# ASG 05

**Name:** Yatharth Thakare

**Roll No:** 51

**PRN:** 12111403

**Subject:** OS

**Implementation of Classical problems Producer Consumer using Threads and Semaphore.**

CODE:

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#include <stdlib.h>

#include <unistd.h>

sem\_t empty, full;

pthread\_mutex\_t mutex;

int buffer[5];

int count = 0;

void \*producer(void \*arg) {

long int num = (long int)arg;

sem\_wait(&empty);

pthread\_mutex\_lock(&mutex);

buffer[count] = num \* rand()%1000 +1;

printf("\n Producer %ld produced: %d", num + 1, buffer[count]);

printf("\n");

for(int i=0;i<5;i++){

printf("%d\n",buffer[i]);

}

count++;

sleep(1);

pthread\_mutex\_unlock(&mutex);

sem\_post(&full);

pthread\_exit(NULL);

}

void \*consumer(void \*arg) {

long int num = (long int)arg;

sem\_wait(&full);

pthread\_mutex\_lock(&mutex);

count--;

printf("\n Consumer %ld consumed: %d", num+ 1, buffer[count]);

buffer[count] = 0;

printf("\n");

for(int i=0;i<5;i++){

printf("%d\n",buffer[i]);

}

sleep(1);

pthread\_mutex\_unlock(&mutex);

sem\_post(&empty);

pthread\_exit(NULL);

}

int main() {

int nt = 0;

unsigned long int i;

printf("Enter total number of threads to create: ");

scanf("%d", &nt);

int num\_producer = nt / 2 + 1;

int num\_consumer = nt - num\_producer;

pthread\_t producers[num\_producer];

pthread\_t consumers[num\_consumer];

sem\_init(&empty, 0, 5);

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

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

srand(time(0));

for (i = 0; i < num\_producer; i++) {

pthread\_create(&producers[i], NULL, producer, (void \*)i);

}

for (i = 0; i < num\_consumer; i++) {

pthread\_create(&consumers[i], NULL, consumer, (void \*)i);

}

for (i = 0; i < num\_producer; i++) {

pthread\_join(producers[i], NULL);

}

for (i = 0; i < num\_consumer; i++) {

pthread\_join(consumers[i], NULL);

}

sem\_destroy(&empty);

sem\_destroy(&full);

pthread\_mutex\_destroy(&mutex);

return 0;

}

Output:

