OPERATING SYSTEMS
LAB – 6

Name:- PANDUGA VENKATA JAYA SRIKANTH REDDY
Reg No:- 21MIS1095

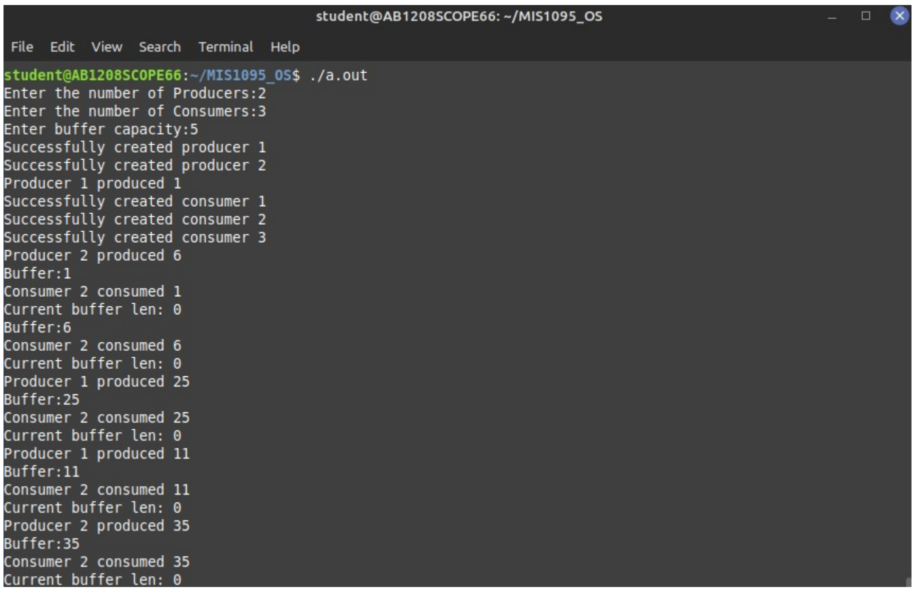
Synchronization

1) Producer Consumer Problem

CODE:-

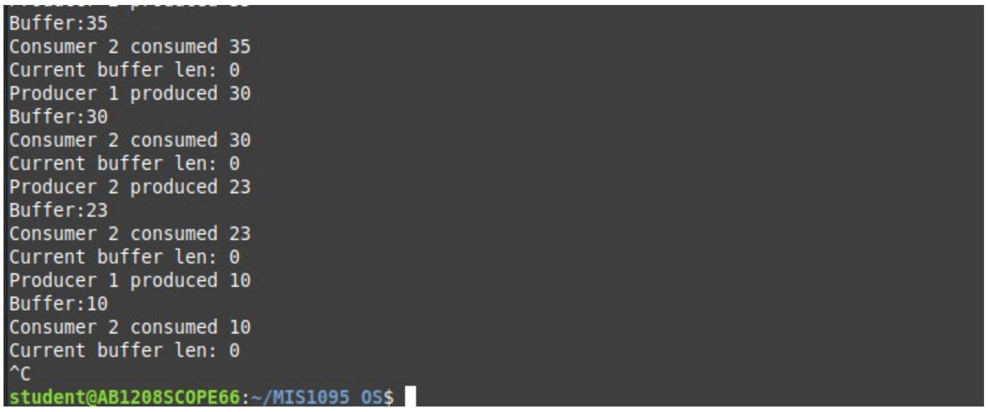
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>
pthread_t *producers;
pthread_t *consumers;

sem_t buf_mutex,empty_count,fill_count;
int *buf,buf_pos=-1,prod_count,con_count,buf_len;

