Assignment 4

Thread synchronisation using counting semaphores

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

#define BufferSize 5

void *producer(void *arg);
void *consumer(void *arg);

sem_t *empty;
sem_t *full;
pthread_mutex_t mutex;
int in=0;
int out=0;
int buffer[BufferSize];

int main() {

  int res,i,sizeP,sizeC;
  pthread_t *threadP,*threadC;

  //Accepting Number of Prodecers and Consumers
  printf("\nNumber of Producers: ");
  scanf("%d",&sizeP);

  printf("Number of Consumers: ");
  scanf("%d",&sizeC);
  printf("\n");

  //Creating one thread per precessor and per consumer. For
example if 3 producer then 3 threads for each producer.
  threadP=(pthread_t*)malloc(sizeof(pthread_t)*sizeP);
  threadC=(pthread_t*)malloc(sizeof(pthread_t)*sizeC);
  int pno[sizeP],cno[sizeC];

  //Creating semaphores
  full=sem_open("/fullSem", O_CREAT, 0644, 0);
  empty=sem_open("/emptySem", O_CREAT, 0644, BufferSize);
  pthread_mutex_init(&mutex, NULL);
  //Function Calls
  for(i=0;i<sizeP;i++) {
    pno[i]=i+1;
    res=pthread_create(&threadP[i], NULL, producer, &pno[i]);
  }
  for(i=0;i<sizeC;i++) {
    cno[i]=i+1;
    res=pthread_create(&threadC[i], NULL, consumer, &cno[i]);
  }
  for(i=0;i<sizeP;i++)
    res=pthread_join(threadP[i], NULL);
  for(i=0;i<sizeC;i++)
    res=pthread_join(threadC[i], NULL);

  //destroying the semaphores
  pthread_mutex_destroy(&mutex);
  sem_unlink("/fullSem");
  sem_unlink("/emptySem");

  exit(EXIT_SUCCESS);
}

void *producer(void *arg) {

  while(1) {
    int item=rand();

    sem_wait(empty);
    pthread_mutex_lock(&mutex); //Entering Critical Region

    buffer[in]=item%100;
    int pos=in;
    in=(in+1)%BufferSize;

    printf("Thread ID: %d. Producer: %d",(int)pthread_self()
%100,*(int*)arg); //Printing Thread ID and Producer Number
    printf("\nProduced item %d at %d position\nBuffer:
[ ",item%100,pos);

    //Displaying Buffer after Production
    for(int i=0;i<BufferSize;i++) {
      printf("%d ",buffer[i]);
    }
    printf("]\n\n");

    pthread_mutex_unlock(&mutex); //Exiting Critical Region
    sem_post(full);
    sleep(3);
  }
  pthread_exit(NULL);
}

void *consumer(void *arg) {

  while(1) {

    sem_wait(full);
    pthread_mutex_lock(&mutex); //Entering Critical Region

    int item=buffer[out];
    int pos=out;
    out=(out+1)%BufferSize;

    printf("Thread ID: %d. Consumer: %d",(int)pthread_self()
%100,*(int*)arg); //Printing Thread ID and Consumer Number
    printf("\nConsuming item %d from %d position\nBuffer:
[ ",item,pos);

    //Displaying Buffer before Consumption
    for(int i=0;i<BufferSize;i++) {
      printf("%d ",buffer[i]);
    }
    printf("]->[ ");

    //Displaying Buffer after Consumption
    buffer[pos]=0;
    for(int i=0;i<BufferSize;i++) {
      printf("%d ",buffer[i]);
    }
    printf("]\n\n");

    pthread_mutex_unlock(&mutex); //Exiting Critical Region
    sem_post(empty);
    sleep(3);
  }
  pthread_exit(NULL);
}
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <semaphore.h>

#define BufferSize 5
```

```
void *producer(void *arg);
void *consumer(void *arg);

sem_t *empty;
sem_t *full;
pthread_mutex_t mutex;
int in=0;
int out=0;
int buffer[BufferSize];

int main() {

  int res,i,sizeP,sizeC;
  pthread_t *threadP,*threadC;

  //Accepting Number of Prodecers and Consumers
  printf("\nNumber of Producers: ");
  scanf("%d",&sizeP);

  printf("Number of Consumers: ");
  scanf("%d",&sizeC);
  printf("\n");

  //Creating one thread per precessor and per consumer. For
example if 3 producer then 3 threads for each producer.
  threadP=(pthread_t*)malloc(sizeof(pthread_t)*sizeP);
  threadC=(pthread_t*)malloc(sizeof(pthread_t)*sizeC);
  int pno[sizeP],cno[sizeC];

  //Creating semaphores
  full=sem_open("/fullSem", O_CREAT, 0644, 0);
  empty=sem_open("/emptySem", O_CREAT, 0644, BufferSize);
  pthread_mutex_init(&mutex, NULL);

  //Function Calls
  for(i=0;i<sizeP;i++) {
    pno[i]=i+1;
    res=pthread_create(&threadP[i], NULL, producer, &pno[i]);
  }
  for(i=0;i<sizeC;i++) {
    cno[i]=i+1;
    res=pthread_create(&threadC[i], NULL, consumer, &cno[i]);
  }
  for(i=0;i<sizeP;i++)
    res=pthread_join(threadP[i], NULL);
  for(i=0;i<sizeC;i++)
    res=pthread_join(threadC[i], NULL);
  //destroying the semaphores
  pthread_mutex_destroy(&mutex);
  sem_unlink("/fullSem");
  sem_unlink("/emptySem");

  exit(EXIT_SUCCESS);
}

void *producer(void *arg) {

  while(1) {
    int item=rand();

    sem_wait(empty);
    pthread_mutex_lock(&mutex); //Entering Critical Region

    buffer[in]=item%100;
    int pos=in;
    in=(in+1)%BufferSize;

    printf("Thread ID: %d. Producer: %d",(int)pthread_self()
%100,*(int*)arg); //Printing Thread ID and Producer Number
    printf("\nProduced item %d at %d position\nBuffer:
[ ",item%100,pos);

