



**Program (Concurrent Execution):**

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#define BUFFER\_SIZE 5

#define MAX\_ITEMS 20 // Total items to produce

int buffer[BUFFER\_SIZE];

int count = 0;

int in = 0; // Producer index

int out = 0; // Consumer index

sem\_t empty, full;

pthread\_mutex\_t mutex;

void\* producer(void\* arg) {

for (int i = 1; i <= MAX\_ITEMS; i++) {

sem\_wait(&empty); // Wait for empty slot

pthread\_mutex\_lock(&mutex);

// Produce item

buffer[in] = i;

in = (in + 1) % BUFFER\_SIZE;

count++;

printf("Produced: %d\n", i);

pthread\_mutex\_unlock(&mutex);

sem\_post(&full); // Signal new item available

sleep(1); // Simulate production time

}

return NULL;

}

void\* consumer(void\* arg) {

for (int i = 1; i <= MAX\_ITEMS; i++) {

sem\_wait(&full); // Wait for available item

pthread\_mutex\_lock(&mutex);

// Consume item

int item = buffer[out];

out = (out + 1) % BUFFER\_SIZE;

count--;

printf("Consumed: %d\n", item);

pthread\_mutex\_unlock(&mutex);

sem\_post(&empty); // Signal empty slot available

sleep(2); // Simulate consumption time (slower than production)

}

return NULL;

}

int main() {

pthread\_t prod\_thread, cons\_thread;

// Initialize semaphores and mutex

sem\_init(&empty, 0, BUFFER\_SIZE);

sem\_init(&full, 0, 0);

pthread\_mutex\_init(&mutex, NULL);

// Create threads

pthread\_create(&prod\_thread, NULL, producer, NULL);

pthread\_create(&cons\_thread, NULL, consumer, NULL);

// Wait for threads to finish

pthread\_join(prod\_thread, NULL);

pthread\_join(cons\_thread, NULL);

// Cleanup

sem\_destroy(&empty);

sem\_destroy(&full);

pthread\_mutex\_destroy(&mutex);

printf("Finished producing and consuming %d items\n", MAX\_ITEMS);

return 0;

}

**Output:**

