Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

UCS1411 – Operating Systems Laboratory

Lab Exercise 6: Implementation of Producer/Consumer Problem using Semaphores

<u>Study the following system calls</u>

Semaphores – sem_init, sem_wait, sem_post, sem_destroy – POSIX, pthread

semget , semctl, semop (for your understanding)

Shared memory - shmget, shmat, shmdt, shmctl

Assignment 1:

Aim:

To write a C program to create parent/child processes to implement the producer/consumer problem using semaphores in pthread library.

Procedure:

1. Create a Shared memory for buffer and semaphores - empty, full, mutex

2. Create a parent and a child process one acting as a producer and the other consumer.

3. In the producer process, produce an item, place it in the buffer. Increment full and decrement empty using wait and signal operations appropriately.

4. In the consumer process, consume an item from the buffer and display it on the terminal. Increment empty and decrement full using wait and signal operations appropriately.

5. Compile the sample program with pthread library

```
cc prg.c - lpthread
```

Assignment 2:

Modify the program as separate client / server process programs to generate ‘N’ random numbers in producer and write them into shared memory. Consumer process should read them from shared memory and display them in terminal

Base Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <semaphore.h>
#include <pthread.h> // for semaphore operations sem_init,sem_wait,sem_post
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <sys/wait.h>
#include <sys/errno.h>
#include <sys/types.h>
extern int errno;
#define SIZE 10 /* size of the shared buffer*/
#define VARSIZE 1 /* size of shared variable=1byte*/
#define INPUTSIZE 20
#define SHMPERM 0666 /* shared memory permissions */
int segid; /* id for shared memory bufer */
int empty_id;
int full_id;
int mutex_id;
char * buff;
char * input_string;
sem_t *empty;
sem_t *full;
sem_t *mutex;
int p=0,c=0;

// Producer function
void produce()
{
        int i=0;
        while (1)
        {
            // If buffer not full, gain access to shared memory
               //write into memory one character at a time
            //release the semphores
         } //end-while
} //producer fn

// Consumer function
void consume()
{
     // If buffer not empty, gain access to shared memory
     //consume a character from memory and display it on screen
             //release the semphores
} //end -while
} //consumer fn
//-------------------------------------------------------------
Main function
//--------------------------------------------------------------
int main()
{
        int i=0;
        pid_t temp_pid;
        segid = shmget (IPC_PRIVATE, SIZE, IPC_CREAT | IPC_EXCL | SHMPERM );
        empty_id=shmget(IPC_PRIVATE,sizeof(sem_t),IPC_CREAT|IPC_EXCL|
        SHMPERM);
        full_id=shmget(IPC_PRIVATE,sizeof(sem_t),IPC_CREAT|IPC_EXCL|
        SHMPERM);
        mutex_id=shmget(IPC_PRIVATE,sizeof(sem_t),IPC_CREAT|IPC_EXCL|
        SHMPERM);
        buff = shmat( segid, (char *)0, 0 );
        empty = shmat(empty_id,(char *)0,0);
        full = shmat(full_id,(char *)0,0);
        mutex = shmat(mutex_id,(char *)0,0);
        // write code to initialize Semaphores Empty , Full & Mutex using sem_init()
        printf("\n Main Process Started \n");
        printf("\n Enter the input string (20 characters MAX) : ");

         //get string from user, write code to create parent and child and call producer and
consumer functions

        shmdt(buff);
        shmdt(empty);
        shmdt(full);
        shmdt(mutex);
        shmctl(segid, IPC_RMID, NULL);
        semctl( empty_id, 0, IPC_RMID, NULL);
        semctl( full_id, 0, IPC_RMID, NULL);
        semctl( mutex_id, 0, IPC_RMID, NULL);
        sem_destroy(empty);
        sem_destroy(full);
        sem_destroy(mutex);
        printf("\n Main process exited \n\n");
        return(0);
} //main
```