    //Displaying Buffer after Production
    for(int i=0;i<BufferSize;i++) {
      printf("%d ",buffer[i]);
    }
    printf("]\n\n");

    pthread_mutex_unlock(&mutex); //Exiting Critical Region
    sem_post(full);
    sleep(3);
  }
  pthread_exit(NULL);
}

void *consumer(void *arg) {

  while(1) {

    sem_wait(full);
    pthread_mutex_lock(&mutex); //Entering Critical Region

    int item=buffer[out];
    int pos=out;
    out=(out+1)%BufferSize;
    printf("Thread ID: %d. Consumer: %d",(int)pthread_self()
%100,*(int*)arg); //Printing Thread ID and Consumer Number
    printf("\nConsuming item %d from %d position\nBuffer:
[ ",item,pos);

    //Displaying Buffer before Consumption
    for(int i=0;i<BufferSize;i++) {
      printf("%d ",buffer[i]);
    }
    printf("]->[ ");

    //Displaying Buffer after Consumption
    buffer[pos]=0;
    for(int i=0;i<BufferSize;i++) {
      printf("%d ",buffer[i]);
    }
    printf("]\n\n");

    pthread_mutex_unlock(&mutex); //Exiting Critical Region
    sem_post(empty);
    sleep(3);
  }
  pthread_exit(NULL);
}
```

Output:

```
Someshwars-MacBook-Pro:Assignments someshwargaikwad$ ./
Assignment

Number of Producers: 3
Number of Consumers: 2

Thread ID: 40. Consumer: 2
Consuming item 0 from 0 position
Buffer:[ 0 0 0 0 0 ]->[ 0 0 0 0 0 ]

Thread ID: 64. Consumer: 1
Consuming item 0 from 1 position
Buffer:[ 0 0 0 0 0 ]->[ 0 0 0 0 0 ]

Thread ID: 12. Producer: 2
Produced item 49 at 0 position
Buffer:[ 49 0 0 0 0 ]

Thread ID: 36. Producer: 1
Produced item 7 at 1 position
```

```
Buffer:[ 49 7 0 0 0 ]

Thread ID: 40. Consumer: 2
Consuming item 0 from 2 position
Buffer:[ 49 7 0 0 0 ]->[ 49 7 0 0 0 ]

Thread ID: 64. Consumer: 1
Consuming item 0 from 3 position
Buffer:[ 49 7 0 0 0 ]->[ 49 7 0 0 0 ]

Thread ID: 88. Producer: 3
Produced item 73 at 2 position
Buffer:[ 49 7 73 0 0 ]

Thread ID: 36. Producer: 1
Produced item 58 at 3 position
Buffer:[ 49 7 73 58 0 ]

Thread ID: 64. Consumer: 1
Consuming item 0 from 4 position
Buffer:[ 49 7 73 58 0 ]->[ 49 7 73 58 0 ]

Thread ID: 40. Consumer: 2
Consuming item 49 from 0 position
Buffer:[ 49 7 73 58 0 ]->[ 0 7 73 58 0 ]

Thread ID: 12. Producer: 2
Produced item 30 at 4 position
Buffer:[ 0 7 73 58 30 ]

Thread ID: 36. Producer: 1
Produced item 72 at 0 position
Buffer:[ 72 7 73 58 30 ]

Thread ID: 40. Consumer: 2
Consuming item 7 from 1 position
Buffer:[ 72 7 73 58 30 ]->[ 72 0 73 58 30 ]

Thread ID: 88. Producer: 3
Produced item 72 at 1 position
Buffer:[ 72 72 73 58 30 ]

Thread ID: 64. Consumer: 1
Consuming item 73 from 2 position
Buffer:[ 72 72 73 58 30 ]->[ 72 72 0 58 30 ]

Thread ID: 12. Producer: 2
Produced item 44 at 2 position
```

```
Buffer:[ 72 72 44 58 30 ]

Thread ID: 40. Consumer: 2
Consuming item 58 from 3 position
Buffer:[ 72 72 44 58 30 ]->[ 72 72 44 0 30 ]

Thread ID: 64. Consumer: 1
Consuming item 30 from 4 position
Buffer:[ 72 72 44 0 30 ]->[ 72 72 44 0 0 ]

Thread ID: 36. Producer: 1
Produced item 78 at 3 position
Buffer:[ 72 72 44 78 0 ]

Thread ID: 88. Producer: 3
Produced item 23 at 4 position
Buffer:[ 72 72 44 78 23 ]

Thread ID: 40. Consumer: 2
Consuming item 72 from 0 position
Buffer:[ 72 72 44 78 23 ]->[ 0 72 44 78 23 ]

Thread ID: 64. Consumer: 1
Consuming item 72 from 1 position
Buffer:[ 0 72 44 78 23 ]->[ 0 0 44 78 23 ]

Thread ID: 12. Producer: 2
Produced item 9 at 0 position
Buffer:[ 9 0 44 78 23 ]

Thread ID: 36. Producer: 1
Produced item 40 at 1 position
Buffer:[ 9 40 44 78 23 ]

^C
Someshwars-MacBook-Pro:Assignments someshwargaikwad$
```