OSLAB Q1

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#define NUM_THREADS 5

int shared_data = 0;
pthread_mutex_t mutex;
void *func1(void *arg)
{
    int _id = *((int *) arg);
    pthread_mutex_lock(&mutex);
    shared_data++;
    pthread_mutex_unlock(&mutex);
    return NULL;
}
int main()
{
    pthread_t tids[NUM_THREADS];
    int thread_args[NUM_THREADS];
    pthread_mutex_init(&mutex,NULL);

    for(int i=0;i<NUM_THREADS;i++)
    {
        thread_args[i] = i;
        pthread_create(&tids[i],NULL,func1,&thread_args[i]);
    }
    for(int i=0;i<NUM_THREADS;i++)
    {
        pthread_join(tids[i],NULL);
    }
    printf("The shared data is: %d",shared_data);
    pthread_mutex_destroy(&mutex);

    return 0;
}
```

```
student@VW:~$ gcc oslab10.c -o out
student@VW:~$ ./out
The shared data is: 5
student@VW:~$
```

OSLAB Q2

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include<unistd.h>
#define BUFFER_SIZE 5
sem_t mutex, empty, full;
int buffer[BUFFER_SIZE];
int in = 0, out = 0;
void *producer(void *arg) {
int item;
while (1) {
item = rand() % 100;
sem_wait(&empty);
sem_wait(&mutex);
buffer[in] = item;
printf("Produced: %d\n", item);
in = (in + 1) % BUFFER_SIZE;
sem_post(&mutex);
sem_post(&full);
sleep(rand() % 3);
}
}
void *consumer(void *arg) {
int item;
while (1) {
sem_wait(&full);
sem_wait(&mutex);
item = buffer[out];
printf("Consumed: %d\n", item);
out = (out + 1) % BUFFER_SIZE;
sem_post(&mutex);
sem_post(&empty);
sleep(rand() % 3);
}
}
```

```c
int main() {
pthread_t producer_thread, consumer_thread;

sem_init(&mutex, 0, 1);
sem_init(&empty, 0, BUFFER_SIZE);
sem_init(&full, 0, 0);
pthread_create(&producer_thread, NULL, producer, NULL);
pthread_create(&consumer_thread, NULL, consumer, NULL);
pthread_join(producer_thread, NULL);
pthread_join(consumer_thread, NULL);
sem_destroy(&mutex);
sem_destroy(&empty);
sem_destroy(&full);
return 0;
}
```

```
student@VW:~$ gcc boundedbuffer.c -o out
student@VW:~$ ./out
Produced: 83
Consumed: 83
Produced: 15
Consumed: 15
Produced: 86
Consumed: 86
Produced: 21
Consumed: 21
Produced: 90
Consumed: 90
```

```
Produced: 42
Consumed: 42
Produced: 21
Consumed: 21
Produced: 37
Consumed: 37
Produced: 15
Consumed: 15
Produced: 13
Consumed: 13
Produced: 56
Produced: 62
^Z
[1]+  Stopped                 ./out
student@VW:~$
```

OSLAB Q3

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<semaphore.h>
#include<pthread.h>
#include<unistd.h>
#define NUM_READERS 5
#define NUM_WRITERS 2
#define STRING_LENGTH 60
pthread_t readers[NUM_READERS], writers[NUM_WRITERS];
sem_t mutex, rw_mutex;
int readers_count = 0;
FILE *file;
char generateRandomChar() {
return (char)('a' + rand() % 26);
}
void *reader(void *arg) {
while (1) {
sem_wait(&mutex);
readers_count++;
if (readers_count == 1) {
sem_wait(&rw_mutex);
}
sem_post(&mutex);
fseek(file, 0, SEEK_SET);
char buffer[256];
while (fgets(buffer, sizeof(buffer), file) != NULL) {
fprintf(stdout,"Reader %ld: %s", (long)arg, buffer);
}
sem_wait(&mutex);
readers_count--;
if (readers_count == 0) {
sem_post(&rw_mutex);
}
sem_post(&mutex);
usleep(1000);
}
}
```

