

Lab10

Name: Dawar Abbas

Reg. No.: 1144

```

1  #include <stdio.h>
2  #include <pthread.h>
3  #include <semaphore.h>
4  #include <unistd.h>
5  #define BUFFER_SIZE 5
6  int buffer[BUFFER_SIZE];
7  int in = 0; // Producer index
8  int out = 0; // Consumer index
9  sem_t empty; // Counts empty slots
10 sem_t full; // Counts full slots
11 pthread_mutex_t mutex;
12 void *producer(void *arg)
13 {
14     int id = *(int *)arg;
15     for (int i = 0; i < 3; i++)
16     { // Each producer makes 3 items
17         int item = id * 100 + i;
18         // TODO: Wait for empty slot
19         sem_wait(&empty);
20         // TODO: Lock the buffer
21         pthread_mutex_lock(&mutex);
22         // Add item to buffer
23         buffer[in] = item;
24         printf("Producer %d produced item %d at position %d\n",
25                id, item, in);
26         in = (in + 1) % BUFFER_SIZE;
27         // TODO: Unlock the buffer
28         pthread_mutex_unlock(&mutex);
29         // TODO: Signal that buffer has a full slot
30         sem_post(&full);
31         sleep(1);
32     }
33     return NULL;
34 }
35 void *consumer(void *arg)
36 {
37     int id = *(int *)arg;
38     for (int i = 0; i < 3; i++)
39     {
40         // TODO: Students complete this similar to producer
41         sem_wait(&full);
42         pthread_mutex_lock(&mutex);
43         int item = buffer[out];
44         printf("Consumer %d consumed item %d from position %d\n",
45                id, item, out);
46         out = (out + 1) % BUFFER_SIZE;
47         pthread_mutex_unlock(&mutex);
48         sem_post(&empty);
49         sleep(2); // Consumers are slower
50     }
51     return NULL;
52 }
53 int main()
54 {
55     pthread_t prod[2], cons[2];
56     int ids[2] = {1, 2};
57     // Initialize semaphores
58     sem_init(&empty, 0, BUFFER_SIZE); // All slots empty initially
59     sem_init(&full, 0, 0);
60     pthread_mutex_init(&mutex, NULL);
61     // No slots full initially
62     // Create producers and consumers
63     for (int i = 0; i < 2; i++)
64     {
65         pthread_create(&prod[i], NULL, producer, &ids[i]);
66         pthread_create(&cons[i], NULL, consumer, &ids[i]);
67     }
68     // Wait for completion
69     for (int i = 0; i < 2; i++)
70     {
71         pthread_join(prod[i], NULL);
72         pthread_join(cons[i], NULL);
73     }
74     // Cleanup
75     sem_destroy(&empty);
76     sem_destroy(&full);
77     pthread_mutex_destroy(&mutex);
78     return 0;
79 }

```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
dawar@DESKTOP-EG875J4:~/operating-system-1144/lab10$ ./task1.out
Car 10 parked successfully!
Car 9 is leaving.
Car 10 is leaving.
• dawar@DESKTOP-EG875J4:~/operating-system-1144/lab10$ gcc task2.c -o task2.out -lpthread
• dawar@DESKTOP-EG875J4:~/operating-system-1144/lab10$ ./task2.out
Producer 1 produced item 100 at position 0
Consumer 1 consumed item 100 from position 0
Producer 2 produced item 200 at position 1
Consumer 2 consumed item 200 from position 1
Producer 1 produced item 101 at position 2
Producer 2 produced item 201 at position 3
Consumer 1 consumed item 101 from position 2
Consumer 2 consumed item 201 from position 3
Producer 1 produced item 102 at position 4
Producer 2 produced item 202 at position 0
Consumer 2 consumed item 102 from position 4
Consumer 1 consumed item 202 from position 0
• dawar@DESKTOP-EG875J4:~/operating-system-1144/lab10$ gcc task2.c -o task2.out -lpthread
• dawar@DESKTOP-EG875J4:~/operating-system-1144/lab10$ ./task2.out
```

In this program we are using counting semaphore for producer consumer problem.

There are two producers and two consumers for a fixed 5 sized buffer. So when buffer is full we use the full thread to check and consume it by consumers when buffer is empty it will wait for it to produce first.

There are five steps both functions:

Producer:

- -1 from empty (wait(&empty))
- Lock
- Critical section
- Unlock
- +1 in full (post(&full))

Consumer:

- -1 from full (wait(&full))
- Lock
- Critical section
- Unlock
- +1 in empty (post(&empty))

If buffer is empty and nothing to consume then it will go to deadlock.