**VAISHNAVI E 231901059**

**Ex. No.: 8**

**Date:26.03.2025**

**PRODUCER CONSUMER USING SEMAPHORES**

**Aim:** To write a program to implement a 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.
7. Allow the other thread to enter its critical section.
8. Terminate after looping ten times in producer and consumer Threads each.

**Program Code:**

#include <stdio.h> #include <stdlib.h> int mutex = 1; // Initialize a mutex to 1 int full = 0; // Number of full slots as 0 int empty = 10, x = 0; // Number of empty slots as size of buffer void producer()

{

// Decrease mutex value by 1

--mutex;

// Increase the number of full

// slots by 1

++full;

// Decrease the number of empty

// slots by 1

--empty;

// Item produced

x++;

printf("\nProducer produces" "item %d", x);

// Increase mutex value by 1

++mutex;

}

// Function to consume an item and // remove it from buffer void consumer()

{

// Decrease mutex value by 1

--mutex;

// Decrease the number of full

// slots by 1

--full;

// Increase the number of empty

// slots by 1

++empty;

printf("\nConsumer consumes "

"item %d", x); x--;

// Increase mutex value by 1

++mutex;

}

// Driver Code int main()

{

int n, i;

printf("\n1. Press 1 for Producer" "\n2. Press 2 for Consumer"

"\n3. Press 3 for Exit");

// Using '#pragma omp parallel for' // can give wrong value due to // synchronization issues.

// 'critical' specifies that code is

// executed by only one thread at a

// time i.e., only one thread enters

// the critical section at a given time

#pragma omp critical

for (i = 1; i > 0; i++) {

printf("\nEnter your choice:");

scanf("%d", &n);

// Switch Cases switch (n) { case 1:

// If mutex is 1 and empty

// is non-zero, then it is // possible to produce if ((mutex == 1)

&& (empty != 0)) {

producer();

}

// Otherwise, print buffer // is full else {

printf("Buffer is full!");

} break;

case 2:

// If mutex is 1 and full

// is non-zero, then it is // possible to consume if ((mutex == 1) && (full != 0)) {

consumer();

}

// Otherwise, print Buffer // is empty

else {

printf("Buffer is empty!");

} break;

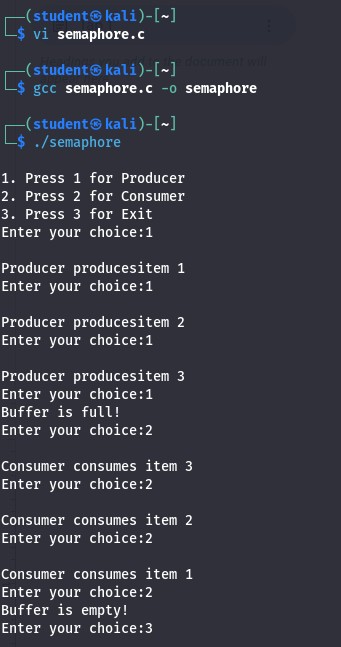
// Exit Condition case 3: exit(0); break;

}

}

}

**OUTPUT:**



**RESULT**:

Hence, producer consumer using semaphores has been executed successfully.