Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester IV

8 March 2020

UCS1411 - Operating Systems Laboratory

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

Objective:

Develop the following applications that uses semaphores concepts using to implement producer consumer problem.

Code:

Q1.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>
```

```c
#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>
#include<unistd.h>

extern int errno;

//Size of the shared buffer
#define SIZE 10
//Size of shared variable=1 byte
#define VARSIZE 1
//Maximum input limit
#define INPUTSIZE 20
//shared memory permissions
#define SHMPERM 0666

void my_wait(int *sem){
    while(*sem<=0);
    (*sem)--;
}

void my_signal(int *sem){
    (*sem)++;
}

// Producer function
void produce(char *input_string,int *empty,int *full,int *
    mutex,int *pctr,char *buff,int *cctr){
    int i=0;
    while (1){

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

        //Acquire semaphore empty
        printf("\nProducer %d trying to aquire Semaphore
    Empty \n",getpid());
```

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

        //Acquire semaphore mutex
        printf("\nProducer %d trying to aquire Semaphore Mutex \n",getpid());
        my_wait(mutex);
        printf("\nProducer %d successfully aquired Semaphore Mutex \n",getpid());

        //Critical Section
        buff[*pctr]=input_string[i];

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

        (*pctr)++;
        printf("\nItems in Buffer %d \n",*pctr-*cctr);

        //Release semaphore mutex
        my_signal(mutex);
        printf("\nProducer %d released Semaphore Mutex \n",getpid());

        //Release semaphore full
        my_signal(full);
        printf("\nProducer %d released Semaphore Full \n",getpid());

        //Remainder section
        sleep(2/random());
    }
}

// Consumer function
void consume(char *input_string,int *empty,int *full,int *mutex,int *cctr,char *buff,int *pctr){
    int i=0;
    while (1){
        if(i>=strlen(input_string)){
            printf("\n Consumer %d exited \n",getpid());
            exit(1);
        }

        //Acquire semaphore full
        printf("\nConsumer %d trying to aquire Semaphore Full \n",getpid());
        my_wait(full);
        printf("\nConsumer %d successfully aquired Semaphore Full \n",getpid());

        //Acquire semaphore mutex
        printf("\nConsumer %d trying to aquire Semaphore Mutex \n",getpid());
        my_wait(mutex);
        printf("\nConsumer %d successfully aquired Semaphore Mutex\n",getpid());

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

        printf("\nItems in Buffer %ld \n",strlen(buff)-*cctr);
        i++;

        //Release semaphore mutex
        my_signal(mutex);
        printf("\nConsumer %d released Semaphore Mutex \n",getpid());

        //Release semaphore empty
        my_signal(empty);
        printf("\nConsumer %d released Semaphore Empty \n",getpid());

        //Remainder Section
        sleep(1);
    }
}

int main(){
    //ID for shared memory bufer
    int shmid;

    /*

    IMPORTANT:
    The semaphores, integer pointers in this case are all
   initialised to NULL for now, instead of the respective
   values they should be taking.
    Initialisation is done later. Refer further notes.

    */

    //Semaphore empty
    int empty_id;
    int *empty=(int*)malloc(sizeof(int));

    //Semaphore full
    int full_id;
    int *full=(int*)malloc(sizeof(int));

    //Semaphore mutex
    int mutex_id;
    int *mutex=(int*)malloc(sizeof(int));

    //Buffer to read from/write onto the shared memory
    char *buff;

    //Input string
    char *input_string=(char*)malloc(20*sizeof(char));

    //Buffer counters in producer and consumer respectively
    int pctr=0,cctr=0;

    pid_t temp_pid;

    //Acquiring memory for shared memory
    shmid   = shmget(IPC_PRIVATE, SIZE, IPC_CREAT | IPC_EXCL
   | SHMPERM );
    full_id = shmget(IPC_PRIVATE, SIZE, IPC_CREAT | IPC_EXCL
   | SHMPERM);
    empty_id= shmget(IPC_PRIVATE, SIZE, IPC_CREAT | IPC_EXCL
   | SHMPERM);
    mutex_id= shmget(IPC_PRIVATE, SIZE, IPC_CREAT | IPC_EXCL
   | SHMPERM);

    //Attaching buffer to memory location
    buff  = shmat(shmid,(char*)0,0);
    full  = shmat(full_id,(int*)0,0);
```

```
 empty = shmat(empty_id,(int*)0,0);
 mutex =shmat(mutex_id,(int*)0,0);

 //Initialising the semaphores
 /*

 IMPORTANT:
 Note that the initialisation is done after the attachment
 to the memory locations.
 Doing these two operations in reverse, for some yet
unknown reason, leads to all the semaphores being
initialised to 0.

 */

 *empty=SIZE;
 *full=0;
 *mutex=1;

 printf("\n Main Process \n");
 printf("\nEnter string: ");scanf(" %s",input_string);
 printf("Entered string : %s",input_string);

 temp_pid=fork();
 if(temp_pid>0){
     produce(input_string,empty,full,mutex,&pctr,buff,&
cctr);
 }
 else{
     consume(input_string,empty,full,mutex,&cctr,buff,&
pctr);
 }

 //Detaching buffer from memory location
 shmdt(buff);
 shmdt(mutex);
 shmdt(empty);
 shmdt(full);

 //Destroying acquired location
 shmctl(shmid, IPC_RMID, NULL);
 shmctl(mutex_id, IPC_RMID, NULL);
 shmctl(empty_id, IPC_RMID, NULL);
 shmctl(full_id, IPC_RMID, NULL);

