Sowmya.R

185001159

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1411 - OPERATING SYSTEMS LAB

Batch: 2018-22 **Academic Year: 2019-20**
Class: CSE C **Faculty:Mrs.S.Lakshmi Priya & Mr.N.Sujaudeen**

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

Assignment 1:

Aim:

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

```c
#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 */
```

```c
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(i>=strlen(input_string))
        {
            printf("\n Producer %d exited \n",getpid());
            wait(NULL);
            exit(1);

        }


        printf("\nProducer %d trying to aquire Semaphore Empty \n",getpid());
        sem_wait(empty);

        printf("\nProducer %d successfully aquired Semaphore Empty \n",getpid(
));

        printf("\nProducer %d trying to aquire Semaphore Mutex \n",getpid());
        sem_wait(mutex);

        printf("\nProducer %d successfully aquired Semaphore Mutex \n",getpid(
));

        buff[p]=input_string[i];

        printf("\nProducer %d Produced Item [ %c ] \n",getpid(),input_string[i
]);
```

```
        i++;
        p++;

        printf("\nItems in Buffer %d \n",p);
        sem_post(mutex);

        printf("\nProducer %d released Semaphore Mutex \n",getpid());
        sem_post(full);

        printf("\nProducer %d released Semaphore Full \n",getpid());
        sleep(2/random());
    } //while
} //producer fn


//
// Consumer function
//
void consume()
{
    int i=0;
    while (1)
    {
        if(i>=strlen(input_string))
        {
            printf("\n Consumer %d exited \n",getpid());
            exit(1);
        }

        printf("\nConsumer %d trying to aquire Semaphore Full \n",getpid());
        sem_wait(full);

        printf("\nConsumer %d successfully aquired Semaphore Full \n",getpid()
);

        printf("\nConsumer %d trying to aquire Semaphore Mutex \n",getpid());
        sem_wait(mutex);

        printf("\nConsumer %d successfully aquired Semaphore Mutex\n",getpid()
);

        printf("\nConsumer %d Consumed Item [ %c ] \n",getpid(),buff[c]);
        buff[c]=' ';
        c++;

        printf("\nItems in Buffer %d \n",strlen(input_string));
        i++;
        sem_post(mutex);
        printf("\nConsumer %d released Semaphore Mutex \n",getpid());
```

```
        sem_post(empty);
        printf("\nConsumer %d released Semaphore Empty \n",getpid());
        sleep(1);
    } //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);


    // Initializing Semaphores Empty , Full & Mutex
    sem_init(empty,1,SIZE);
    sem_init(full,1,0);
    sem_init(mutex,1,1);


    printf("\n Main Process Started \n");

    printf("\n Enter the input string (20 characters MAX) : ");
    input_string=(char *)malloc(20);
    scanf("%s",input_string);
    printf("Entered string : %s",input_string);

    temp_pid=fork();

    if(temp_pid>0) //parent
    {
        produce();
    }
    else //child
    {
         consume();
    }
```

```
    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


/*
sowmya@LAPTOP-IJOMHH91:~$ gcc -o a pc.c -lpthread
sowmya@LAPTOP-IJOMHH91:~$ ./a

 Main Process Started

 Enter the input string (20 characters MAX) : sowmya
Entered string : sowmya
Entered string : sowmya
Producer 155 trying to aquire Semaphore Empty
Consumer 156 trying to aquire Semaphore Full

Producer 155 successfully aquired Semaphore Empty

Producer 155 trying to aquire Semaphore Mutex

Producer 155 successfully aquired Semaphore Mutex

Producer 155 Produced Item [ s ]

Items in Buffer 1

Producer 155 released Semaphore Mutex


Producer 155 released Semaphore Full
