PRODUCER CONSUMER PROBLEM / BOUNDED-BUFFER PROBLEM

```
#include<stdio.h>                    // using semaphore

#include<pthread.h>

#include<semaphore.h>


void* producer(void *arg);           //producer function

void* consumer(void *arg);           //consumer function

char buff[20];                       // common buffer size is 20

sem_t full,empty;                    // semaphores count no of full and

                                     empty buffer slots


int main()

{

pthread_t pid,cid;


sem_init(&empty,0,1);          // Create the empty semaphore and initialize to 1

sem_init(&full,0,0);           // Create the full semaphore and initialize to 0

pthread_create(&pid,NULL,producer,NULL);

pthread_create(&cid,NULL,consumer,NULL);

pthread_join(pid,NULL);

pthread_join(cid,NULL);


}
```

```
void* producer(void*arg)

{

int run=1;

while (run)

{

sem_wait(&empty);                    // acquires empty lock


printf("\nEnter Mes to be add into buffer:");

scanf("%s",buff);


if(strcmp(buff,"end",3)==0)

run=0;


sem_post(&full);                                    //release the full

}

return NULL;

}

void* consumer(void *arg)

{

int run=1;

while(run)

{

sem_wait(&full);                     // acquire full lock

printf("\nConsumed item is %s\n",buff);
if(strcmp(buff,"end",3)==0)

run=0;



sem_post(&empty);

}

return NULL;

}
```

READER WRITER PROBLEM

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

sem_t mutex, wrt;         // semaphore mutex is used to ensure mutual
                          //exclusion when readcnt is updated
                          //  semaphore wrt is used by both readers and
                          //writers
                          // mutex (m)= reader, write block(w)= writer

int data = 0, rcount = 0;          //read count is used to maintain the number of
                                   readers currently accessing the resource or
                                   readcnt tells the number of processes
                                   performing read in the critical section, initially
                                   0

void *reader(void *arg)
{
  int f;
  f = ((int)arg);
  sem_wait(&mutex);                          // acquires locks
  rcount = rcount + 1;
  if(rcount==1)
   sem_wait(&wrt);                           // acquires locks
  sem_post(&mutex);                // releases the lock
  printf("Data read by the reader%d is %d\n",f,data);       // perform
                                                                  the reading
                                                                  operation
  sleep(1);
  sem_wait(&mutex);                                    //acquires lock
  rcount = rcount - 1;
  if(rcount==0)
   sem_post(&wrt);                                          // releases lock
  sem_post(&mutex);                                   // releases lock
}



void *writer(void *arg)
{
  int f;
  f = ((int) arg);
  sem_wait(&wrt);                                      //acquires lock
  data++;
  printf("Data writen by the writer%d is %d\n",f,data);     // perform
                                                                      the write
                                                                      operation
  sleep(1);
  sem_post(&wrt);                               // releases lock
}

main()
{
  int i;
  pthread_t  rtid[5], wtid[5];
  sem_init(&mutex,0,1);
  sem_init(&wrt,0,1);
  for(i=0;i<=2;i++)
  {
    pthread_create(&wtid[i],NULL,writer,i);
    pthread_create(&rtid[i],NULL,reader,i);
  }
  for(i=0;i<=2;i++)
  {
    pthread_join(wtid[i],NULL);
    pthread_join(rtid[i],NULL);
  }
}
```