int produce(pthread_t self){
    int i = 0;
    int p = 1 + rand()%40;
    while(!pthread_equal(*(producers+i),self) && i < prod_count){
                        i++;
    }
    printf("Producer %d produced %d \n",i+1,p);
    return p;
}
void consume(int p,pthread_t self){
    int i = 0;
    while(!pthread_equal(*(consumers+i),self) && i < con_count){
        i++;
    }
    printf("Buffer:");
    for(i=0;i<=buf_pos;++i)
        printf("%d ",*(buf+i));
    printf("\nConsumer %d consumed %d \nCurrent buffer len: %d\n",i+1,p,buf_pos);
}
void* producer(void *args){
    while(1){
        int p = produce(pthread_self());
        sem_wait(&empty_count);
        sem_wait(&buf_mutex);
        ++buf_pos;          // critical section
        *(buf + buf_pos) = p;
        sem_post(&buf_mutex);
        sem_post(&fill_count);
        sleep(1 + rand()%3);
    }
    return NULL;
}
void* consumer(void *args){
    int c;
    while(1){
        sem_wait(&fill_count);
        sem_wait(&buf_mutex);
        c = *(buf+buf_pos);
consume(c,pthread_self());
--buf_pos;
sem_post(&buf_mutex);
sem_post(&empty_count);
sleep(1+rand()%5);
}
return NULL;
}
int main(void){
    int i,err;
    srand(time(NULL));
    sem_init(&buf_mutex,0,1);
    sem_init(&fill_count,0,0);
    printf("Enter the number of Producers:");
    scanf("%d",&prod_count);
    producers = (pthread_t*) malloc(prod_count*sizeof(pthread_t));
    printf("Enter the number of Consumers:");
    scanf("%d",&con_count);
    consumers = (pthread_t*) malloc(con_count*sizeof(pthread_t));
    printf("Enter buffer capacity:");
    scanf("%d",&buf_len);
    buf = (int*) malloc(buf_len*sizeof(int));
    sem_init(&empty_count,0,buf_len);
    for(i=0;i<prod_count;i++){
        err = pthread_create(producers+i,NULL,&producer,NULL);
        if(err != 0){
            printf("Error creating producer %d: %s\n",i+1,strerror(err));
        }else{
printf("Successfully created producer %d\n",i+1);
}
}
for(i=0;i<con_count;i++){
    err = pthread_create(consumers+i,NULL,&consumer,NULL);
    if(err != 0){
        printf("Error creating consumer %d: %s\n",i+1,strerror(err));
    }else{
        printf("Successfully created consumer %d\n",i+1);
    }
}
for(i=0;i<prod_count;i++){
    pthread_join(*(producers+i),NULL);
}
for(i=0;i<con_count;i++){
    pthread_join(*(consumers+i),NULL);
}
return 0;
}
```

OUTPUT:-



student@AB1208SCOPE66: ~/MIS1095_OS
File Edit View Search Terminal Help

```
student@AB1208SCOPE66:~/MIS1095_OS$ ./a.out
Enter the number of Producers:2
Enter the number of Consumers:3
Enter buffer capacity:5
Successfully created producer 1
Successfully created producer 2
Producer 1 produced 1
Successfully created consumer 1
Successfully created consumer 2
Successfully created consumer 3
Producer 2 produced 6
Buffer:1
Consumer 2 consumed 1
Current buffer len: 0
Buffer:6
Consumer 2 consumed 6
Current buffer len: 0
Producer 1 produced 25
Buffer:25
Consumer 2 consumed 25
Current buffer len: 0
Producer 1 produced 11
Buffer:11
Consumer 2 consumed 11
Current buffer len: 0
Producer 2 produced 35
Buffer:35
Consumer 2 consumed 35
Current buffer len: 0
```



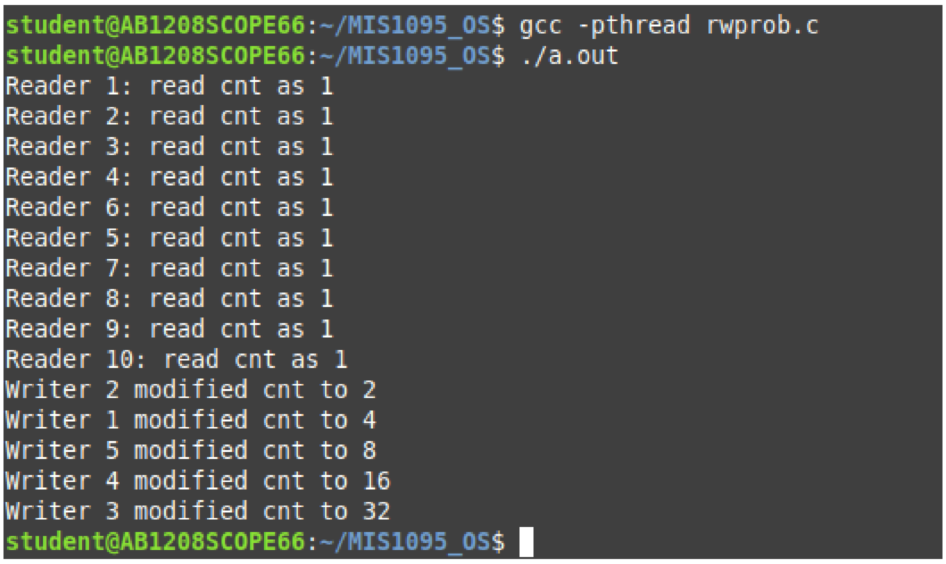
```
Buffer:35
Consumer 2 consumed 35
Current buffer len: 0
Producer 1 produced 30
Buffer:30
Consumer 2 consumed 30
Current buffer len: 0
Producer 2 produced 23
Buffer:23
Consumer 2 consumed 23
Current buffer len: 0
Producer 1 produced 10
Buffer:10
Consumer 2 consumed 10
Current buffer len: 0
^C
student@AB1208SCOPE66:~/MIS1095 OS$
```

2) Reader Writer’s Problem

CODE:-