Consumer 156 successfully aquired Semaphore Full
```

```
Producer 155 trying to aquire Semaphore Empty
Consumer 156 trying to aquire Semaphore Mutex


Producer 155 successfully aquired Semaphore Empty
Consumer 156 successfully aquired Semaphore Mutex


Producer 155 trying to aquire Semaphore Mutex
Consumer 156 Consumed Item [ s ]

Items in Buffer 6


Consumer 156 released Semaphore Mutex
Producer 155 successfully aquired Semaphore Mutex


Consumer 156 released Semaphore Empty
Producer 155 Produced Item [ o ]

Items in Buffer 2

Producer 155 released Semaphore Mutex

Producer 155 released Semaphore Full

Producer 155 trying to aquire Semaphore Empty

Producer 155 successfully aquired Semaphore Empty

Producer 155 trying to aquire Semaphore Mutex

Producer 155 successfully aquired Semaphore Mutex

Producer 155 Produced Item [ w ]

Items in Buffer 3

Producer 155 released Semaphore Mutex

Producer 155 released Semaphore Full

Producer 155 trying to aquire Semaphore Empty

Producer 155 successfully aquired Semaphore Empty

Producer 155 trying to aquire Semaphore Mutex
```

```
Producer 155 successfully aquired Semaphore Mutex

Producer 155 Produced Item [ m ]

Items in Buffer 4

Producer 155 released Semaphore Mutex

Producer 155 released Semaphore Full

Producer 155 trying to aquire Semaphore Empty

Producer 155 successfully aquired Semaphore Empty

Producer 155 trying to aquire Semaphore Mutex

Producer 155 successfully aquired Semaphore Mutex

Producer 155 Produced Item [ y ]

Items in Buffer 5

Producer 155 released Semaphore Mutex

Producer 155 released Semaphore Full

Producer 155 trying to aquire Semaphore Empty

Producer 155 successfully aquired Semaphore Empty

Producer 155 trying to aquire Semaphore Mutex

Producer 155 successfully aquired Semaphore Mutex

Producer 155 Produced Item [ a ]

Items in Buffer 6

Producer 155 released Semaphore Mutex

Producer 155 released Semaphore Full

 Producer 155 exited

Consumer 156 trying to aquire Semaphore Full

Consumer 156 successfully aquired Semaphore Full
```

```
Consumer 156 trying to aquire Semaphore Mutex

Consumer 156 successfully aquired Semaphore Mutex

Consumer 156 Consumed Item [ o ]

Items in Buffer 6

Consumer 156 released Semaphore Mutex

Consumer 156 released Semaphore Empty

Consumer 156 trying to aquire Semaphore Full

Consumer 156 successfully aquired Semaphore Full

Consumer 156 trying to aquire Semaphore Mutex

Consumer 156 successfully aquired Semaphore Mutex

Consumer 156 Consumed Item [ w ]

Items in Buffer 6

Consumer 156 released Semaphore Mutex

Consumer 156 released Semaphore Empty

Consumer 156 trying to aquire Semaphore Full

Consumer 156 successfully aquired Semaphore Full

Consumer 156 trying to aquire Semaphore Mutex

Consumer 156 successfully aquired Semaphore Mutex

Consumer 156 Consumed Item [ m ]

Items in Buffer 6

Consumer 156 released Semaphore Mutex

Consumer 156 released Semaphore Empty

Consumer 156 trying to aquire Semaphore Full

Consumer 156 successfully aquired Semaphore Full
```

```
Consumer 156 trying to aquire Semaphore Mutex

Consumer 156 successfully aquired Semaphore Mutex

Consumer 156 Consumed Item [ y ]

Items in Buffer 6

Consumer 156 released Semaphore Mutex

Consumer 156 released Semaphore Empty

Consumer 156 trying to aquire Semaphore Full

Consumer 156 successfully aquired Semaphore Full

Consumer 156 trying to aquire Semaphore Mutex

Consumer 156 successfully aquired Semaphore Mutex

Consumer 156 Consumed Item [ a ]

Items in Buffer 6

Consumer 156 released Semaphore Mutex

Consumer 156 released Semaphore Empty

 Consumer 156 exited

 */
```