```

```

    printf("\n Main process exited \n\n");
    return(0);
}
```

Output:

```

 Main Process

Enter string: asdfgh
Entered string : asdfgh
Producer 3465 trying to aquire Semaphore Empty

Producer 3465 successfully aquired Semaphore Empty

Producer 3465 trying to aquire Semaphore Mutex

Producer 3465 successfully aquired Semaphore Mutex

Producer 3465 Produced Item [ a ]

Items in Buffer 1
Entered string : asdfgh

Producer 3465 released Semaphore Mutex
Consumer 3466 trying to aquire Semaphore Full

Producer 3465 released Semaphore Full

Consumer 3466 successfully aquired Semaphore Full

Consumer 3466 trying to aquire Semaphore Mutex

Consumer 3466 successfully aquired Semaphore Mutex

Consumer 3466 Consumed Item [ a ]


Producer 3465 trying to aquire Semaphore Empty
Items in Buffer 0


Producer 3465 successfully aquired Semaphore Empty
Consumer 3466 released Semaphore Mutex
```

```

Producer 3465 trying to aquire Semaphore Mutex


Consumer 3466 released Semaphore Empty
Producer 3465 successfully aquired Semaphore Mutex

Producer 3465 Produced Item [ s ]

Items in Buffer 2

Producer 3465 released Semaphore Mutex

Producer 3465 released Semaphore Full

Producer 3465 trying to aquire Semaphore Empty

Producer 3465 successfully aquired Semaphore Empty

Producer 3465 trying to aquire Semaphore Mutex

Producer 3465 successfully aquired Semaphore Mutex

Producer 3465 Produced Item [ d ]

Items in Buffer 3

Producer 3465 released Semaphore Mutex

Producer 3465 released Semaphore Full

Producer 3465 trying to aquire Semaphore Empty

Producer 3465 successfully aquired Semaphore Empty

Producer 3465 trying to aquire Semaphore Mutex

Producer 3465 successfully aquired Semaphore Mutex

Producer 3465 Produced Item [ f ]

Items in Buffer 4

Producer 3465 released Semaphore Mutex

Producer 3465 released Semaphore Full

Producer 3465 trying to aquire Semaphore Empty

Producer 3465 successfully aquired Semaphore Empty

Producer 3465 trying to aquire Semaphore Mutex

Producer 3465 successfully aquired Semaphore Mutex

Producer 3465 Produced Item [ g ]

Items in Buffer 5

Producer 3465 released Semaphore Mutex

Producer 3465 released Semaphore Full

Producer 3465 trying to aquire Semaphore Empty

Producer 3465 successfully aquired Semaphore Empty

Producer 3465 trying to aquire Semaphore Mutex

Producer 3465 successfully aquired Semaphore Mutex

Producer 3465 Produced Item [ h ]

Items in Buffer 6

Producer 3465 released Semaphore Mutex

Producer 3465 released Semaphore Full

 Producer 3465 exited

Consumer 3466 trying to aquire Semaphore Full

Consumer 3466 successfully aquired Semaphore Full

Consumer 3466 trying to aquire Semaphore Mutex

Consumer 3466 successfully aquired Semaphore Mutex

Consumer 3466 Consumed Item [ s ]
```

```

Items in Buffer 4

Consumer 3466 released Semaphore Mutex

Consumer 3466 released Semaphore Empty

Consumer 3466 trying to aquire Semaphore Full

Consumer 3466 successfully aquired Semaphore Full

Consumer 3466 trying to aquire Semaphore Mutex

Consumer 3466 successfully aquired Semaphore Mutex

Consumer 3466 Consumed Item [ d ]

Items in Buffer 3

Consumer 3466 released Semaphore Mutex

Consumer 3466 released Semaphore Empty

Consumer 3466 trying to aquire Semaphore Full

Consumer 3466 successfully aquired Semaphore Full

Consumer 3466 trying to aquire Semaphore Mutex

Consumer 3466 successfully aquired Semaphore Mutex

Consumer 3466 Consumed Item [ f ]

Items in Buffer 2

Consumer 3466 released Semaphore Mutex

Consumer 3466 released Semaphore Empty

Consumer 3466 trying to aquire Semaphore Full

Consumer 3466 successfully aquired Semaphore Full

Consumer 3466 trying to aquire Semaphore Mutex

Consumer 3466 successfully aquired Semaphore Mutex

Consumer 3466 Consumed Item [ g ]

Items in Buffer 1

Consumer 3466 released Semaphore Mutex

Consumer 3466 released Semaphore Empty

Consumer 3466 trying to aquire Semaphore Full

Consumer 3466 successfully aquired Semaphore Full

Consumer 3466 trying to aquire Semaphore Mutex

Consumer 3466 successfully aquired Semaphore Mutex

Consumer 3466 Consumed Item [ h ]

Items in Buffer 0

Consumer 3466 released Semaphore Mutex

Consumer 3466 released Semaphore Empty
```

