



National Textile University
Department of Computer Science

Subject:

Opreating System

Submitted to:

Sir Nasir

Submitted by:

Ahmad Fawad

Reg number:

1129

Lab no. :

10

Semester:

5th

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define BUFFER_SIZE 5
int buffer[BUFFER_SIZE];
int in = 0; // Producer index
int out = 0; // Consumer index
sem_t empty; // Counts empty slots
sem_t full; // Counts full slots
pthread_mutex_t mutex;
void* producer(void* arg) {
    int id = *(int*)arg;
    for(int i = 0; i < 3; i++) { // Each producer makes 3 items
```

```
int item = id * 100 + i;

// TODO: Wait for empty slot

sem_wait(&empty);

// TODO: Lock the buffer

pthread_mutex_lock(&mutex);

// Add item to buffer

buffer[in] = item;

printf("Producer %d produced item %d at position %d\n",
id, item, in);

in = (in + 1) % BUFFER_SIZE;

// TODO: Unlock the buffer

pthread_mutex_unlock(&mutex);

// TODO: Signal that buffer has a full slot

sem_post(&full);

sleep(1);

}

return NULL;
}

void* consumer(void* arg) {

int id = *(int*)arg;

for(int i = 0; i < 4; i++) {

// TODO: Students complete this similar to producer

sem_wait(&full);

pthread_mutex_lock(&mutex);

int item = buffer[out];

printf("Consumer %d consumed item %d from position %d\n",
id, item, out);

out = (out + 1) % BUFFER_SIZE;
}
```

```
    id, item, out);

    out = (out + 1) % BUFFER_SIZE;

    pthread_mutex_unlock(&mutex);

    sem_post(&empty);

    sleep(2); // Consumers are slower

}

return NULL;
}

int main() {

pthread_t prod[2], cons[2];

int ids[2] = {1, 2};

// Initialize semaphores

sem_init(&empty, 0, BUFFER_SIZE); // All slots empty initially

sem_init(&full, 0, 0);

pthread_mutex_init(&mutex, NULL);

// No slots full initially

// Create producers and consumers

for(int i = 0; i < 2; i++) {

pthread_create(&prod[i], NULL, producer, &ids[i]);

pthread_create(&cons[i], NULL, consumer, &ids[i]);

}

// Wait for completion

for(int i = 0; i < 2; i++) {

pthread_join(prod[i], NULL);

pthread_join(cons[i], NULL);

}
```

```

// Cleanup

sem_destroy(&empty);

sem_destroy(&full);

pthread_mutex_destroy(&mutex);

return 0;

}

```

```

File Edit Selection View Go Run Terminal Help < >
EXPLORER ... Task2.c
LAB 10 [WSL: UBUNTU]
exe2
Task2.c
10  Welcome Task2.c
11  Semaphores // COUNTS FULL SLOTS
12  pthread_mutex_t mutex;
13  void* producer(void* arg) {
14      int id = *(int*)arg;
15      for(int i = 0; i < 3; i++) { // Each producer makes 3 items
16          int item = id * 100 + i;
17          // TODO: Wait for empty slot
18          sem_wait(&empty);
19          // TODO: Lock the buffer
20          pthread_mutex_lock(&mutex);
21          // Add item to buffer
22          buffer[in] = item;
23          printf("Producer %d produced item %d at position %d\n",
24                 id, item, in);
25          in = (in + 1) % BUFFER_SIZE;
26          // TODO: Unlock the buffer
27          pthread_mutex_unlock(&mutex);
28          // TODO: Signal that buffer has a full slot
29          sem_post(&full);
30          sleep(1);
31      }
32  }
33
34  void* consumer(void* arg) {
35      int id = *(int*)arg;
36      for(int i = 0; i < 2; i++) { // Each consumer consumes 2 items
37          // TODO: Wait for full slot
38          sem_wait(&full);
39          // TODO: Lock the buffer
40          pthread_mutex_lock(&mutex);
41          int item = buffer[in];
42          in = (in + 1) % BUFFER_SIZE;
43          // TODO: Unlock the buffer
44          pthread_mutex_unlock(&mutex);
45          printf("Consumer %d consumed item %d from position %d\n",
46                 id, item, in);
47      }
48  }
49
50  int main() {
51      pthread_t producer1, producer2, consumer1, consumer2;
52      int producer_id = 1, consumer_id = 1;
53
54      // Initialize semaphores
55      sem_init(&empty, 0, 0);
56      sem_init(&full, 1, BUFFER_SIZE);
57
58      // Create threads
59      pthread_create(&producer1, NULL, producer, &producer_id);
60      pthread_create(&producer2, NULL, producer, &producer_id);
61      pthread_create(&consumer1, NULL, consumer, &consumer_id);
62      pthread_create(&consumer2, NULL, consumer, &consumer_id);
63
64      // Join threads
65      pthread_join(producer1, NULL);
66      pthread_join(producer2, NULL);
67      pthread_join(consumer1, NULL);
68      pthread_join(consumer2, NULL);
69
70      // Clean up
71      sem_destroy(&empty);
72      sem_destroy(&full);
73      pthread_mutex_destroy(&mutex);
74
75      return 0;
76  }

```

The terminal output shows:

```

ahmedfawad@DESKTOP-DEKBNV1:/Lab 10$ ./exe2
Producer 1 produced item 102 at position 4
Producer 2 produced item 202 at position 0
Consumer 1 consumed item 102 from position 4
Consumer 2 consumed item 202 from position 0
^C
ahmedfawad@DESKTOP-DEKBNV1:/Lab 10$ 

```

If we can increase the value of consumer then it goes in DeadLock Program can not Execute

Demonstration:

1. One semaphore is for producer and 1 for consumer and mutex is used to lock and unlock the space
2. space
3. Semaphore name are full and empty ,empty count empty space ,and full count filled space
4. Sem_wait() wait for the empty space
5. Sem_post() for mean space is taken

6. Wait for the completing all thread and and destroy at the end

Possible value will be

- $Id * 100 + [i], i=0,1,2 \quad id=1,2$
- 101
- 102
- 103
- 201
- 202
- 201