**Problem**  
To make sure that the producer won’t try to add data into the buffer if it’s full and that the consumer won’t try to remove data from an empty buffer.

**Solution**  
The producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same way, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer.  
An inadequate solution could result in a deadlock where both processes are waiting to be awakened.

#include<stdio.h>

#include<semaphore.h>

#include<pthread.h>

#include<stdlib.h>

#define buffersize 10

**pthread\_mutex\_t** mutex;

**pthread\_t** tidP[**20**],tidC[**20**];

**sem\_t** full,empty;

**int** counter;

**int** buffer[buffersize];

**void** **initialize**()

{

pthread\_mutex\_init(&mutex,NULL);

sem\_init(&full,**1**,**0**);

sem\_init(&empty,**1**,buffersize);

counter=**0**;

}

**void** **write**(**int** item)

{

buffer[counter++]=item;

}

**int** **read**()

{

**return**(buffer[--counter]);

}

**void** \* **producer** (**void** \* param)

{

**int** waittime,item,i;

item=rand()%**5**;

waittime=rand()%**5**;

sem\_wait(&empty);

pthread\_mutex\_lock(&mutex);

printf("**\n**Producer has produced item: %d**\n**",item);

write(item);

pthread\_mutex\_unlock(&mutex);

sem\_post(&full);

}

**void** \* **consumer** (**void** \* param)

{

**int** waittime,item;

waittime=rand()%**5**;

sem\_wait(&full);

pthread\_mutex\_lock(&mutex);

item=read();

printf("**\n**Consumer has consumed item: %d**\n**",item);

pthread\_mutex\_unlock(&mutex);

sem\_post(&empty);

}

**int** **main**()

{

**int** n1,n2,i;

initialize();

printf("**\n**Enter the no of producers: ");

scanf("%d",&n1);

printf("**\n**Enter the no of consumers: ");

scanf("%d",&n2);

**for**(i=**0**;i<n1;i++)

pthread\_create(&tidP[i],NULL,producer,NULL);

**for**(i=**0**;i<n2;i++)

pthread\_create(&tidC[i],NULL,consumer,NULL);

**for**(i=**0**;i<n1;i++)

pthread\_join(tidP[i],NULL);

**for**(i=**0**;i<n2;i++)

pthread\_join(tidC[i],NULL);

//sleep(5);

exit(**0**);

}

**OUTPUT**

Enter theno of producers: 2

Enter the no of consumers: 2

Producer has produced item: 3

Producer has produced item: 2

Consumer has consumed item: 2

Consumer has consumed item: 3

#include<stdio.h> // using semaphore

#include<pthread.h>

#include<semaphore.h>

void\* producer(void \*arg);

void\* consumer(void \*arg);

char buff[20];

sem\_t full,empty;

int main()

{

pthread\_t pid,cid;

sem\_init(&empty,0,1);

sem\_init(&full,0,0);

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);

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

scanf("%s",buff);

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

run=0;

sem\_post(&full);

}

return NULL;

}

void\* consumer(void \*arg)

{

int run=1;

while(run)

{

sem\_wait(&full);

printf("\nConsumed item is %s\n",buff);

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

run=0;

sem\_post(&empty);

}

return NULL;

}