```
#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)
{
    // 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 cnt as %d\n",*((int *)rno),cnt);



    // 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()
{
    pthread_t read[10],write[5];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&wrt,0,1);



    int a[10] = {1,2,3,4,5,6,7,8,9,10}; //Just used for numbering the producer and consumer



    for(int i = 0; i < 10; i++) {
        pthread_create(&read[i], NULL, (void *)reader, (void *)&a[i]);
    }
    for(int i = 0; i < 5; i++) {
        pthread_create(&write[i], NULL, (void *)writer, (void *)&a[i]);
    }



    for(int i = 0; i < 10; i++) {
        pthread_join(read[i], NULL);
    }
    for(int i = 0; i < 5; i++) {
        pthread_join(write[i], NULL);
    }



    pthread_mutex_destroy(&mutex);
    sem_destroy(&wrt);



    return 0;

}
```

OUTPUT:-



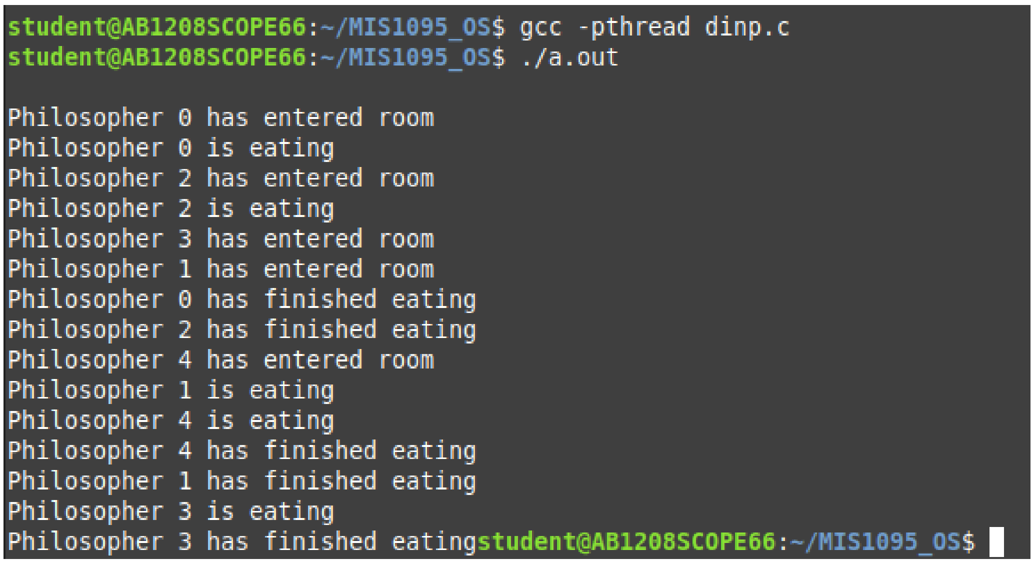
```
student@AB1208SCOPE66:~/MIS1095_OS$ gcc -pthread rwprob.c
student@AB1208SCOPE66:~/MIS1095_OS$ ./a.out
Reader 1: read cnt as 1
Reader 2: read cnt as 1
Reader 3: read cnt as 1
Reader 4: read cnt as 1
Reader 6: read cnt as 1
Reader 5: read cnt as 1
Reader 7: read cnt as 1
Reader 8: read cnt as 1
Reader 9: read cnt as 1
Reader 10: read cnt as 1
Writer 2 modified cnt to 2
Writer 1 modified cnt to 4
Writer 5 modified cnt to 8
Writer 4 modified cnt to 16
Writer 3 modified cnt to 32
student@AB1208SCOPE66:~/MIS1095_OS$
```

3) Dining Philosophers Probelm

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:-



```
student@AB1208SCOPE66:~/MIS1095_OS$ gcc -pthread dinp.c
student@AB1208SCOPE66:~/MIS1095_OS$ ./a.out

Philosopher 0 has entered room
Philosopher 0 is eating
Philosopher 2 has entered room
Philosopher 2 is eating
Philosopher 3 has entered room
Philosopher 1 has entered room
Philosopher 0 has finished eating
Philosopher 2 has finished eating
Philosopher 4 has entered room
Philosopher 1 is eating
Philosopher 4 is eating
Philosopher 4 has finished eating
Philosopher 1 has finished eating
Philosopher 3 is eating
Philosopher 3 has finished eatingstudent@AB1208SCOPE66:~/MIS1095_OS$
```