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

Assignment 1:

Algorithm:

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

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include
<semaphore.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
#define SHMPERM 0666
int segid;
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 produce(void) {
int i = 0; while (1)
{
if (i >=
strlen(input_string))
{
printf("\nProducer %d
exited!\n",
getpid());
wait(NULL);
return;
}
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] =
input_string[i];
printf("\nProducer %d
produced item [ %c ]
```

```
!\n", getpid(),
input_string[i]);

i++; p++;

printf("\nNumber of
items written 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); }

}

// Consumer function

//void consume() {

int consume(void){

int i = 0; while (1)
{

if (i >=
strlen(input_string))
{

printf("\n Consumer
%d exited \n",
getpid());

return;

}

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("\nNumber of
items read in Buffer:
%d \n", c);

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);

}

}

int main() {

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);

sem_init(empty, 1,
SIZE);

sem_init(full, 1, 0);

sem_init(mutex, 1,
1);

printf("\nMain
Process
Started...\n");

printf("\nEnter the
input string (20
characters MAX) : ");

input_string = (char
*) malloc(20);

scanf("%s",
input_string);

getchar();

printf("\nEntered
string : %s\n",
input_string);

temp_pid = fork();

if (temp_pid > 0) {

produce();
```

```
}
else {
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("\nMain
process exited \n");
return (0);
}
```

OUTPUT :

root@spl24:~/Desktop# gcc -g 61.c -pthread -o aout

ROOT@SPL24:~/DESKTOP# ./AOUT

MAIN PROCESS STARTED...

ENTER THE INPUT STRING (20 CHARACTERS MAX) : KRITHI

ENTERED STRING : KRITHI

PRODUCER 4881 TRYING TO ACQUIRE SEMAPHORE EMPTY...

PRODUCER 4881 SUCCESSFULLY ACQUIRED SEMAPHORE EMPTY!

CONSUMER 4883 TRYING TO ACQUIRE SEMAPHORE FULL...

PRODUCER 4881 TRYING TO ACQUIRE SEMAPHORE MUTEX...

PRODUCER 4881 SUCCESSFULLY ACQUIRED SEMAPHORE MUTEX!

PRODUCER 4881 PRODUCED ITEM [K] !

NUMBER OF ITEMS WRITTEN IN BUFFER: 1

PRODUCER 4881 RELEASED SEMAPHORE MUTEX!

PRODUCER 4881 RELEASED SEMAPHORE FULL!

CONSUMER 4883 SUCCESSFULLY ACQUIRED SEMAPHORE FULL!

CONSUMER 4883 TRYING TO ACQUIRE SEMAPHORE MUTEX...

CONSUMER 4883 SUCCESSFULLY ACQUIRED SEMAPHORE MUTEX!

CONSUMER 4883 CONSUMED ITEM [K]!

NUMBER OF ITEMS READ IN BUFFER: 1

CONSUMER 4883 RELEASED SEMAPHORE MUTEX!

CONSUMER 4883 RELEASED SEMAPHORE EMPTY!

CONSUMER 4883 TRYING TO ACQUIRE SEMAPHORE FULL...

PRODUCER 4881 TRYING TO ACQUIRE SEMAPHORE EMPTY...

PRODUCER 4881 SUCCESSFULLY ACQUIRED SEMAPHORE EMPTY!

PRODUCER 4881 TRYING TO ACQUIRE SEMAPHORE MUTEX...

PRODUCER 4881 SUCCESSFULLY ACQUIRED SEMAPHORE MUTEX!

