**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:

