



# National Textile University

## Department of Computer Science

Subject: Operating System

---

Submitted to: Nasir Mahmood

---

Submitted by: esha

---

Reg number:23-NTU-CS-1146

---

Lab : 06

---

Semester:5th

# Lab manual 10

## Task 1:

Code :

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

sem_t parking_spaces;

void* car(void* arg) {
    int id = *(int*)arg;
    printf("Car %d is trying to park...\n"
           , id);
    sem_wait(&parking_spaces); // Try to get a space
    printf("Car %d parked successfully!\n"
           , id);
    sleep(2); // Stay parked for 2 seconds
    printf("Car %d is leaving.\n"
           , id);
    sem_post(&parking_spaces); // Free the space
    return NULL;
}

int main() {
    pthread_t cars[10];
    int ids[10];
    // Initialize: 3 parking spaces available
    sem_init(&parking_spaces, 0, 3);
    // Create 10 cars (more than spaces!)
    for(int i = 0; i < 10; i++) {
        ids[i] = i + 1;
        pthread_create(&cars[i], NULL, car, &ids[i]);
    }
    // Wait for all cars
    for(int i = 0; i < 10; i++) {
```

```

pthread_join(cars[i], NULL);
}

sem_destroy(&parking_spaces);

return 0;
}

```

The screenshot shows a Windows desktop environment with the Visual Studio Code (VS Code) application open. The title bar indicates the window is titled "Operating-System [WSL: Ubuntu]".

- File Explorer:** Shows the project structure. It includes files like docker-compose.yml, Dockerfile, package.json, server.js, index.html, and several .c and .out files under the "Operating-System" folder. A terminal tab is also visible.
- Code Editor:** Displays the content of q1.c. The code includes #include directives for stdio.h, pthread.h, semaphore.h, and unistd.h. It defines a sem\_t variable parking\_spaces and a void\* car function that prints messages for each car's action (parking or leaving), waits on the semaphore, sleeps for 2 seconds, and then prints it left.
- Terminal:** Shows the command esha@ESHA-DELL:~/Operating-System/Operating-System-1/1151-lab10\$ ./q1.out being run. The output of the program is displayed, showing 10 cars (Car 1 to Car 10) attempting to park, successfully parking, and then leaving.
- Taskbar:** At the bottom, the taskbar shows various icons for running applications like a browser, file manager, and system tools.

```

Operating-System-1 > 1151-lab10 > C q1.c ...
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <semaphore.h>
4 #include <unistd.h>
5 sem_t parking_spaces;
6 void* car(void* arg) {
7     int id = *(int*)arg;
8     printf("Car %d is trying to park...\n"
9 , id);
10    sem_wait(&parking_spaces); // Try to get a space
11    printf("Car %d parked successfully!\n"
12 , id);
13    sleep(2); // Stay parked for 2 seconds
14    printf("Car %d is leaving.\n"
...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
esha@ESHA-DELL:~/Operating-System/Operating-System-1/1151-lab10$ ./q1.out
Car 7 is trying to park...
Car 2 is trying to park...
Car 8 is trying to park...
Car 9 is trying to park...
Car 10 is trying to park...
Car 1 is leaving.
Car 7 parked successfully!
Car 5 is leaving.
Car 6 is leaving.
Car 2 parked successfully!
Car 8 parked successfully!
Car 7 is leaving.
Car 9 parked successfully!
Car 8 is leaving.
Car 2 is leaving.
Car 4 parked successfully!
Car 10 parked successfully!
Car 9 is leaving.
Car 3 parked successfully!
Car 4 is leaving.
Car 10 is leaving.
Car 3 is leaving.
8:28 PM 1/4/2026

```

## Task 2:

### Code:

```

#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 < 3; 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); // No slots full initially
    pthread_mutex_init(&mutex, NULL);
    // 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;
}
```

```

Operating-System-1 > 1151-lab10 > C q2.c > producer(void *)
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     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

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

esh@ESHA-DELL:~/Operating-System/Operating-System-1/1151-lab10$ ./q1.out
Car 4 is leaving.
Car 10 is leaving.
Car 3 is leaving.

● esha@ESHA-DELL:~/Operating-System/Operating-System-1/1151-lab10$ gcc q2.c -o q2.out -lpthread
● esha@ESHA-DELL:~/Operating-System/Operating-System-1/1151-lab10$ ./q2.out
Producer 2 produced item 200 at position 0
Consumer 2 consumed item 200 from position 0
Producer 1 produced item 100 at position 1
Consumer 1 consumed item 100 from position 1
Producer 1 produced item 101 at position 2
Producer 2 produced item 201 at position 3
Consumer 2 consumed item 101 from position 2
Consumer 1 consumed item 201 from position 3
Producer 2 produced item 202 at position 4
Producer 1 produced item 102 at position 0
Consumer 2 consumed item 202 from position 4
Consumer 1 consumed item 102 from position 0

```

esh@ESHA-DELL:~/Operating-System/Operating-System-1/1151-lab10\$

## Task 3: Block condition

Code :

```
#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;
```

```
// ----- PRODUCER -----
```

```

void* producer(void* arg) {
    int id = *(int*)arg;

    for(int i = 0; i < 3; i++) { // Each producer produces 3 items
        int item = id * 100 + i;

        sem_wait(&empty);           // Wait if buffer is full
        pthread_mutex_lock(&mutex); // Lock shared buffer

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

        in = (in + 1) % BUFFER_SIZE;

        pthread_mutex_unlock(&mutex); // Unlock buffer
        sem_post(&full);           // Signal item added

        sleep(1);
    }

    return NULL;
}

// ----- CONSUMER -----
void* consumer(void* arg) {
    int id = *(int*)arg;

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

        sem_wait(&full);           // Wait if buffer empty
        pthread_mutex_lock(&mutex); // Lock buffer

        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); // Unlock buffer
        sem_post(&empty);           // Signal item consumed
    }
}

```

```

    id, item, out);

out = (out + 1) % BUFFER_SIZE;

pthread_mutex_unlock(&mutex);    // Unlock buffer
sem_post(&empty);               // Signal empty slot

sleep(2); // Consumers are slower
}

return NULL;
}

// ----- MAIN -----
int main() {
    pthread_t prod[2], cons[2];
    int ids[2] = {1, 2};

    // Initialize semaphores
    sem_init(&empty, 0, BUFFER_SIZE); // All slots empty
    sem_init(&full, 0, 0);          // No slots full

    pthread_mutex_init(&mutex, NULL);

    // Create producer & consumer threads
    for(int i = 0; i < 2; i++) {
        pthread_create(&prod[i], NULL, producer, &ids[i]);
        pthread_create(&cons[i], NULL, consumer, &ids[i]);
    }

    // Join threads
    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;  
}
```

## Remarks:

- In this it will be in block deadlock situation
  - Because the values are only 6 but consumer is consuming 8 vaalues so it will go in block situation increasing value of consumer less values produce.

Q A:

- Total values produce: 6 in task 1
  - Threads : 4 ( read and write)
  - Semaphore : 2 (b empty and full)