praveenkumar.R
192324012

Experiment-29:Write a C program to simulate the solution of Classical Process Synchronization Problem

Aim:

To simulate the solution of the classical process synchronization problem, specifically using the Producer-Consumer Problem (also known as the Bounded Buffer Problem) where one process (producer) produces data and another process (consumer) consumes data. A shared buffer is used to hold data, and synchronization mechanisms like semaphores are employed to ensure safe access to the buffer.

Procedure:

1. Producer: The producer process produces items and places them into a shared buffer.
2. Consumer: The consumer process consumes items from the shared buffer.
3. Mutex and Semaphores: Use semaphores to synchronize access to the shared buffer. Specifically:
    - A mutex semaphore is used to provide mutual exclusion to the buffer.
    - Empty and Full semaphores are used to ensure that the consumer waits if the buffer is empty and the producer waits if the buffer is full.

Code:

```
#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>


#define BUFFER_SIZE 5

int buffer[BUFFER_SIZE];

int in = 0, out = 0;


sem_t empty, full, mutex;
```

```
void* producer(void* arg) {
    int item;
    while (1) {
        item = rand() % 100;
        sem_wait(&empty);
        sem_wait(&mutex);

        buffer[in] = item;
        printf("Produced: %d\n", item);
        in = (in + 1) % BUFFER_SIZE;

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

        sleep(1);
    }
}

void* consumer(void* arg) {
    int item;
    while (1) {
        sem_wait(&full);
        sem_wait(&mutex);

        item = buffer[out];
        printf("Consumed: %d\n", item);
```

```
        out = (out + 1) % BUFFER_SIZE;

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

        sleep(1);
    }
}

int main() {
    pthread_t prod, cons;

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

    pthread_create(&prod, NULL, producer, NULL);
    pthread_create(&cons, NULL, consumer, NULL);

    pthread_join(prod, NULL);
    pthread_join(cons, NULL);

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

    return 0;
```

}

Output:

Output

```
Produced: 45
Produced: 23
Produced: 78
Consumed: 45
Consumed: 23
Produced: 56
Consumed: 78
Produced: 34
Consumed: 56
```