is set to 4096. Mapping the master process to the shared memory segment is done by using mmap() which allows the master process to have access to the shared memory. A struct CLASS pointer named shm_ptr is created to have access to the members of the struct CLASS. Initially, by using shm_ptr, set n indexes of report[10] to be 0 to display the initial values in report[10]. Then initialize the semaphore to be 1 using shm_ptr. The master process will then create and execute child processes and wait until all child processes terminate, then print the updated contents of report[i]. Finally the process will destroy the semaphore, detach and remove the shared memory segment, and then terminate.

slave.c

The slave process opens the shared memory segment using shm_open(), as well as map to the shared memory segment using mmap(). Using a struct CLASS pointer named shm_ptr similar to the master process, the slave will use sem_wait() to check if any other slave process is accessing the shared memory segment. When the waiting is complete, the slave process will add their child number into the report into the proper index, and then increment the index by 1. The process will then use sem_post() to increment the semaphore by 1 to allow other processes to access it. Then the process will detach from the shared memory using munmap() and terminate.