NATIONAL INSTITUTE OF TECHNOLOGY
KARNATAKA

OPERATING SYSTEM LAB

EXPERIMENT : 4

Name : Shivam Yadav
Roll No : 211CS257
Sec : S2
Code : CS257

Q1) Producer consumer Problem

```
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>


#define MaxItems 4 // Maximum items a producer can produce or a consumer can consume
#define BufferSize 4 // Size of the buffer

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: Insert Item %d at %d\n", *((int *)pno),buffer[in],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: Remove Item %d from %d\n",*((int *)cno),item, out);
      out = (out+1)%BufferSize;
      pthread_mutex_unlock(&mutex);
      sem_post(&empty);
   }
}

int main()
{

   pthread_t pro[4],con[4];
   pthread_mutex_init(&mutex, NULL);
   sem_init(&empty,0,BufferSize);
   sem_init(&full,0,0);

   int a[4] = {1,2,3,4}; //Just used for numbering the producer and consumer

   for(int i = 0; i < 4; i++) {
      pthread_create(&pro[i], NULL, (void *)producer, (void *)&a[i]);
   }
   for(int i = 0; i < 4; i++) {
      pthread_create(&con[i], NULL, (void *)consumer, (void *)&a[i]);
   }

   for(int i = 0; i < 4; i++) {
      pthread_join(pro[i], NULL);
   }
   for(int i = 0; i < 4; i++) {
      pthread_join(con[i], NULL);
   }
```

```
    pthread_mutex_destroy(&mutex);
    sem_destroy(&empty);
    sem_destroy(&full);

    return 0;

}
```

output :

```
Producer 2: Insert Item 1804289383 at 0
Consumer 1: Remove Item 1804289383 from 0
Producer 3: Insert Item 846930886 at 1
Consumer 3: Remove Item 846930886 from 1
Producer 4: Insert Item 1681692777 at 2
Producer 3: Insert Item 424238335 at 3
Consumer 2: Remove Item 1681692777 from 2
Producer 2: Insert Item 1957747793 at 0
Producer 4: Insert Item 719885386 at 1
Producer 1: Insert Item 1714636915 at 2
Consumer 3: Remove Item 424238335 from 3
```

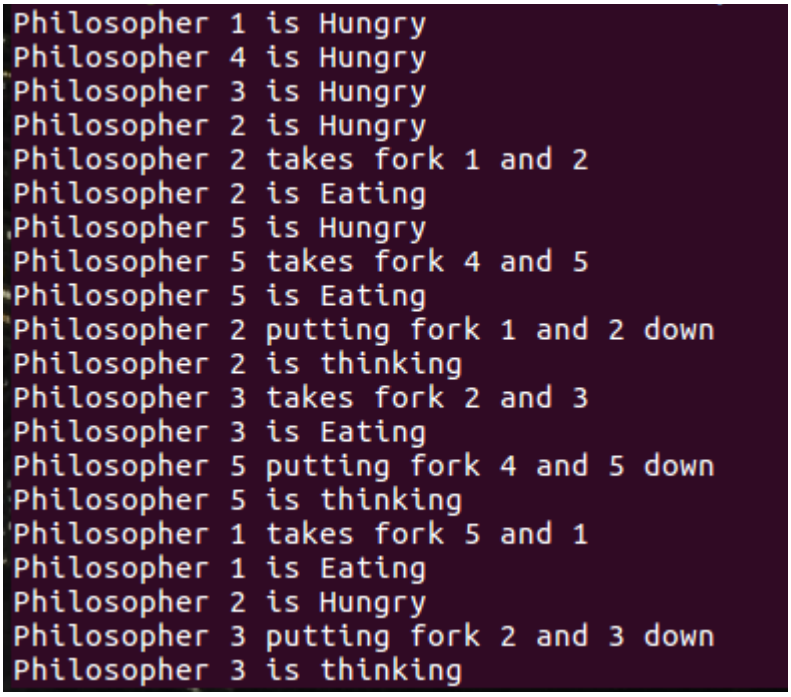
Q2) Dining Philoshpher Problem

```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <unistd.h>
#define N 5
#define THINKING 2
#define HUNGRY 1
#define EATING 0
#define LEFT (phnum + 4) % N
#define RIGHT (phnum + 1) % N

int state[N];
int phil[N] = { 0, 1, 2, 3, 4 };

sem_t mutex;
sem_t S[N];

void test(int phnum)
{
```

```
    if (state[phnum] == HUNGRY
        && state[LEFT] != EATING
        && state[RIGHT] != EATING) {
        // state that eating
        state[phnum] = EATING;

        sleep(2);

        printf("Philosopher %d takes fork %d and %d\n",
                    phnum + 1, LEFT + 1, phnum + 1);

        printf("Philosopher %d is Eating\n", phnum + 1);

        // sem_post(&S[phnum]) has no effect
        // during takefork
        // used to wake up hungry philosophers
        // during putfork
        sem_post(&S[phnum]);
    }
}

// take up chopsticks
void take_fork(int phnum)
{

    sem_wait(&mutex);

    // state that hungry
    state[phnum] = HUNGRY;

    printf("Philosopher %d is Hungry\n", phnum + 1);

    // eat if neighbours are not eating
    test(phnum);

    sem_post(&mutex);

    // if unable to eat wait to be signalled
    sem_wait(&S[phnum]);

    sleep(1);
}

// put down chopsticks
void put_fork(int phnum)
{

    sem_wait(&mutex);

    // state that thinking
    state[phnum] = THINKING;
```

```
    printf("Philosopher %d putting fork %d and %d down\n",
           phnum + 1, LEFT + 1, phnum + 1);
    printf("Philosopher %d is thinking\n", phnum + 1);

    test(LEFT);
    test(RIGHT);

    sem_post(&mutex);
}

void* philosopher(void* num)
{

    while (1) {

        int* i = num;

        sleep(1);

        take_fork(*i);

        sleep(0);

        put_fork(*i);
    }
}

int main()
{

    int i;
    pthread_t thread_id[N];

    // initialize the semaphores
    sem_init(&mutex, 0, 1);

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

        sem_init(&S[i], 0, 0);

    for (i = 0; i < N; i++) {

        // create philosopher processes
        pthread_create(&thread_id[i], NULL,
                       philosopher, &phil[i]);


    }

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

        pthread_join(thread_id[i], NULL);}
```