```c
void *writer(void *arg) {
while (1) {
sem_wait(&rw_mutex);
srand(time(NULL));
char randomString[STRING_LENGTH + 1];
for (int i = 0; i < STRING_LENGTH; i++) {
randomString[i] = generateRandomChar();
}
randomString[STRING_LENGTH] = '\0';
fseek(file, 0, SEEK_END);
fprintf(file, "%s\n",randomString);
fprintf(stdout, "Writer %ld: %s\n", (long)arg, randomString);
fflush(file);
sem_post(&rw_mutex);

usleep(1000);
}
}
```

```c
int main() {
    file = fopen("shared_file.txt", "a+");
    if (file == NULL) {
    perror("Error opening file");
    exit(EXIT_FAILURE);
    }
    sem_init(&mutex, 0, 1); sem_init(&rw_mutex, 0, 1);
    int i;
    for (i = 0; i < NUM_WRITERS; i++) pthread_create(&writers[i], NULL, writer, (void *)(long)i);
    for (i = 0; i < NUM_READERS; i++) pthread_create(&readers[i], NULL, reader, (void *)(long)i);
    for (i = 0; i < NUM_READERS; i++) pthread_join(readers[i], NULL);
    for (i = 0; i < NUM_WRITERS; i++) pthread_join(writers[i], NULL);
    fprintf(stdout, "reader pthread join completed\n");
    sem_destroy(&mutex); sem_destroy(&rw_mutex);
    fclose(file);
    return 0;
}
```

```
[7]+  Stopped                 ./out
student@VW:~$ gcc oslabreaders_writers.c -o out -pthread
student@VW:~$
```

```
Reader 0: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 3: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 4: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 0: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 0: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
Reader 2: roaxnixjmfvvjzwxrekekuozvaergnmxbourwrciyyghzehqjtuvokwlkafq
```

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<semaphore.h>
#include<pthread.h>
#include<unistd.h>
#define NUM_READERS 5
#define NUM_WRITERS 2
#define STRING_LENGTH 60
pthread_t readers[NUM_READERS], writers[NUM_WRITERS];
sem_t mutex, rw_mutex;
int readers_count = 0;
FILE *file;
char generateRandomChar() {
return (char)('a' + rand() % 26);
}
void *reader(void *arg) {
while (1) {
sem_wait(&mutex);
readers_count++;
if (readers_count == 1) {
sem_wait(&rw_mutex);
}
sem_post(&mutex);
fseek(file, 0, SEEK_SET);
char buffer[256];
while (fgets(buffer, sizeof(buffer), file) != NULL) {
fprintf(stdout,"Reader %ld: %s", (long)arg, buffer);
}
sem_wait(&mutex);
readers_count--;
if (readers_count == 0) {
sem_post(&rw_mutex);
}
sem_post(&mutex);
usleep(1000);
}
}
```

```
39  void *writer(void *arg) {
40  while (1) {
41  sem_wait(&rw_mutex);
42  srand(time(NULL));
43  char randomString[STRING_LENGTH + 1];
44  for (int i = 0; i < STRING_LENGTH; i++) {
45  randomString[i] = generateRandomChar();
46  }
47  randomString[STRING_LENGTH] = '\0';
48  fseek(file, 0, SEEK_END);
49  fprintf(file, "%s\n",randomString);
50  fprintf(stdout, "Writer %ld: %s\n", (long)arg, randomString);
51  fflush(file);
52  sem_post(&rw_mutex);
53
54  usleep(1000);
55  }
56  }
```

```
57  int main() {
58      file = fopen("shared_file.txt", "a+");
59      if (file == NULL) {
60      perror("Error opening file");
61      exit(EXIT_FAILURE);
62      }
63      sem_init(&mutex, 0, 1); sem_init(&rw_mutex, 0, 1);
64      int i;
65      for (i = 0; i < NUM_WRITERS; i++) pthread_create(&writers[i], NULL, writer, (void *)(long)i);
66      for (i = 0; i < NUM_READERS; i++) pthread_create(&readers[i], NULL, reader, (void *)(long)i);
67      for (i = 0; i < NUM_READERS; i++) pthread_join(readers[i], NULL);
68      for (i = 0; i < NUM_WRITERS; i++) pthread_join(writers[i], NULL);
69      fprintf(stdout, "reader pthread join completed\n");
70      sem_destroy(&mutex); sem_destroy(&rw_mutex);
71      fclose(file);
72      return 0;
73  }
```