ITE2002-Operating System (Lab)

WINTER SEM 20-21

Assessment-3

Name :Pravin G

Reg No :19BIT0393

Slot :L41+L42

Question 1

Write a program to implement the producer –consumer problem using semaphores

Code:-

```
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>
#define MaxItems 5
#define BufferSize 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,i;
    for(i = 0; i < MaxItems; i++) {
        item = rand();
        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)
{
int i;
    for(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()
{
    int i;
    pthread_t pro[5],con[5];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&empty,0,BufferSize);
    sem_init(&full,0,0);
    int a[5] = {1,2,3,4,5};
    for(i = 0; i < 5; i++) {
        pthread_create(&pro[i], NULL, (void *)producer,
                                            (void *)&a[i]);
    }
    for(i = 0; i < 5; i++) {
        pthread_create(&con[i], NULL, (void *)consumer,
                                            (void *)&a[i]);
    }
    for(i = 0; i < 5; i++) {
        pthread_join(pro[i], NULL);
    }
    for(i = 0; i < 5; i++) {
        pthread_join(con[i], NULL);
    }
    pthread_mutex_destroy(&mutex);
    sem_destroy(&empty);
    sem_destroy(&full);
    return 0;

}
```

Output:-

```
PRAVIN@DESKTOP-B2LB8FB ~/oslab/da3
$ gcc producerconsumer.c -o producerconsumer.exe

PRAVIN@DESKTOP-B2LB8FB ~/oslab/da3
$ ./producerconsumer.exe
Producer 1: Insert Item 1481765933 at 0
Producer 2: Insert Item 1481765933 at 1
Producer 3: Insert Item 1481765933 at 2
Producer 1: Insert Item 1085377743 at 3
Producer 4: Insert Item 1481765933 at 4
Consumer 1: Remove Item 1481765933 from 0
Consumer 2: Remove Item 1481765933 from 1
Consumer 3: Remove Item 1481765933 from 2
Consumer 4: Remove Item 1085377743 from 3
Consumer 5: Remove Item 1481765933 from 4
Producer 5: Insert Item 1481765933 at 0
Producer 2: Insert Item 1085377743 at 1
Producer 3: Insert Item 1085377743 at 2
Producer 1: Insert Item 1270216262 at 3
Producer 4: Insert Item 1085377743 at 4
Consumer 1: Remove Item 1481765933 from 0
Consumer 2: Remove Item 1085377743 from 1
Consumer 3: Remove Item 1085377743 from 2
Consumer 4: Remove Item 1270216262 from 3
Consumer 5: Remove Item 1085377743 from 4
Producer 5: Insert Item 1085377743 at 0
Producer 2: Insert Item 1270216262 at 1
Producer 3: Insert Item 1270216262 at 2
```

```
Consumer 3: Remove Item 1270216262 from 2
Consumer 4: Remove Item 1191391529 from 3
Consumer 5: Remove Item 1270216262 from 4
Producer 5: Insert Item 1270216262 at 0
Producer 2: Insert Item 1191391529 at 1
Producer 3: Insert Item 1191391529 at 2
Producer 1: Insert Item 812669700 at 3
Producer 4: Insert Item 1191391529 at 4
Consumer 1: Remove Item 1270216262 from 0
Consumer 2: Remove Item 1191391529 from 1
Consumer 3: Remove Item 1191391529 from 2
Consumer 4: Remove Item 812669700 from 3
Consumer 5: Remove Item 1191391529 from 4
Producer 5: Insert Item 1191391529 at 0
Producer 2: Insert Item 812669700 at 1
Producer 3: Insert Item 812669700 at 2
Producer 4: Insert Item 812669700 at 3
Producer 5: Insert Item 812669700 at 4
Consumer 1: Remove Item 1191391529 from 0
Consumer 2: Remove Item 812669700 from 1
Consumer 3: Remove Item 812669700 from 2
Consumer 4: Remove Item 812669700 from 3
Consumer 5: Remove Item 812669700 from 4
```