PRODUCER 4881 PRODUCED ITEM [R] !

NUMBER OF ITEMS WRITTEN IN BUFFER: 2

PRODUCER 4881 RELEASED SEMAPHORE MUTEX!

PRODUCER 4881 RELEASED SEMAPHORE FULL!

CONSUMER 4883 SUCCESSFULLY ACQUIRED SEMAPHORE FULL!

CONSUMER 4883 TRYING TO ACQUIRE SEMAPHORE MUTEX...

CONSUMER 4883 SUCCESSFULLY ACQUIRED SEMAPHORE MUTEX!

CONSUMER 4883 CONSUMED ITEM [R]!

NUMBER OF ITEMS READ IN BUFFER: 2

CONSUMER 4883 RELEASED SEMAPHORE MUTEX!

CONSUMER 4883 RELEASED SEMAPHORE EMPTY!

CONSUMER 4883 TRYING TO ACQUIRE SEMAPHORE FULL...

PRODUCER 4881 TRYING TO ACQUIRE SEMAPHORE EMPTY...

PRODUCER 4881 SUCCESSFULLY ACQUIRED SEMAPHORE EMPTY!

PRODUCER 4881 TRYING TO ACQUIRE SEMAPHORE MUTEX...

PRODUCER 4881 SUCCESSFULLY ACQUIRED SEMAPHORE MUTEX!

PRODUCER 4881 PRODUCED ITEM [I] !

NUMBER OF ITEMS WRITTEN IN BUFFER: 3

PRODUCER 4881 RELEASED SEMAPHORE MUTEX!

PRODUCER 4881 RELEASED SEMAPHORE FULL!

CONSUMER 4883 SUCCESSFULLY ACQUIRED SEMAPHORE FULL!

CONSUMER 4883 TRYING TO ACQUIRE SEMAPHORE MUTEX...

CONSUMER 4883 SUCCESSFULLY ACQUIRED SEMAPHORE MUTEX!

CONSUMER 4883 CONSUMED ITEM [I]!

NUMBER OF ITEMS READ IN BUFFER: 3

CONSUMER 4883 RELEASED SEMAPHORE MUTEX!

CONSUMER 4883 RELEASED SEMAPHORE EMPTY!

CONSUMER 4883 TRYING TO ACQUIRE SEMAPHORE FULL...

PRODUCER 4881 TRYING TO ACQUIRE SEMAPHORE EMPTY...

PRODUCER 4881 SUCCESSFULLY ACQUIRED SEMAPHORE EMPTY!

PRODUCER 4881 TRYING TO ACQUIRE SEMAPHORE MUTEX...

PRODUCER 4881 SUCCESSFULLY ACQUIRED SEMAPHORE MUTEX!

PRODUCER 4881 PRODUCED ITEM [T] !

NUMBER OF ITEMS WRITTEN IN BUFFER: 4

PRODUCER 4881 RELEASED SEMAPHORE MUTEX!

PRODUCER 4881 RELEASED SEMAPHORE FULL!

CONSUMER 4883 SUCCESSFULLY ACQUIRED SEMAPHORE FULL!

CONSUMER 4883 TRYING TO ACQUIRE SEMAPHORE MUTEX...

CONSUMER 4883 SUCCESSFULLY ACQUIRED SEMAPHORE MUTEX!

CONSUMER 4883 CONSUMED ITEM [T]!

NUMBER OF ITEMS READ IN BUFFER: 4

CONSUMER 4883 RELEASED SEMAPHORE MUTEX!

CONSUMER 4883 RELEASED SEMAPHORE EMPTY!

CONSUMER 4883 TRYING TO ACQUIRE SEMAPHORE FULL...

PRODUCER 4881 TRYING TO ACQUIRE SEMAPHORE EMPTY...

PRODUCER 4881 SUCCESSFULLY ACQUIRED SEMAPHORE EMPTY!

PRODUCER 4881 TRYING TO ACQUIRE SEMAPHORE MUTEX...

PRODUCER 4881 SUCCESSFULLY ACQUIRED SEMAPHORE MUTEX!

PRODUCER 4881 PRODUCED ITEM [H] !

NUMBER OF ITEMS WRITTEN IN BUFFER: 5

PRODUCER 4881 RELEASED SEMAPHORE MUTEX!

PRODUCER 4881 RELEASED SEMAPHORE FULL!

CONSUMER 4883 SUCCESSFULLY ACQUIRED SEMAPHORE FULL!

CONSUMER 4883 TRYING TO ACQUIRE SEMAPHORE MUTEX...

CONSUMER 4883 SUCCESSFULLY ACQUIRED SEMAPHORE MUTEX!

CONSUMER 4883 CONSUMED ITEM [H]!

NUMBER OF ITEMS READ IN BUFFER: 5

CONSUMER 4883 RELEASED SEMAPHORE MUTEX!

CONSUMER 4883 RELEASED SEMAPHORE EMPTY!

CONSUMER 4883 TRYING TO ACQUIRE SEMAPHORE FULL...

PRODUCER 4881 TRYING TO ACQUIRE SEMAPHORE EMPTY...

PRODUCER 4881 SUCCESSFULLY ACQUIRED SEMAPHORE EMPTY!

PRODUCER 4881 TRYING TO ACQUIRE SEMAPHORE MUTEX...