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

Server.c

```
#include<sys/ipc.h>
#define NULL 0
#include<sys/shm.h>
#include<sys/types.h>
#include<unistd.h>
#include<stdio.h>
```

```c
#include<stdlib.h>
#include<string.h>
#include<sys/wait.h>
#include<ctype.h>
#include<fcntl.h>
#include <semaphore.h>
#include <pthread.h>
#include <sys/sem.h>
#include <sys/wait.h>
#include <sys/errno.h>
#include<time.h>

extern int errno;
#define SIZE 20 /* size of the shared buffer*/
#define SHMPERM 0666

int shmid; /* id for shared memory bufer */
int empty_id;
int full_id;
int mutex_id;
int *buff;



sem_t *empty;
sem_t *full;
sem_t *mutex;


void produce()
{
    int i=0,j=1,n;
    int num[20];

    printf("Enter the no. of random no. to generate:");
    scanf("%d",&buff[0]);


    printf("Generating random numbers...\n");
    srand(time(0));

    for(i=1;i<=buff[0];i++)
    {
        num[i]=(rand()%(20-0+1));
    }


    while(1)
    {
```

```
        if(j>=i)
        {
            printf("\n Producer exited \n");
            exit(1);
        }

        printf("\nProducer trying to aquire Semaphore Empty \n");
        sem_wait(empty);

        printf("\nProducer successfully aquired Semaphore Empty \n");

        printf("\nProducer trying to aquire Semaphore Mutex \n");
        sem_wait(mutex);

        printf("\nProducer successfully aquired Semaphore Mutex \n");

        buff[j]=num[j];

        printf("\nProducer Produced Item [ %d ] \n",buff[j]);


        printf("\nItems in Buffer %d \n",j);
        j++;
        sem_post(mutex);

        printf("\nProducer released Semaphore Mutex \n");
        sem_post(full);

        printf("\nProducer released Semaphore Full \n");

    }
}

int main()
{

    if((shmid = shmget (213,SIZE, IPC_CREAT | IPC_EXCL | SHMPERM ))<0)
    {
            perror("shmid failed\n");
            exit(-1);
    }

     if((empty_id=shmget(301,sizeof(sem_t),IPC_CREAT|IPC_EXCL|SHMPERM))<0)
    {
            perror("empty_id failed\n");
            exit(-1);
    }

    if((full_id=shmget(920,sizeof(sem_t),IPC_CREAT|IPC_EXCL|SHMPERM))<0)
```

```
    {
            perror("full_id failed\n");
            exit(-1);
    }

    if((mutex_id=shmget(137,sizeof(sem_t),IPC_CREAT|IPC_EXCL|SHMPERM))<0)
    {
            perror("mutexid failed\n");
            exit(-1);
    }




    buff = shmat(shmid,NULL, 0 );
    empty = shmat(empty_id,(char *)0,0);
    full = shmat(full_id,(char *)0,0);
    mutex = shmat(mutex_id,(char *)0,0);


    // Initializing Semaphores Empty , Full & Mutex
    sem_init(empty,1,SIZE);
    sem_init(full,1,0);
    sem_init(mutex,1,1);

    printf("SERVER\n");
    produce();

    shmdt(buff);
    shmdt(empty);
    shmdt(full);
    shmdt(mutex);

    shmctl(shmid, 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 Server process exited \n\n");
    return(0);

}

/*
```

```
sowmya@LAPTOP-IJOMHH91:~$ gcc -o a server.c -lpthread
sowmya@LAPTOP-IJOMHH91:~$ ./a

SERVER
Enter the no. of random no. to generate:4
Generating random numbers...

Producer trying to aquire Semaphore Empty

Producer successfully aquired Semaphore Empty

Producer trying to aquire Semaphore Mutex

Producer successfully aquired Semaphore Mutex

Producer Produced Item [ 17 ]

Items in Buffer 1

Producer released Semaphore Mutex

Producer released Semaphore Full

Producer trying to aquire Semaphore Empty

Producer successfully aquired Semaphore Empty

Producer trying to aquire Semaphore Mutex

Producer successfully aquired Semaphore Mutex

Producer Produced Item [ 8 ]

Items in Buffer 2

Producer released Semaphore Mutex

Producer released Semaphore Full

Producer trying to aquire Semaphore Empty

Producer successfully aquired Semaphore Empty

Producer trying to aquire Semaphore Mutex

Producer successfully aquired Semaphore Mutex

Producer Produced Item [ 15 ]
```

```
Items in Buffer 3

Producer released Semaphore Mutex

Producer released Semaphore Full

Producer trying to aquire Semaphore Empty

Producer successfully aquired Semaphore Empty

Producer trying to aquire Semaphore Mutex

Producer successfully aquired Semaphore Mutex

Producer Produced Item [ 7 ]

Items in Buffer 4

Producer released Semaphore Mutex

Producer released Semaphore Full

 Producer exited
*/
```

