



National Textile University
Department of Computer Science

Subject:

Operating 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;

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
```

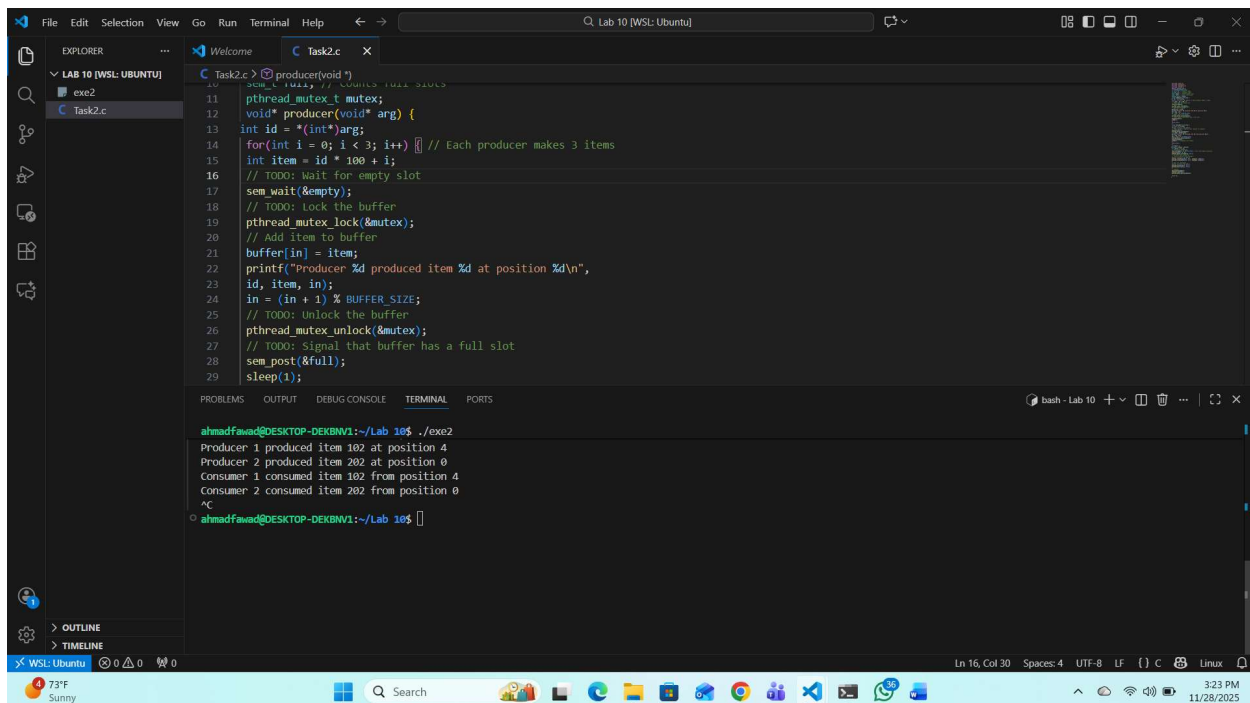
```
sem_destroy(&empty);
```

```
sem_destroy(&full);
```

```
pthread_mutex_destroy(&mutex);
```

```
return 0;
```

```
}
```



```
10 sem_t full; // Counts full slots
11 pthread_mutex_t mutex;
12 void* producer(void* arg) {
13     int id = *(int*)arg;
14     for(int i = 0; i < 3; i++) { // Each producer makes 3 items
15         int item = id * 100 + i;
16         // TODO: Wait for empty slot
17         sem_wait(&empty);
18         // TODO: Lock the buffer
19         pthread_mutex_lock(&mutex);
20         // Add item to buffer
21         buffer[in] = item;
22         printf("Producer %d produced item %d at position %d\n",
23               id, item, in);
24         in = (in + 1) % BUFFER_SIZE;
25         // TODO: Unlock the buffer
26         pthread_mutex_unlock(&mutex);
27         // TODO: Signal that buffer has a full slot
28         sem_post(&full);
29         sleep(1);
30     }
31 }
```

```
ahmadfawad@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
ahmadfawad@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
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