Ex. No.: 8 **PRODUCER CONSUMER USING SEMAPHORES**

Date: ***25.3.2025***

Aim:

To write a C program to implement a solution to the Producer-Consumer problem using semaphores.

Algorithm:

1. Initialize semaphores empty, full, and mutex.
2. Create two threads — one for the producer and another for the consumer.
3. Use pthread_create to create threads and pthread_join to wait for them to finish.
4. In each thread, use sem_wait() on empty and then on mutex before entering the critical section.
5. Produce or consume the item inside the critical section.
6. After the critical section, call sem_post() on mutex and then full (producer) or empty (consumer).
7. Let the threads alternate based on buffer availability.
8. Exit the loop after 10 iterations for both producer and consumer.
9. Terminate the program.

Program Code:

```
#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>


#define SIZE 5


int buffer[SIZE];

int in = 0, out = 0, item = 0;
```

```
sem_t empty, full, mutex;

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

        item++;
        buffer[in] = item;
        printf("Producer produces the item %d\n", item);
        in = (in + 1) % SIZE;

        sem_post(&mutex);
        sem_post(&full);
        sleep(1);
    }
    return NULL;
}

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

        int consumed_item = buffer[out];
        printf("Consumer consumes item %d\n", consumed_item);
        out = (out + 1) % SIZE;
```

```
        sem_post(&mutex);
        sem_post(&empty);
        sleep(1);
    }
    return NULL;
}

int main() {
    pthread_t prod, cons;

    sem_init(&empty, 0, SIZE);
    sem_init(&full, 0, 0);
    sem_init(&mutex, 0, 1);

    int choice;
    while (1) {
        printf("1. Producer\n2. Consumer\n3. Exit\nEnter your choice: ");
        scanf("%d", &choice);
        if (choice == 1) {
            pthread_create(&prod, NULL, producer, NULL);
            pthread_join(prod, NULL);
        } else if (choice == 2) {
            pthread_create(&cons, NULL, consumer, NULL);
            pthread_join(cons, NULL);
        } else {
            break;
        }
    }
```

```
    sem_destroy(&empty);
    sem_destroy(&full);
    sem_destroy(&mutex);

    return 0;
}
```

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

Result:

Thus, the Producer-Consumer problem was implemented successfully using semaphores in C, ensuring proper synchronization and avoiding race conditions.