PRODUCER 4881 SUCCESSFULLY ACQUIRED SEMAPHORE MUTEX!

PRODUCER 4881 PRODUCED ITEM [I] !

NUMBER OF ITEMS WRITTEN IN BUFFER: 6

PRODUCER 4881 RELEASED SEMAPHORE MUTEX!

CONSUMER 4883 SUCCESSFULLY ACQUIRED SEMAPHORE FULL!

CONSUMER 4883 TRYING TO ACQUIRE SEMAPHORE MUTEX...

CONSUMER 4883 SUCCESSFULLY ACQUIRED SEMAPHORE MUTEX!

CONSUMER 4883 CONSUMED ITEM [I]!

PRODUCER 4881 RELEASED SEMAPHORE FULL!

NUMBER OF ITEMS READ IN BUFFER: 6

CONSUMER 4883 RELEASED SEMAPHORE MUTEX!

CONSUMER 4883 RELEASED SEMAPHORE EMPTY!

CONSUMER 4883 EXITED

MAIN PROCESS EXITED

PRODUCER 4881 EXITED!

MAIN PROCESS 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

Algorithm :

1. Include all needed functions,and declare the variables.
2. Create a shared memory id in both the processes using shmget system call.
3. Also create the empty,full,mutex id using the shmget system call.

4. Generate the random numbers using the srand function.
5. Then store this in shared memory in the producer program.
6. Consumer programs should read them from shared memory and display them.
7. This is continued until the size of the given random number.
8. At last the both processes are detached from shared memory using the shmdt system call.
9. Data in shared memory is removed using the shmctl system call in both processes.

CODE:

server.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include
<semaphore.h>
#include <pthread.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <sys/wait.h>
#include<time.h>
#include
<sys/errno.h>
#include
<sys/types.h>


#define SIZE 3 /*
size of the shared
buffer */
#define SHMPERM 0666
/* shared memory
permissions */
int segid; /* ID for
shared memory buffer
*/
sem_t *empty;
sem_t *full;
sem_t *mutex;


char *buff; int i= 0,
N=5;
```

```
int main()
{
srand(time(0));

segid = shmget(100,
SIZE,
IPC_CREAT|0666);
buff = shmat(segid,
(char *)0, 0);

int
empty_id=shmget(101,s
izeof(sem_t),IPC_CREA
T|0666);
empty =
shmat(empty_id,(char
*)0,0);
sem_init(empty,1,SIZE
);

int
full_id=shmget(102,si
zeof(sem_t),IPC_CREAT
|0666);
full =
shmat(full_id,(char
*)0,0);
sem_init(full,1,0);

int
mutex_id=shmget(103,s
izeof(sem_t),IPC_CREA
T|0666);
mutex =
shmat(mutex_id,(char
*)0,0);
sem_init(mutex,1,1);

sleep(3);
while(1){
```

```
if(!N){
break;
}
printf("Server trying
to acquire semaphore
empty\n");
sem_wait(empty);
printf("Semaphore
empty acquired by
server\n");
printf("Server trying
to acquire semaphore
mutex\n");
sem_wait(mutex);
printf("Semaphore
mutex acquired by
server\n");
int val = rand()%10;
printf("Writing
%c\n",(char)(val+48))
;
buff[i] =
(char)(val+48);
N--;
i++;
sem_post(full);
printf("Server
released semaphore
full\n");
sem_post(mutex);
printf("Server
released semaphore
mutex\n\n");
sleep(1);
}

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);
}
```

