**CECS 326 Sec01**

Operating Systems

Elias Woldie (ID 027805724)

Assignment 5

Due Date: 12/05/2023

Submission Date: 11/07/2023

**Program Description**

**1. What all the programs together are designed to do?**

The program (**master.c, slave.c,** and **myShm.h**) are designed to demonstrate inter-process communication and synchronization using shared memory and semaphores in a POSIX-compliant environment. The primary goal is to manage multiple child processes (**slaves**) that interact with a shared memory segment under controlled conditions to prevent race conditions. The **master** process coordinates the creation and management of shared resources and child processes. The child processes perform specific tasks in a synchronized manner using these shared resources.

**2. What each individual program does?**

* **master.c**: This program initiates the shared memory segment and two semaphores, creates multiple child processes, and waits for them to complete their execution. It manages the creation and cleanup of shared resources, including shared memory and semaphores. The **master** process creates a shared memory segment for inter-process communication, a semaphore for I/O control, and another semaphore for managing access to a shared index variable. After all child processes have finished their tasks, it prints the contents of the shared memory segment, showing the work done by each child process. Finally, it cleans up by detaching from and removing the shared memory segment and unlinking the semaphores.
* **slave.c:** Each instance of this program, created by **master.c**, is responsible for interacting with the shared memory segment and semaphores. A **slave** process accesses the shared memory and uses the index semaphore to safely read and increment an index variable in the shared memory. This ensures synchronized access to the shared resource. The **slave** writes its process number in the shared memory array at the position indicated by the index variable. Additionally, it uses the I/O semaphore to manage its output operations, ensuring that print statements from multiple slaves do not overlap. This program demonstrates how to safely access and modify shared memory using semaphores to avoid race conditions.