Name : Rajvardhan Reddy
Reg No : 180905093
Sec : B
Roll No : 19

OS LAB – 7 : IPC 3 - Deadlock, Locking, Synchronization

Lab Exercises :

P1) Modify the above Producer-Consumer Program so that, a producer can produce at the most 10 items more than what the consumer has consumed.

Program :

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <unistd.h>

int buf[5], f, r;
sem_t mutex, full, empty;

void *produce(void *arg)
{
   for (int i = 0; i < 10; i++)
   {
      sem_wait(&empty);
      sem_wait(&mutex);

      printf("Produced item is %d\n", i);

      buf[(++r) % 10] = i;
      sleep(1);

      sem_post(&mutex);
```

```
        sem_post(&full);
        // printf("Full %u\n", full);
    }
}

void *consume(void *arg)
{
    int item;

    for (int i = 0; i < 10; i++)
    {
        sem_wait(&full);

        // printf("Full %u\n", full);

        sem_wait(&mutex);
        item = buf[(++f) % 10];

        printf("Consumed item is %d\n", item);
        sleep(1);

        sem_post(&mutex);
        sem_post(&empty);
    }
}
int main()
{
    pthread_t t1, t2;
    sem_init(&mutex, 0, 1);
    sem_init(&full, 0, 1);
    sem_init(&empty, 0, 10);
    pthread_create(&t1, NULL, produce, NULL);
    pthread_create(&t2, NULL, consume, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
}
```

Output :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LA
B/lab 7$ gcc -pthread lab7_p1.c -o p1
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LA
B/lab 7$ ./p1
Produced item is 0
Produced item is 1
Produced item is 2
Produced item is 3
Produced item is 4
Produced item is 5
Produced item is 6
Produced item is 7
Produced item is 8
Produced item is 9
Consumed item is 0
Consumed item is 1
Consumed item is 2
Consumed item is 3
Consumed item is 4
Consumed item is 5
Consumed item is 6
Consumed item is 3
Consumed item is 8
Consumed item is 9
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LA
B/lab 7$
```

P2) Write a C program for the first readers-writers problem using semaphores.

Program :

```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

sem_t wrt;
pthread_mutex_t mutex;
int cnt = 1;
int numreader = 0;

void *writer(void *wno)
{
    sem_wait(&wrt);
    cnt *= 2;
```

```
    printf("Writer %d modified 'cnt' to %d\n", (*((int *)wno)), cnt);
    sem_post(&wrt);
}

void *reader(void *rno)
{
    pthread_mutex_lock(&mutex);
    numreader++;

    if (numreader == 1)
        sem_wait(&wrt); // first reader will block the writer
    pthread_mutex_unlock(&mutex);

    // Reading Section, no locks

    printf("Reader %d: read 'cnt' as %d\n", *((int *)rno), cnt);

    // Reader acquire the lock before modifying numreader
    pthread_mutex_lock(&mutex);
    numreader--;

    if (numreader == 0)
        sem_post(&wrt); // If this is the last reader, it will wake up the writer.

    pthread_mutex_unlock(&mutex);
}

int main()
{
    pthread_t read[10], write[5];
    pthread_mutex_init(&mutex, NULL);

    sem_init(&wrt, 0, 1);

    int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; //used for numbering the
producer and consumer

    for (int i = 0; i < 10; i++)
        pthread_create(&read[i], NULL, reader, &a[i]);
```

```
    for (int i = 0; i < 5; i++)
        pthread_create(&write[i], NULL, writer, &a[i]);
    for (int i = 0; i < 10; i++)
        pthread_join(read[i], NULL);
    for (int i = 0; i < 5; i++)
        pthread_join(write[i], NULL);

    pthread_mutex_destroy(&mutex);
    sem_destroy(&wrt);
    return 0;
}
```

Output :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LA
B/lab 7$ gcc -pthread lab7_p2.c -o p2
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LA
B/lab 7$ ./p2
Reader 1: read 'cnt' as 1
Reader 2: read 'cnt' as 1
Reader 5: read 'cnt' as 1
Reader 6: read 'cnt' as 1
Reader 3: read 'cnt' as 1
Reader 4: read 'cnt' as 1
Reader 7: read 'cnt' as 1
Reader 8: read 'cnt' as 1
Reader 9: read 'cnt' as 1
Reader 10: read 'cnt' as 1
Writer 1 modified 'cnt' to 2
Writer 2 modified 'cnt' to 4
Writer 3 modified 'cnt' to 8
Writer 4 modified 'cnt' to 16
Writer 5 modified 'cnt' to 32
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LA
B/lab 7$
```

P3) Write a Code to access a shared resource which causes deadlock using improper use of semaphore.

Program :

