// RW

```c
#include<pthread.h>
#include<semaphore.h>
#include<stdio.h>
#include<unistd.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("Writing is done by %d value: %d \n",(*((int *)wno)),cnt);
    sem_post(&wrt);
    //sleep(1);
    return NULL;
}
void *reader(void *rno){
    pthread_mutex_lock(&mutex);
    numreader++;
    if(numreader==1){
    sem_wait(&wrt);
    }
    pthread_mutex_unlock(&mutex);
    printf("%d is Reading the data: %d \n",*((int *)rno),cnt);
    pthread_mutex_lock(&mutex);
    numreader--;
    if(numreader==0){
        sem_post(&wrt);
    }
    pthread_mutex_unlock(&mutex);
    //sleep(1);
    return NULL;
}
int main(){
    pthread_t read[10],write[5];
    pthread_mutex_init(&mutex,NULL);
    //int wcnt=1,rcnt=1;
    sem_init(&wrt,0,1);
    int a[10]={1,2,3,4,5,6,7,8,9,10};
    for (int i=0;i<10;i++){
        pthread_create(&read[i],NULL,reader,(void*)&a[i]);//a[i] is just for
giving numbering to readers and writers
    }
    for(int i=0;i<5;i++){
        pthread_create(&write[i],NULL,writer,(void*)&a[i]);
    }
    for(int i=0;i<10;i++){
        pthread_join(read[i],NULL);// this is for waiting .. if we dont use it
then main function is also a thread so it can execute before the read-write
thread and terminate the program.
    }
    for(int i=0;i<5;i++){
        pthread_join(write[i],NULL);
    }
    pthread_mutex_destroy(&mutex);
    sem_destroy(&wrt);

return 0;
}
```

//PC

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

#define MaxItems 5
#define BufferSize 5

sem_t empty;
sem_t full;
int in=0,out=0,buffer[BufferSize];
pthread_mutex_t mutex;
void *producer(void *pno){
    int item;
    for (int i=0;i<MaxItems ;i++){
        item = rand()%10;
        sem_wait(&empty);// jar kahi empty asel tar next .. if some slots are
empty then go to next for production
        pthread_mutex_lock(&mutex);
        buffer[in]=item;
        printf("%d Produces: %d  at %d \n",*((int *)pno),item,in);
        in=(in+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
    }
    return NULL;
}
void *consumer(void *cno){
    for (int i=0;i<MaxItems;i++){
        sem_wait(&full);// jar kahi jaga full asel tar .. if some slots are
full then go to next
        pthread_mutex_lock(&mutex);
        int item = buffer[out];
        printf("%d Consumes %d from %d \n",*((int *)cno),item,out);
        out = (out+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
    }
    return NULL;
}
int main()
{
    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 (int i=0;i<5;i++){
        pthread_create(&pro[i],NULL,producer,(void *)&a[i]);
    }
    for(int i=0;i<5;i++){
        pthread_create(&con[i],NULL,consumer,(void *)&a[i]);
    }
    for(int i=0;i<5;i++){
        pthread_join(pro[i],NULL);
    }
    for(int i=0;i<5;i++){
        pthread_join(con[i],NULL);
    }
    pthread_mutex_destroy(&mutex);
    sem_destroy(&empty);
    sem_destroy(&full);
return 0;
}
```