output :

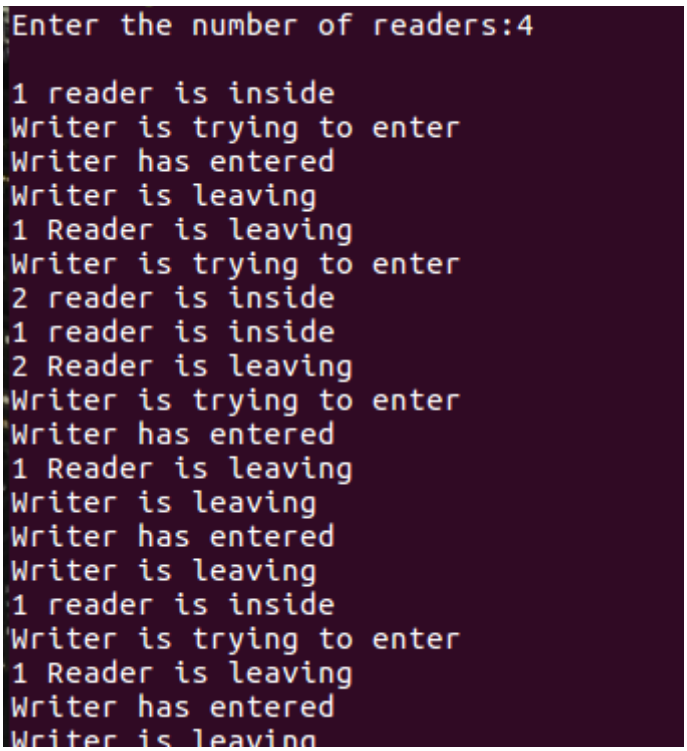


Q3) Reader writer Problem

```
#include<semaphore.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
sem_t x,y;
pthread_t tid;
pthread_t writerthreads[100],readerthreads[100];
int readercount = 0;


void *reader(void* p)
{
```

```
    sem_wait(&x);
    readercount++;
    if(readercount==1)
        sem_wait(&y);
    sem_post(&x);
    printf("%d reader is inside\n",readercount);
    usleep(3);
    sem_wait(&x);
    readercount--;
    if(readercount==0)
    {
        sem_post(&y);
    }
    sem_post(&x);
    printf("%d Reader is leaving\n",readercount+1);
    return NULL;
}

void *writer(void* p)
{
    printf("Writer is trying to enter\n");
    sem_wait(&y);
    printf("Writer has entered\n");
    sem_post(&y);
    printf("Writer is leaving\n");
    return NULL;
}

int main()
{
    int n2,i;
    printf("Enter the number of readers:");
    scanf("%d",&n2);
    printf("\n");
    int n1[n2];
    sem_init(&x,0,1);
    sem_init(&y,0,1);
    for(i=0;i<n2;i++)
    {
        pthread_create(&writerthreads[i],NULL,reader,NULL);
        pthread_create(&readerthreads[i],NULL,writer,NULL);
    }
    for(i=0;i<n2;i++)
    {
        pthread_join(writerthreads[i],NULL);
        pthread_join(readerthreads[i],NULL);
    }

}
```

output :


Enter the number of readers:4
1 reader is inside
Writer is trying to enter
Writer has entered
Writer is leaving
1 Reader is leaving
Writer is trying to enter
2 reader is inside
1 reader is inside
2 Reader is leaving
Writer is trying to enter
Writer has entered
1 Reader is leaving
Writer is leaving
Writer has entered
Writer is leaving
1 reader is inside
Writer is trying to enter
1 Reader is leaving
Writer has entered
Writer is leaving