Q2.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

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// For semaphore operations sem_init,sem_wait,sem_post
#include <semaphore.h>
#include <pthread.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <sys/wait.h>
#include <sys/errno.h>
```

```
#include <sys/types.h>
#include<unistd.h>
extern int errno;
#define SIZE 10 /* size of the shared buffer */
#define VARSIZE 1 /* size of shared variable = 1 byte */
#define INPUTSIZE 20
#define SHMPERM 0666 /* shared memory permissions */
int segid; /* ID for shared memory buffer */
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;
int main()
{
    int i = 0;
    pid_t temp_pid;
    segid = shmget(104, SIZE, IPC_CREAT | IPC_EXCL | SHMPERM
   );
    empty_id=shmget(105, sizeof(sem_t), IPC_CREAT | IPC_EXCL
   | SHMPERM);
    full_id=shmget(106, sizeof(sem_t), IPC_CREAT | IPC_EXCL |
    SHMPERM);
    mutex_id=shmget(107,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, 10);
    sem_init(full, 1, 0);
    sem_init(mutex, 1, 1);
    printf("\nProducer Process Started\n");
    while (i < 10)
    {
        int val = random()%10;
        printf("\nProducer %d trying to acquire Semaphore
   Empty\n", getpid());
        sem_wait(empty);
        printf("\nProducer %d successfully acquired Semaphore
 Empty\n", getpid());
        printf("\nProducer %d trying to acquire Semaphore
    Mutex\n", getpid());
        sem_wait(mutex);
        printf("\nProducer %d successfully acquired Semaphore
     Mutex\n", getpid());
        buff[p] = (char)(val + 48);
        printf("\nProducer %d Produced Item [%d]\n", getpid()
    , val);
        i++;
        p++;
        printf("\nItems produced: %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);
    }
    shmdt(buff);
    shmdt(empty);
    shmdt(full);
    shmdt(mutex);
    printf("\nProducer Process Ended\n");
    return(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
//For semaphore operations - sem_init, sem_wait, sem_post
#include <semaphore.h>
#include <pthread.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <sys/wait.h>
#include <sys/errno.h>
#include <sys/types.h>
#include<unistd.h>
extern int errno;

#define SIZE 10 /* size of the shared buffer */
#define VARSIZE 1 /* size of shared variable = 1 byte */
```

