**Name: Atharva Salitri Division: CSAI B**

**Roll No.: 37 PRN: 12310120**

**Subject: OS Lab Assignment 3**

**Title: Synchronisation Problems**

1. **Producer – Consumer:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#define MaxItems 5

#define BufferSize 5

sem\_t empty;

sem\_t full;

sem\_t S;

int in = 0;

int out = 0;

int buffer[BufferSize];

void \*producer(void\* arg)

{

int item;

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

item = rand()%100;

sem\_wait(&empty);

sem\_wait(&S);

buffer[in] = item;

printf("Producer : Insert Item %d at %d\n",buffer[in],in);

in = (in+1)%BufferSize;

sem\_post(&S);

sem\_post(&full);

}

}

void \*consumers(void\* arg)

{

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

sem\_wait(&full);

sem\_wait(&S);

int item = buffer[out];

printf("Consumer: Remove Item %d from %d\n",item, out);

out = (out+1)%BufferSize;

sem\_post(&S);

sem\_post(&empty);

}

}

int main()

{

pthread\_t prod, cons;

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

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

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

pthread\_create(&prod, NULL, (void \*)producer, NULL);

pthread\_create(&cons, NULL, (void \*)consumers, NULL);

pthread\_join(prod, NULL);

pthread\_join(cons, NULL);

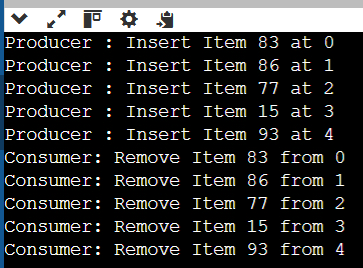
sem\_destroy(&empty);

sem\_destroy(&full);

sem\_destroy(&S);

return 0;

}



1. **Reader – Writer:**

#include <stdio.h>

#include <string.h>

#include <pthread.h>

#include <semaphore.h>

#include <stdlib.h>

#include <unistd.h>

#define BUFFER\_SIZE 16

int buffer[BUFFER\_SIZE];

sem\_t database, mutex;

int counter, readerCount;

pthread\_t readerThread[50], writerThread[50];

void init(){

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

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

counter = 0;

readerCount = 0;

}

void \*writer(void \*param){

sem\_wait(&database);

int item;

item = rand() % 5;

buffer[counter] = item;

printf("Data writen by the writer%d is %d\n", (\*(int \*)param),

buffer[counter]);

counter++;

sleep(1);

sem\_post(&database);

}

void \*reader(void \*param){

sem\_wait(&mutex);

readerCount++;

if (readerCount == 1){

sem\_wait(&database);

}

sem\_post(&mutex);

counter--;

printf("Data read by the reader%d is %d\n", (\*(int \*)param),buffer[counter]);

sleep(1);

sem\_wait(&mutex);

readerCount--;

if (readerCount == 0){

sem\_post(&database);

}

sem\_post(&mutex);

}

int main(){

init();

int no\_of\_writers, no\_of\_readers;

printf("Enter number of readers: ");

scanf("%d", &no\_of\_readers);

printf("Enter number of writers: ");

scanf("%d", &no\_of\_writers);

int i;

for (i = 0; i < no\_of\_writers; i++){

pthread\_create(&writerThread[i], NULL, writer, &i);

}

for (i = 0; i < no\_of\_readers; i++){

pthread\_create(&readerThread[i], NULL, reader, &i);

}

for (i = 0; i < no\_of\_writers; i++){

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

}

for (i = 0; i < no\_of\_readers; i++){

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

}

return 0;

}

1. **Dining – Philosopher:**

#include <stdio.h>

int p[5];

int ch[5];

void signal(int z)

{

int j = (z + 1) % 5;

p[z] = 0;

ch[z] = 0;

ch[j] = 0;

}

void wait(int y)

{

int r = (y + 1) % 5;

if ((ch[y] == 0) && (ch[r] == 0))

{

p[y] = 1;

ch[y] = 1;

ch[r] = 1;

}

else if (p[y] == 1)

{

int w;

printf("Do you want Philosopher %d to stop eating", y);

scanf("%d", &w);

if (w == 1)

signal(y);

}

else

{

printf("chopstick %d %d are busy \n", y, r);

printf("Philosopher %d has to wait \n", y);

}

}

int main()

{

int i, u;

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

{

p[i] = 0;

ch[i] = 0;

}

do

{

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

{

if (p[i] == 0)

{

printf("Philosopher %d is thinking \n", i);

printf("\n");

}

else

{

printf("Philosopher %d is eating \n", i);

printf("\n");

}

}

int s;

printf("Which Philosopher wants to eat \n");

scanf("%d", &s);

wait(s);

printf("Do you want to continue Press 1 \n");

scanf("%d", &u);

} while (u == 1);

return 0;

}

