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 <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#define BUF. 5
#define MAX 10
int buffer[BUF];
int in=0;
int out=0;
int proc =0;
int conc = 0;
sem_t mutex;
sem_t full;
sem_t empty;
~~void* producer(void* arg)~~
{
 +
void * produces (void * arg)
{
    int Item= 1;
    while (proc < MAX)
    {
        sem_wait(& empty);
        sem_wait (& mutex);
        buffer [in] = item;
        printf ("Produced: %d ", item);
        item++;
        in = (in + 1) % BUF;
        proc ++
        sem_post (&mutex);
        sem_post (& full);
    }
    pthread_exit (NULL);
}
 void* consumer (void * arg.)
 {
     while ( conc < MAX)
     {
         sem-wait (& full);
         sem- wait (& mutex);
         int item = buffer [out];
         printf (" Consumed : %d .", item);
         out = (out + 1) % BUF;
         conc ++
         sem_ post (& mutex );
         sem- post (& empty);
     }
     pthread_exit (NULL);
 }
 int main ( )
 {
     int choice;
     pthread_t prothr, conthr;
     sem_init (& mutex, 0, 1);
     sem_init (& full, 0, 0);
     sem_init (&empty, 0, BUF)
```

```
for(int i=0; i<10;i++)
{
  printf("\nMenu:\n");
  Printf ("1.Producer\n");
  printf ("2.Consumer\n");
  printf ("3. Exit\n");
  printf (" Enter your choice : ");
  Scanf ("%d", &choice);
  Switch(choice)
  {
    Case 1:
     if(proc < MAX)
       pthread_create(&prothr, NULL, producer, NULL);
       pthread_join(~~producer~~ prothr, NULL);
     else
       printf("Buffer is Full");
    break;
    Case 2:
     If (conc < MAX)
       pthread_create(&conthr, NULL, consumer, NULL);
       pthread_join( conthr, NULL);
     else
       printf ("Buffer is Empty");
    break;
    Case 3:
      pthread_exit (NULL);
      break;
   default:
     printf (" Invalid Choice");
 }
}
```

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
```

```
Menu :
1. Producer
2. Consumer
3. Exit
Enter your choice: 1
 Produced : 1
```

```
Menu :
1. Producer
2. Consumer
3. Exit
Enter your choice: 2
 consumed : 1
```

```
Menu :
1. Producer.
2. Consumer
3. Exit
Enter your choice : 3
```

Result:

Hence the Producer Consumer Problem using Semaphore has been implemented and executed