Client.c

```c
#include<sys/ipc.h>
#define NULL 0
#include<sys/shm.h>
#include<sys/types.h>
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/wait.h>
#include<ctype.h>
#include<fcntl.h>
#include <semaphore.h>
#include <pthread.h>
#include <sys/sem.h>
#include <sys/wait.h>
#include <sys/errno.h>
#include<time.h>

extern int errno;
#define SIZE 20 /* size of the shared buffer*/
```

```c
#define SHMPERM 0666

int shmid; /* id for shared memory bufer */
int empty_id;
int full_id;
int mutex_id;
int *buff;



sem_t *empty;
sem_t *full;
sem_t *mutex;


void consume()
{
    int i=1,n;

    n=buff[0];

    while (1)
    {
        if(i>n)
        {
            printf("\n Consumer exited \n");
            exit(1);
        }

        printf("\nConsumer trying to aquire Semaphore Full \n");
        sem_wait(full);

        printf("\nConsumer successfully aquired Semaphore Full \n");

        printf("\nConsumer trying to aquire Semaphore Mutex \n");
        sem_wait(mutex);

        printf("\nConsumer successfully aquired Semaphore Mutex\n");

        printf("\nConsumer Consumed Item [ %d ] \n",buff[i]);
       // buff[i]=' ';
        i++;

        printf("\nItems in Buffer %d\n",n);

        sem_post(mutex);
        printf("\nConsumer released Semaphore Mutex \n");
        sem_post(empty);
        printf("\nConsumer released Semaphore Empty \n");
```

```
    }
}


int main()
{

    shmid = shmget (213,SIZE,0);
    empty_id=shmget(301,sizeof(sem_t),0);
    full_id=shmget(920,sizeof(sem_t),0);
    mutex_id=shmget(137,sizeof(sem_t),0);

    buff = shmat(shmid,NULL,0);
    empty = shmat(empty_id,(char *)0,0);
    full = shmat(full_id,(char *)0,0);
    mutex = shmat(mutex_id,(char *)0,0);


    printf("\n\nCLIENT\n\n");
    consume();

    shmdt(buff);
    shmdt(empty);
    shmdt(full);
    shmdt(mutex);

    shmctl(shmid, 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 Server process exited \n\n");
    return(0);

}
/*
sowmya@LAPTOP-IJOMHH91:~$ gcc -o b client.c -lpthread
sowmya@LAPTOP-IJOMHH91:~$ ./b


CLIENT

Consumer trying to aquire Semaphore Full
```

```
Consumer successfully aquired Semaphore Full

Consumer trying to aquire Semaphore Mutex

Consumer successfully aquired Semaphore Mutex

Consumer Consumed Item [ 17 ]

Items in Buffer 4

Consumer released Semaphore Mutex

Consumer released Semaphore Empty

Consumer trying to aquire Semaphore Full

Consumer successfully aquired Semaphore Full

Consumer trying to aquire Semaphore Mutex

Consumer successfully aquired Semaphore Mutex

Consumer Consumed Item [ 8 ]

Items in Buffer 4

Consumer released Semaphore Mutex

Consumer released Semaphore Empty

Consumer trying to aquire Semaphore Full

Consumer successfully aquired Semaphore Full

Consumer trying to aquire Semaphore Mutex

Consumer successfully aquired Semaphore Mutex

Consumer Consumed Item [ 15 ]

Items in Buffer 4

Consumer released Semaphore Mutex

Consumer released Semaphore Empty

Consumer trying to aquire Semaphore Full
```

```
Consumer successfully aquired Semaphore Full

Consumer trying to aquire Semaphore Mutex

Consumer successfully aquired Semaphore Mutex

Consumer Consumed Item [ 7 ]

Items in Buffer 4

Consumer released Semaphore Mutex

Consumer released Semaphore Empty

 Consumer exited
 */
```