Question 2

Write a Program to implement the solution for dining philosopher's problem

Code:-

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<semaphore.h>
#include<unistd.h>
sem_t room;
sem_t chopstick[5];
void * philosopher(void *);
void eat(int);
int main()
{
    int i,a[5];
    pthread_t tid[5];
    sem_init(&room,0,4);
    for(i=0;i<5;i++)
        sem_init(&chopstick[i],0,1);

    for(i=0;i<5;i++){
        a[i]=i;
        pthread_create(&tid[i],NULL,philosopher,
                                            (void *)&a[i]);
    }
    for(i=0;i<5;i++)
        pthread_join(tid[i],NULL);
}

void * philosopher(void * num)
{
    int phil=*(int *)num;

    sem_wait(&room);
    printf("\nPhilosopher %d has entered room",phil);
```

```
    sem_wait(&chopstick[phil]);
    sem_wait(&chopstick[(phil+1)%5]);

    eat(phil);
    sleep(2);
    printf("\nPhilosopher %d has finished eating",phil);

    sem_post(&chopstick[(phil+1)%5]);
    sem_post(&chopstick[phil]);
    sem_post(&room);
}
void eat(int phil)
{
    printf("\nPhilosopher %d is eating",phil);
}
```

Output:-

```
PRAVIN@DESKTOP-B2LB8FB ~/oslab/da3
$ gcc diningphilosopher.c -o diningphilosopher.exe

PRAVIN@DESKTOP-B2LB8FB ~/oslab/da3
$ ./diningphilosopher.exe

Philosopher 0 has entered room
Philosopher 1 has entered room
Philosopher 2 has entered room
Philosopher 0 is eating
Philosopher 3 has entered room
Philosopher 2 is eating
Philosopher 0 has finished eating
Philosopher 2 has finished eating
Philosopher 4 has entered room
Philosopher 3 is eating
Philosopher 1 is eating
Philosopher 3 has finished eating
Philosopher 1 has finished eating
Philosopher 4 is eating
Philosopher 4 has finished eating
```

Question 3

Write a program to implement the solution for Readers Writers Problem using semaphores.

Code:-

```c
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
sem_t wrt;
pthread_mutex_t mutex;
int cnt = 1;
int numreader = 0;
void *writer(void *wno)
{
    sem_wait(&wrt);
    cnt = cnt*2;
    printf("Writer %d modified cnt to %d\n",
                                    (*((int *)wno)),cnt);
    sem_post(&wrt);
}
void *reader(void *rno)
{
    pthread_mutex_lock(&mutex);
    numreader++;
    if(numreader == 1)
    {
        sem_wait(&wrt);
    }
    pthread_mutex_unlock(&mutex);
    printf("Reader %d: read cnt as %d\n",*((int *)rno),cnt);
    pthread_mutex_lock(&mutex);
    numreader--;
    if(numreader == 0)
        sem_post(&wrt);
    pthread_mutex_unlock(&mutex);
}
```

```c
int main()
{
    pthread_t read[10],write[5];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&wrt,0,1);
    int i,j;
    int a[10] = {1,2,3,4,5,6,7,8,9,10};
    for(i = 0; i < 10; i++)
        pthread_create(&read[i], NULL, (void *)reader,
                                            (void *)&a[i]);
    for(i = 0; i < 5; i++)
        pthread_create(&write[i], NULL, (void *)writer,
                                            (void *)&a[i]);
    for(i = 0; i < 10; i++)
        pthread_join(read[i], NULL);
    for(i = 0; i < 5; i++)
        pthread_join(write[i], NULL);
    pthread_mutex_destroy(&mutex);
    sem_destroy(&wrt);
    return 0;
}
```

Output:-

```
PRAVIN@DESKTOP-B2LB8FB ~/oslab/da3
$ gcc readerwriter.c -o readerwriter.exe

PRAVIN@DESKTOP-B2LB8FB ~/oslab/da3
$ ./readerwriter.exe
Reader 1: read cnt as 1
Reader 2: read cnt as 1
Reader 3: read cnt as 1
Reader 4: read cnt as 1
Reader 5: read cnt as 1
Reader 6: read cnt as 1
Reader 7: read cnt as 1
Reader 8: read cnt as 1
Reader 9: read cnt as 1
Reader 10: read cnt as 1
Writer 1 modified cnt to 2
Writer 2 modified cnt to 4
Writer 3 modified cnt to 8
Writer 4 modified cnt to 16
Writer 5 modified cnt to 32
```

Question 4

Write a Program to implement banker's algorithm for Deadlock avoidance

Code:-

```
#include<stdio.h>
int i,j;
int n;
int m;
int instance[10];
int max[10][10];
int allocation[10][10];
int available[10];
int need[10][10];
int sequence[10];
int availseq[10][10];
int flag1[10];
int doneprocess=0;

void readsize()
{
  printf("Enter number of Processess : ");
  scanf("%d",&n);
  printf("Enter number of Types of Resources : ");
  scanf("%d",&m);
}

void readinstances()
{
  printf("Enter Maximum Instance of Each Resources : \n");
  for(i=0;i<m;i++)
  {
    printf("%c : ",i+'A');
    scanf("%d",&instance[i]);
  }
}
```

