Ex. No.: 8
Date: 2/4/25

PRODUCER CONSUMER USING SEMAPHORES

Aim: To write a program to implement solution to producer consumer problem using semaphores.

Algorithm:

1. Initialize semaphore empty, full and mutex.
2. Create two threads- producer thread and consumer thread.
3. Wait for target thread termination.
4. Call sem_wait on empty semaphore followed by mutex semaphore before entry into critical section.
5. Produce/Consume the item in critical section.
6. Call sem_post on mutex semaphore followed by full semaphore
7. before exiting critical section.
8. Allow the other thread to enter its critical section.
9. Terminate after looping ten times in producer and consumer Threads each.

Program Code:

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

#define MAX_ITEMS 10
int buffer = 0;

sem_t empty;
sem_t full;
sem_t mutex;

void* producer(void* arg){
  sem_wait(&empty);
  sem_wait(&mutex);
  if (buffer < MAX_ITEMS){
    buffer++;
    printf("Producer produced the item %d\n", buffer);
```

```
    sem_post(&mutex);
    sem_post(&empty);
    return NULL;
}
int main(){
    sem_init(&empty, 0, MAX_ITEMS);
    sem_init(&full, 0, 0);
    sem_init(&mutex, 0, 1);
    pthread_t producer_thread, consumer_thread;
    int choice;
    while(1){
        printf("\n1. Producer\n2. Consumer\n3. Exit\nEnter your choice: ");
        scanf("%d", &choice);
        if (choice == 1) {
            pthread_create(&producer_thread, NULL, producer, NULL);
            pthread_join(producer_thread, NULL);
        }
        else if(choice == 2){
            pthread_create(&consumer_thread, NULL, consumer, NULL);
            pthread_join(consumer_thread, NULL);
        }
        else if(choice == 3){
            printf("Exiting ....\n");
            break;
        }
        sem_destroy(&empty);
        sem_destroy(&full);
        sem_destroy(&mutex);
        return 0;
}
```

```
    } else {
        printf("Buffer is full\n");
    }

    sem_post(&mutex);
    sem_post(&full);

    return NULL;
}

void *consumer(void* arg){
    sem_wait(&full);
    sem_wait(&mutex);
    if(buffer > 0){
        printf("Consumer consuma item %d\n", buffer);
        buffer--;
    } else {
        printf("Buffer is empty!!\n");
    }

    sem_post(&mutex);
    sem_post(&empty);

    return NULL;
}

void *consumer(void* arg){
    sem_wait(&full);
    sem_wait(&mutex);
    if(buffer > 0){
        printf("Consumer consuma item %d\n", buffer);
        buffer--;
    } else {
        printf("Buffer is empty!!\n");
    }
}
```

Sample Output:

1. Producer
2.Consumer
3.Exit
Enter your choice:1
Producer produces the item 1
Enter your choice:2
Consumer consumes item
1 Enter your choice:2
Buffer is empty!!
Enter your choice:1
Producer produces the item 1
Enter your choice:1
Producer produces the item 2
Enter your choice:1
Producer produces the item 3
Enter your choice:1
Buffer is full!!
Enter your choice:3

Output:

1. Producer
2. Consumer
3. Exit

Enter your choice : 1

1. producer
2. Consumer
3. Exit

Enter your choice : 3

Exiting ... !

Result:

The program for producer Consumer problem using semaphore is implemented successfully.