**ASSIGNMENT 4**

**## Producer-Consumer**

#include <pthread.h>

#include <semaphore.h>

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#define MaxItems 5 // Maximum items a producer can produce or a consumer can consume

#define BufferSize 5 // Size of the buffer

#define MaxSize 5

sem\_t empty;

sem\_t full;

int in = 0;

int out = 0;

int buffer[BufferSize];

pthread\_mutex\_t mutex;

void \*producer(void \*pno)

{

int item;

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

item = rand(); // Produce an random item

sem\_wait(&empty);

pthread\_mutex\_lock(&mutex);

buffer[in] = item;

printf("Producer %d: produced %d\n", \*((int \*)pno),in);

in = (in+1)%BufferSize;

pthread\_mutex\_unlock(&mutex);

sem\_post(&full);

}

}

void \*consumer(void \*cno)

{

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

sem\_wait(&full);

pthread\_mutex\_lock(&mutex);

int item = buffer[out];

printf("Consumer %d: consumed %d\n",\*((int \*)cno), out);

out = (out+1)%BufferSize;

pthread\_mutex\_unlock(&mutex);

sem\_post(&empty);

}

}

int main()

{

int pn,cn;

pthread\_t pro[5],con[5];

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

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

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

printf("Enter No. of Producer: ");

scanf("%d",&pn);

printf("Enter No. of Consumer: ");

scanf("%d",&cn);

int cArr[MaxSize];

int pArr[MaxSize];

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

pArr[i]=i+1;

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

sleep(2);//optional by omkar

}

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

cArr[i]=i+1;

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

sleep(2);//optional by omkar

}

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

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

}

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

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

}

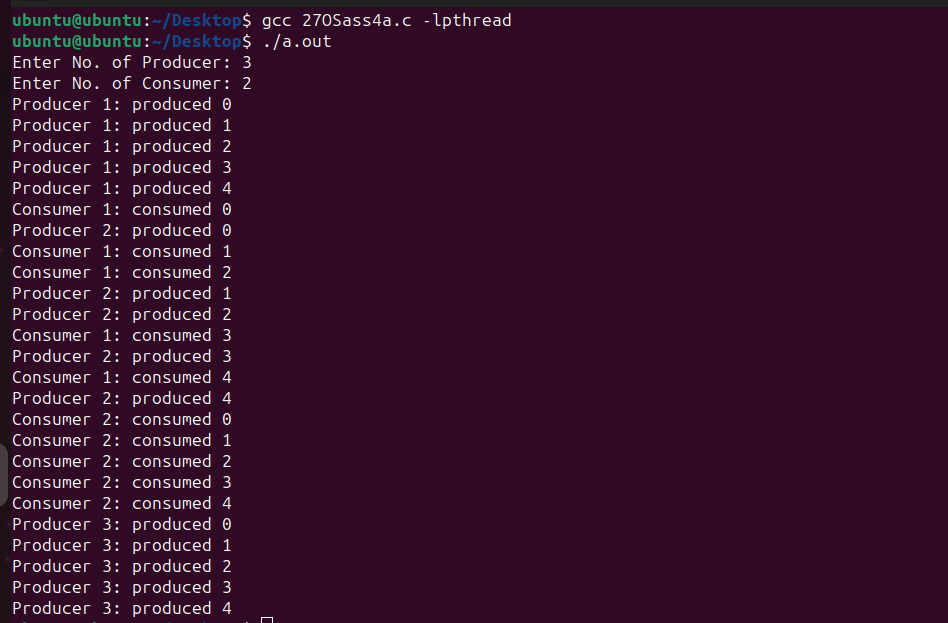
pthread\_mutex\_destroy(&mutex);

sem\_destroy(&empty);

sem\_destroy(&full);

return 0;

}



**## Reader-Writer**

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#define MaxSize 5

#define MaxItems 5

#define BufferSize 5

sem\_t wrt;

int in = 0;

int out = 0;

pthread\_mutex\_t mutex;

int buffer[BufferSize];

//int cnt = 1;

int numreader = 0;

void \*writer(void \*wno)

{

int item;

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

int item;

item = rand();

sem\_wait(&wrt);

buffer[in] = item;

printf("Writer %d writes count as %d\n",(\*((int \*)wno)),in);

in = (in+1)%BufferSize;

//cnt = cnt\*2;

//printf("Writer %d writes count as %d\n",(\*((int \*)wno)),cnt);

sem\_post(&wrt);

}

}

void \*reader(void \*rno)

{

// Reader acquire the lock before modifying numreader

pthread\_mutex\_lock(&mutex);

numreader++;

if(numreader == 1) {

sem\_wait(&wrt); // If this id the first reader, then it will block the writer

}

pthread\_mutex\_unlock(&mutex);

// Reading Section

//printf("Reader %d: read count as %d\n",\*((int \*)rno),cnt);

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

int item = buffer[out];

printf("Reader %d: read count as %d\n",\*((int \*)rno), out);

out = (out+1)%BufferSize;

}

// Reader acquire the lock before modifying numreader

pthread\_mutex\_lock(&mutex);

numreader--;

if(numreader == 0) {

sem\_post(&wrt); // If this is the last reader, it will wake up the writer.

}

pthread\_mutex\_unlock(&mutex);

}

int main()

{

int pn,cn;

pthread\_t read[10],write[5];

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

sem\_init(&wrt,0,1);

printf("Enter No. of Writer: ");

scanf("%d",&pn);

printf("Enter No. of Reader: ");

scanf("%d",&cn);

int cArr[MaxSize];

int pArr[MaxSize];

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

pArr[i]=i+1;

pthread\_create(&read[i], NULL, (void \*)reader, (void \*)&pArr[i]);

//sleep(2); BY KAUSTUBH NO DELETE

}

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

cArr[i]=i+1;

pthread\_create(&write[i], NULL, (void \*)writer, (void \*)&cArr[i]);

//sleep(2); BY KAUSTUBH NO DELETE

}

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

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

}

for(int i = 0; i < 10; i++) {

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

}

pthread\_mutex\_destroy(&mutex);

sem\_destroy(&wrt);

return 0;

}