```c
#define INPUTSIZE 20
#define SHMPERM 0666 /* shared memory permissions */

int segid; /* ID for shared memory buffer */
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;
int main()
{
    int i = 0;
    pid_t temp_pid;
    segid = shmget (104, SIZE, IPC_EXCL | SHMPERM );
    empty_id = shmget(105, sizeof(sem_t), IPC_EXCL | SHMPERM)
   ;
    full_id = shmget(106, sizeof(sem_t), IPC_EXCL | SHMPERM);
    mutex_id=shmget(107, sizeof(sem_t), 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);
    printf("\nConsumer Process Started\n");
    while (i < 10)
    {
        printf("\nConsumer %d trying to acquire Semaphore
   Full\n", getpid());
        sem_wait(full);
        printf("\nConsumer %d successfully acquired Semaphore
    Full\n", getpid());
        printf("\nConsumer %d trying to acquire Semaphore
   Mutex\n", getpid());
        sem_wait(mutex);
        printf("\nConsumer %d successfully acquired Semaphore
    Mutex\n", getpid());
        printf("\nConsumer %d Consumed Item [%c]\n", getpid()
   , buff[c]);
        buff[c]=' ';
        c++;
        printf("\nItems consumed: %d\n", i+1);
        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);
    }
    shmdt(buff);
    shmdt(empty);
    shmdt(full);
    shmdt(mutex);
    sleep(10);
    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("\nConsumer Process Ended\n");
    return(0);
}
```

Output:

```
Producer Process Started

Producer 3733 trying to acquire Semaphore Empty

Producer 3733 successfully acquired Semaphore Empty

Producer 3733 trying to acquire Semaphore Mutex

Producer 3733 successfully acquired Semaphore Mutex

Producer 3733 Produced Item [3]

Items produced: 1

Producer 3733 released Semaphore Mutex

Producer 3733 released Semaphore Full

Producer 3733 trying to acquire Semaphore Empty

Producer 3733 successfully acquired Semaphore Empty

Producer 3733 trying to acquire Semaphore Mutex

Producer 3733 successfully acquired Semaphore Mutex

Producer 3733 Produced Item [6]

Items produced: 2

Producer 3733 released Semaphore Mutex

Producer 3733 released Semaphore Full

Producer 3733 trying to acquire Semaphore Empty

Producer 3733 successfully acquired Semaphore Empty

Producer 3733 trying to acquire Semaphore Mutex

Producer 3733 successfully acquired Semaphore Mutex

Producer 3733 Produced Item [7]

Items produced: 3

Producer 3733 released Semaphore Mutex

Producer 3733 released Semaphore Full

Producer 3733 trying to acquire Semaphore Empty

Producer 3733 successfully acquired Semaphore Empty

Producer 3733 trying to acquire Semaphore Mutex

Producer 3733 successfully acquired Semaphore Mutex

Producer 3733 Produced Item [5]

Items produced: 4

Producer 3733 released Semaphore Mutex

Producer 3733 released Semaphore Full

Producer 3733 trying to acquire Semaphore Empty

Producer 3733 successfully acquired Semaphore Empty

Producer 3733 trying to acquire Semaphore Mutex

Producer 3733 successfully acquired Semaphore Mutex

Producer 3733 Produced Item [3]

Items produced: 5

Producer 3733 released Semaphore Mutex

Producer 3733 released Semaphore Full

Producer 3733 trying to acquire Semaphore Empty

Producer 3733 successfully acquired Semaphore Empty

Producer 3733 trying to acquire Semaphore Mutex

Producer 3733 successfully acquired Semaphore Mutex

Producer 3733 Produced Item [5]

Items produced: 6

Producer 3733 released Semaphore Mutex

Producer 3733 released Semaphore Full

Producer 3733 trying to acquire Semaphore Empty

Producer 3733 successfully acquired Semaphore Empty

Producer 3733 trying to acquire Semaphore Mutex

Producer 3733 successfully acquired Semaphore Mutex

Producer 3733 Produced Item [6]

```

```
Items produced: 7

Producer 3733 released Semaphore Mutex

Producer 3733 released Semaphore Full

Producer 3733 trying to acquire Semaphore Empty

Producer 3733 successfully acquired Semaphore Empty

Producer 3733 trying to acquire Semaphore Mutex

Producer 3733 successfully acquired Semaphore Mutex

Producer 3733 Produced Item [2]

Items produced: 8

Producer 3733 released Semaphore Mutex

Producer 3733 released Semaphore Full

Producer 3733 trying to acquire Semaphore Empty

Producer 3733 successfully acquired Semaphore Empty

Producer 3733 trying to acquire Semaphore Mutex

Producer 3733 successfully acquired Semaphore Mutex

Producer 3733 Produced Item [9]

