Assignment 4

4a - Producer Consumer

Code

4a.c

```
/*
Problem Statement - Thread synchronization using counting semaphores. Application
to demonstrate: producer-consumer problem with counting semaphores and mutex.
*/
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <pthread.h>
#include <string.h>
#include <semaphore.h>

sem_t full, empty;
pthread_mutex_t lock;
int pr, cs;
void *Producer(void *arg);
void *Consumer(void *arg);
int *BUFFER;
int buff_size;


int main()
{
    printf("Enter the buffer size\n");
    scanf("%d",&buff_size);
    printf("Enter the number of producers:");
    scanf("%d", &pr);
    printf("Enter the number of consumers:");
    scanf("%d", &cs);


    BUFFER = (int *)malloc(buff_size * sizeof(int));
    for (int i = 0; i < buff_size; i++) BUFFER[i] = 0;

    pthread_t p[pr], c[cs];
    void *thread_result;
    int res;

    sem_init(&empty, 0, buff_size-1);
    sem_init(&full, 0, 0);
    pthread_mutex_init(&lock, NULL);

    for (int i = 0; i < pr; i++)
    {
        res = pthread_create(&p[i], NULL, Producer, (void *)&i);
        if (res != 0)
        {
            perror("Thread creation failed");
            exit(EXIT_FAILURE);
        }
    }
    for (int i = 0; i < cs; i++)
    {
        res = pthread_create(&c[i], NULL, Consumer, (void *)&i);
        if (res != 0)
        {
            perror("Thread creation failed");
            exit(EXIT_FAILURE);
        }
    }
    for (int i = 0; i < pr; i++)
    {
        res = pthread_join(p[i], NULL);
        if (res != 0)
        {
            perror("Thread join failed");
            exit(EXIT_FAILURE);
        }
    }
    for (int i = 0; i < cs; i++)
    {
        res = pthread_join(c[i], NULL);
        if (res != 0)
        {
            perror("Thread join failed");
            exit(EXIT_FAILURE);
        }
    }
    exit(EXIT_SUCCESS);
    return 0;
}

void *Producer(void *arg)
{
    int index= *(int *)arg;
    while(1)
    {
        sem_wait(&empty);
        pthread_mutex_lock(&lock);
        if(BUFFER[index]==0){
            printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
            printf("Producer thread: %d\n", index);
            BUFFER[index]=1;
            for(int i=0;i<buff_size;i++){
                printf(" %d", BUFFER[i]);
            }
            printf("\n");
        }
        else{
            sleep(1);
        }
        //printf("Hello,This is producer code\n");
        pthread_mutex_unlock(&lock);
        sem_post(&full);
    }
    //pthread_exit("Exit producer..");
}

void *Consumer(void *arg)
{
    int index= *(int *)arg;
    while(1)
    {
        sem_wait(&full);
        pthread_mutex_lock(&lock);
        if(BUFFER[index]==1){
            printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
            printf("Consumer thread: %d\n", index);
            BUFFER[index]=0;
            for(int i=0;i<buff_size;i++){
                printf(" %d", BUFFER[i]);
            }
            printf("\n");
        }
```

```
        else{
            sleep(1);
        }

        pthread_mutex_unlock(&lock);
        sem_post(&empty);
    }
}
```

Output

```
abhishek-jadhav@abhishek-jadhav-ubuntu:~/Codes/OS Assignments/33232$ ./a.out
Enter the buffer size
10
Enter the number of producers:4
Enter the number of consumers:6
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Producer thread: 1
 0 1 0 0 0 0 0 0 0 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Producer thread: 5
 0 1 0 0 0 1 0 0 0 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Producer thread: 2
 0 1 1 0 0 1 0 0 0 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Consumer thread: 5
 0 1 1 0 0 0 0 0 0 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Producer thread: 5
 0 1 1 0 0 1 0 0 0 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Consumer thread: 5
 0 1 1 0 0 0 0 0 0 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Consumer thread: 2
 0 1 0 0 0 0 0 0 0 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Producer thread: 5
 0 1 0 0 0 1 0 0 0 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
^C
abhishek-jadhav@abhishek-jadhav-ubuntu:~/Codes/OS Assignments/33232$
```

4b - Reader Writer

Code

4b.c

```
/*
Problem Statement - Implement C program to demonstrate Reader-Writer problem with
readers having priority using counting semaphores and mutex.
*/

#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
#include<semaphore.h>
#include<string.h>

int readcount,reader_number,writer_number;
sem_t read_sem,write_sem;

void *Reader(void *arg);
void *Writer(void *arg);
int number=1;

int main(){
        printf("Enter the number of reader:");
        scanf("%d",&reader_number);
        printf("Enter the number of writer");
        scanf("%d",&writer_number);

        pthread_t read[reader_number],write[writer_number];
        int res;

        sem_init(&read_sem,0,1);
        sem_init(&write_sem,0,1);

        for(int i=0;i<reader_number;i++)
        {
                res=pthread_create(&read[i],NULL,Reader,(void *)&i);
                if (res!=0){
                        perror("Thread creation failed");
                        exit(EXIT_FAILURE);
                }
        }

        for(int i=0;i<writer_number;i++)
        {
                res=pthread_create(&write[i],NULL,Writer,(void *)&i);
                if(res!=0)
                {
                        perror("Thread creation failed");
                        exit(EXIT_FAILURE);
                }
        }

        for(int i=0;i<reader_number;i++)
        {
                res=pthread_join(read[i],NULL);
                if(res!=0)
                {
                        perror("Thread join failed");
                        exit(EXIT_FAILURE);
                }
        }

        for(int i=0;i<writer_number;i++)
        {
                res=pthread_join(write[i],NULL);
                if(res!=0)
                {
                        perror("Thread join failed");
                        exit(EXIT_FAILURE);
                }
        }
        return 0;
}

void *Reader(void *arg){
        int index=*(int *)arg;
        while(1){
                sem_wait(&read_sem);
                readcount++;
                if(readcount==1)
                {
                        sem_wait(&write_sem);
                }
                sem_post(&read_sem);

                printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
                printf("Reader no%d\n",index);
                printf("Reading value: %d\n",number);
                sem_wait(&read_sem);
                readcount--;
                if(readcount==0){
                        sem_post(&write_sem);
                }
```

```
                sem_post(&read_sem);

                sleep(2);
        }
}

void *Writer(void *arg){
        int index=*(int *)arg;
        while(1){
                sem_wait(&write_sem);
                printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
                printf("Writer no: %d\n",index);
                printf("Value changed from %d",number);
                number++;
                printf(" to %d\n",number);

                sem_post(&write_sem);

                sleep(2);
        }
}
```

Output

```
abhishek-jadhav@abhishek-jadhav-ubuntu:~/Codes/OS Assignments/33232$ ./a.out
Enter the number of reader:5
Enter the number of writer2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Reader no2
Reading value: 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Reader no2
Reading value: 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Reader no4
Reading value: 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Reader no1
Reading value: 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Reader no1
Reading value: 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Writer no: 1
Value changed from 1 to 2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
Writer no: 2
Value changed from 2 to 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Reader no2
Reading value: 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Reader no2
Reading value: 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Reader no4
Reading value: 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Reader no1
Reading value: 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Reader no1
Reading value: 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Writer no: 1
^C
abhishek-jadhav@abhishek-jadhav-ubuntu:~/Codes/OS Assignments/33232$
```