```
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
```

```
#include <sys/sem.h>

#define PERMS 0660

int semId;
int initSem(int semId, int semNum, int initValue)
{
   return semctl(semId, semNum, SETVAL, initValue);
}

int P(int semId, int semNum)
{
   struct sembuf operationList[1];
   operationList[0].sem_num = semNum;
   operationList[0].sem_op = -1;
   operationList[0].sem_flg = 0;
   return semop(semId, operationList, 1);
}

int V(int semId, int semNum)
{
   struct sembuf operationList[1];
   operationList[0].sem_num = semNum;
   operationList[0].sem_op = 1;
   operationList[0].sem_flg = 0;
   return semop(semId, operationList, 1);
}

void* funcA(void* nothing)
{
   printf("Thread A try to lock 0...\n");
   P(semId, 0);
   printf("Thread A locked 0.\n");
   usleep(50*1000);
   printf("Thread A try to lock 1...\n");
   P(semId, 1);
   printf("Thread A locked 1.\n");
   V(semId, 0);
   V(semId, 1);
```

```
    return NULL;
}

void* funcB(void* nothing)
{
    printf("Thread B try to lock 1...\n");
    P(semId, 1);
    printf("Thread B locked 1.\n");
    usleep(5*1000);
    printf("Thread B try to lock 0...\n");
    P(semId, 0);
    printf("Thread B locked 0.\n");
    V(semId, 0);
    V(semId, 1);
    return NULL;
}

int main(int argc, char* argv[])
{
    int i;
    semId = semget(ftok(argv[0], 'A'), 2, IPC_CREAT | PERMS);
    initSem(semId, 0, 1);
    initSem(semId, 1, 1);
    pthread_t thread[2];
    pthread_create(&thread[0], NULL, funcA, NULL);
    pthread_create(&thread[1], NULL, funcB, NULL);

    for (i = 0 ; i < 2 ; i++)
    {
        pthread_join(thread[i], NULL);
    }

    printf("This is not printed in case of deadlock\n");
    semctl(semId, 0, IPC_RMID, 0);
    semctl(semId, 1, IPC_RMID, 0);
    return 0;
}
```

Output :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LA
B/lab 7$ gcc -o p3 lab7_p3.c -lpthread
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LA
B/lab 7$ ./p3
Thread A try to lock 0...
Thread A locked 0.
Thread B try to lock 1...
Thread B locked 1.
Thread B try to lock 0...
Thread A try to lock 1...
^C
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LA
B/lab 7$
```

P4) Write a program using semaphore to demonstrate the working of sleeping barber problem.

Program :

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <unistd.h>

sem_t customer, barber;
pthread_mutex_t seat;
int free1 = 10;
void *br(void *args)
{
   while (1)
   {
      sem_wait(&customer);
      pthread_mutex_lock(&seat);

      if (free1 < 10)
         free1++;
      sleep(2);
```

```
        printf("Cutting completed : free seats : %d\n", free1);
        sem_post(&barber);

        pthread_mutex_unlock(&seat);
    }
}

void *cr(void *args)
{
    while (1)
    {
        pthread_mutex_lock(&seat);

        if (free1 > 0)
        {
            free1--;

            printf("Customer waiting : free seats : %d\n", free1);
            sem_post(&customer);
            pthread_mutex_unlock(&seat);
            sem_wait(&barber);
        }
        else
            pthread_mutex_unlock(&seat);
    }
}

int main()
{
    pthread_t threads[2];
    sem_init(&barber, 0, 1);
    sem_init(&customer, 0, 1);
    pthread_mutex_init(&seat, 0);
    pthread_create(&threads[0], NULL, br, NULL);
    pthread_create(&threads[1], NULL, cr, NULL);
    pthread_join(threads[0], NULL);
    pthread_join(threads[1], NULL);
    sem_destroy(&barber);
    sem_destroy(&customer);
```

```
    pthread_mutex_destroy(&seat);
}
```

Output :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LA
B/lab 7$ gcc -pthread lab7_p4.c -o p4
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/OS_LA
B/lab 7$ ./p4
Cutting completed : free seats : 10
Customer waiting : free seats : 9
Customer waiting : free seats : 8
Customer waiting : free seats : 7
Cutting completed : free seats : 8
Cutting completed : free seats : 9
Cutting completed : free seats : 10
Customer waiting : free seats : 9
Customer waiting : free seats : 8
Customer waiting : free seats : 7
Cutting completed : free seats : 8
Cutting completed : free seats : 9
Cutting completed : free seats : 10
Customer waiting : free seats : 9
Customer waiting : free seats : 8
Customer waiting : free seats : 7
Cutting completed : free seats : 8
Cutting completed : free seats : 9
Cutting completed : free seats : 10
Customer waiting : free seats : 9
Customer waiting : free seats : 8
Customer waiting : free seats : 7
Cutting completed : free seats : 8
Cutting completed : free seats : 9
^C
```