Items produced: 9

Producer 3733 released Semaphore Mutex

Producer 3733 released Semaphore Full

Producer 3733 trying to acquire Semaphore Empty

Producer 3733 successfully acquired Semaphore Empty

Producer 3733 trying to acquire Semaphore Mutex

Producer 3733 successfully acquired Semaphore Mutex

Producer 3733 Produced Item [1]

Items produced: 10

Producer 3733 released Semaphore Mutex

Producer 3733 released Semaphore Full

Producer Process Ended
```

```
Consumer Process Started

Consumer 3734 trying to acquire Semaphore Full

Consumer 3734 successfully acquired Semaphore Full

Consumer 3734 trying to acquire Semaphore Mutex

Consumer 3734 successfully acquired Semaphore Mutex

Consumer 3734 Consumed Item [3]

Items consumed: 1

Consumer 3734 released Semaphore Mutex

Consumer 3734 released Semaphore Empty

Consumer 3734 trying to acquire Semaphore Full

Consumer 3734 successfully acquired Semaphore Full

Consumer 3734 trying to acquire Semaphore Mutex

Consumer 3734 successfully acquired Semaphore Mutex

Consumer 3734 Consumed Item [6]

Items consumed: 2

Consumer 3734 released Semaphore Mutex

Consumer 3734 released Semaphore Empty

Consumer 3734 trying to acquire Semaphore Full

Consumer 3734 successfully acquired Semaphore Full

Consumer 3734 trying to acquire Semaphore Mutex

Consumer 3734 successfully acquired Semaphore Mutex

Consumer 3734 Consumed Item [7]

Items consumed: 3

Consumer 3734 released Semaphore Mutex

Consumer 3734 released Semaphore Empty

Consumer 3734 trying to acquire Semaphore Full

Consumer 3734 successfully acquired Semaphore Full

Consumer 3734 trying to acquire Semaphore Mutex

Consumer 3734 successfully acquired Semaphore Mutex

Consumer 3734 Consumed Item [5]

Items consumed: 4

Consumer 3734 released Semaphore Mutex

Consumer 3734 released Semaphore Empty

Consumer 3734 trying to acquire Semaphore Full

Consumer 3734 successfully acquired Semaphore Full

Consumer 3734 trying to acquire Semaphore Mutex

Consumer 3734 successfully acquired Semaphore Mutex

Consumer 3734 Consumed Item [3]

Items consumed: 5

Consumer 3734 released Semaphore Mutex

Consumer 3734 released Semaphore Empty

Consumer 3734 trying to acquire Semaphore Full

Consumer 3734 successfully acquired Semaphore Full

Consumer 3734 trying to acquire Semaphore Mutex

Consumer 3734 successfully acquired Semaphore Mutex

Consumer 3734 Consumed Item [5]

Items consumed: 6

Consumer 3734 released Semaphore Mutex

Consumer 3734 released Semaphore Empty

Consumer 3734 trying to acquire Semaphore Full

Consumer 3734 successfully acquired Semaphore Full

Consumer 3734 trying to acquire Semaphore Mutex

Consumer 3734 successfully acquired Semaphore Mutex

Consumer 3734 Consumed Item [6]

Items consumed: 7

Consumer 3734 released Semaphore Mutex

Consumer 3734 released Semaphore Empty

Consumer 3734 trying to acquire Semaphore Full

Consumer 3734 successfully acquired Semaphore Full

Consumer 3734 trying to acquire Semaphore Mutex

Consumer 3734 successfully acquired Semaphore Mutex

Consumer 3734 Consumed Item [2]

Items consumed: 8
```

```

Consumer 3734 released Semaphore Mutex

Consumer 3734 released Semaphore Empty

Consumer 3734 trying to acquire Semaphore Full

Consumer 3734 successfully acquired Semaphore Full

Consumer 3734 trying to acquire Semaphore Mutex

Consumer 3734 successfully acquired Semaphore Mutex

Consumer 3734 Consumed Item [9]

Items consumed: 9

Consumer 3734 released Semaphore Mutex

Consumer 3734 released Semaphore Empty

Consumer 3734 trying to acquire Semaphore Full

Consumer 3734 successfully acquired Semaphore Full

Consumer 3734 trying to acquire Semaphore Mutex

Consumer 3734 successfully acquired Semaphore Mutex

Consumer 3734 Consumed Item [1]

Items consumed: 10

Consumer 3734 released Semaphore Mutex

Consumer 3734 released Semaphore Empty

Consumer Process Ended
```