Client.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include<semaphore.h>
#include <pthread.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <sys/wait.h>
#include<time.h>
#include
<sys/errno.h>
#include
<sys/types.h>
#define SIZE 3 /*
size of the shared
buffer */
#define SHMPERM 0666
/* shared memory
permissions */
int segid; /* ID for
shared memory buffer
*/
sem_t *empty;
sem_t *full;
sem_t *mutex;
char *buff;
int i= 0,N=5;
int main()
{
srand(time(0));
segid = shmget(100,
SIZE,
IPC_CREAT|0666);
```

```
buff = shmat(segid,
(char *)0, 0);
int
empty_id=shmget(101,s
izeof(sem_t),IPC_CREA
T|0666);
empty =
shmat(empty_id,(char
*)0,0);
int
full_id=shmget(102,si
zeof(sem_t),IPC_CREAT
|0666);
full =
shmat(full_id,(char
*)0,0);
int
mutex_id=shmget(103,s
izeof(sem_t),IPC_CREA
T|0666);
mutex =
shmat(mutex_id,(char
*)0,0);
while(1){
if(!N){
break;
}
printf("Client trying
to acquire semaphore
full\n");
sem_wait(full);
printf("Semaphore
full acquired by
client\n");
printf("Client trying
to acquire semaphore
mutex\n");
sem_wait(mutex);
```

```
printf("Semaphore
mutex acquired by
client\n");
printf("Reading
%c\n",buff[i]);
i++;
N--;
sem_post(empty);
printf("Client
released semaphore
empty\n");
sem_post(mutex);
printf("Client
released semaphore
mutex\n\n");
sleep(1);
}
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);
}
```

OUTPUT:

Server Terminal:

root@spl24:~/Desktop# gcc -g server6.c -pthread -o server

root@spl24:~/Desktop# ./server

Server trying to acquire semaphore empty
Semaphore empty acquired by server
Server trying to acquire semaphore mutex
Semaphore mutex acquired by server
Writing 8
Server released semaphore full
Server released semaphore mutex

Server trying to acquire semaphore empty
Semaphore empty acquired by server
Server trying to acquire semaphore mutex
Semaphore mutex acquired by server
Writing 4
Server released semaphore full
Server released semaphore mutex

Server trying to acquire semaphore empty
Semaphore empty acquired by server
Server trying to acquire semaphore mutex
Semaphore mutex acquired by server
Writing 3
Server released semaphore full
Server released semaphore mutex

Server trying to acquire semaphore empty
Semaphore empty acquired by server
Server trying to acquire semaphore mutex
Semaphore mutex acquired by server
Writing 5
Server released semaphore full
Server released semaphore mutex

Server trying to acquire semaphore empty
Semaphore empty acquired by server
Server trying to acquire semaphore mutex
Semaphore mutex acquired by server
Writing 1
Server released semaphore full
Server released semaphore mutex

Client Terminal:

root@spl24:~/Desktop# gcc -g client6.c -pthread -o client

```
root@spl24:~/Desktop# ./client
Client trying to acquire semaphore full
Semaphore full acquired by client
Client trying to acquire semaphore mutex
Semaphore mutex acquired by client
Reading 8
Client released semaphore empty
Client released semaphore mutex

Client trying to acquire semaphore full
Semaphore full acquired by client
Client trying to acquire semaphore mutex
Semaphore mutex acquired by client
Reading 4
Client released semaphore empty
Client released semaphore mutex

Client trying to acquire semaphore full
Semaphore full acquired by client
Client trying to acquire semaphore mutex
Semaphore mutex acquired by client
Reading 3
Client released semaphore empty
Client released semaphore mutex

Client trying to acquire semaphore full
Semaphore full acquired by client
Client trying to acquire semaphore mutex
Semaphore mutex acquired by client
Reading 5
Client released semaphore empty
Client released semaphore mutex

Client trying to acquire semaphore full
Semaphore full acquired by client
Client trying to acquire semaphore mutex
Semaphore mutex acquired by client
Reading 1
Client released semaphore empty
Client released semaphore mutex
```