```
void readallocation(){
  printf("Enter Instance of Each Resources Each
                              Process Currently Holds :\n\t");
  for(i=0;i<m;i++)
    printf("%c ",i+'A');
  printf("\n");
  for(i=0;i<n;i++){
    printf("P%d-->   ",i);
    for(j=0;j<m;j++)
      scanf("%d",&allocation[i][j]);
  }
}
void readmax(){
  printf("Enter Instance of Each Resources Each
                          Process can Maximum Request :\n\t");
  for(i=0;i<m;i++)
    printf("%c ",i+'A');
  printf("\n");
  for(i=0;i<n;i++){
    printf("P%d-->   ",i);
    for(j=0;j<m;j++)
      scanf("%d",&max[i][j]);
  }
}
void dispavail(){
  printf("\nAvailable Resources : ");
  for(i=0;i<m;i++){
    available[i]=instance[i];
    for(j=0;j<n;j++)
      available[i]-=allocation[j][i];
    printf("%d ",available[i]);
  }
}
void dispneed(){
  printf("\n\nNeed of Each Resources of
                                        Each Process :\n\t");
  for(i=0;i<m;i++)
    printf("%c ",i+'A');
```

```
  for(i=0;i<n;i++){
    printf("\nP%d\t",i);
    for(j=0;j<m;j++){
      need[i][j]=max[i][j]-allocation[i][j];
      printf("%d ",need[i][j]);
    }
  }
}
int check(){
  int count=0;
  for(i=0;i<n;i++){
    if(flag1[i]==1)
      count++;
  }
  if(count==n)
    return 0;
  else if(doneprocess==count)
    return -1;
  else{
    doneprocess=count;
    return 1;
  }
}
void bankers()
{
  int flag2;
  int ans=0;
  int idx=0;
  for(i=0;i<n;i++)
    flag1[i]=0;
  dispneed();
  do{
    for(i=0;i<n;i++)
    {
      if(flag1[i]==0)
      {
        flag2=0;
        for(j=0;j<m;j++){
```

```
          if(need[i][j]>available[j]){
            flag2=1;
            break;
          }
        }
        if(flag2==0){
          sequence[idx]=i;
          flag1[i]=1;
          for(j=0;j<m;j++){
            availseq[idx][j]=available[j];
            available[j]+=allocation[i][j];
          }
          idx++;
        }
      }
    }
  }while((ans=check())==1);
  if(ans==-1)
    printf("\nSystem in unsafe state\nDeadlock may occur");
  else
  {
    printf("\nSystem in Safe state
                              \nDeadlock will not occur\n");
    printf("\nSafe Sequence : <");
    for(i=0;i<n;i++)
    {
      printf(" P%d(",sequence[i]);
      for(j=0;j<m;j++)
        printf(" %d",availseq[i][j]);
      printf(" )");
    }
    printf(" >");
  }
}
int main()
{
  readsize();
  readinstances();
```

```
  readallocation();
  readmax();
  dispavail();
  bankers();
  return 0;
}
```

Output:-

```
PRAVIN@DESKTOP-B2LB8FB ~/oslab/da3
$ gcc Bankers.c  -o Bankers.exe

PRAVIN@DESKTOP-B2LB8FB ~/oslab/da3
$ ./Bankers.exe
Enter number of Processess : 5
Enter number of Types of Resources : 4
Enter Maximum Instance of Each Resources :
A : 3
B : 14
C : 12
D : 12
Enter Instance of Each Resources Each Process Currently Holds :
       A B C D
P0-->  0 0 1 2
P1-->  1 0 0 0
P2-->  1 3 5 4
P3-->  0 6 3 2
P4-->  0 0 1 4
Enter Instance of Each Resources Each Process can Maximum Request :
       A B C D
P0-->  0 0 1 2
P1-->  1 7 5 0
P2-->  2 3 5 6
P3-->  0 6 5 2
P4-->  0 6 5 6

Available Resources : 1 5 2 0

Need of Each Resources of Each Process :
       A B C D
P0     0 0 0 0
P1     0 7 5 0
P2     1 0 0 2
P3     0 0 2 0
P4     0 6 4 2
System in Safe state
Deadlock will not occur

Safe Sequence : < P0( 1 5 2 0 ) P2( 1 5 3 2 ) P3( 2 8 8 6 ) P4( 2 14 11 8 ) P1( 2 14 12 12 ) >
```