

A. HARSHA MUKUNDHA - 192324070

19. Design a C program to implement process synchronization using mutex locks.

### **AIM**

To design a C program to implement process synchronization using mutex locks, ensuring mutual exclusion in a critical section.

### **ALGORITHM**

#### **1. Start**

- Include necessary header files.

#### **2. Initialize Mutex**

- Declare and initialize the mutex variable.

#### **3. Create Threads**

- Create multiple threads that need synchronization for accessing the critical section.

#### **4. Lock Mutex**

- In each thread function, acquire the mutex lock before entering the critical section.

#### **5. Critical Section Execution**

- Perform operations in the critical section.

#### **6. Unlock Mutex**

- Release the mutex lock after completing the critical section operations.

#### **7. Join Threads**

- Wait for all threads to complete their execution.

#### **8. Destroy Mutex**

- Destroy the mutex variable to free up resources.

#### **9. End**

## PROCEDURE

1. Declare and initialize a mutex.
2. Create threads using `pthread_create`.
3. Use `pthread_mutex_lock` before the critical section and `pthread_mutex_unlock` after.
4. Wait for threads to finish using `pthread_join`.
5. Destroy the mutex after execution.

## CODE:

```
#include <stdio.h>
```

```
#include <pthread.h>
```

```
#include <unistd.h>
```

```
pthread_mutex_t mutex;
```

```
void *critical_section(void *arg) {  
    pthread_mutex_lock(&mutex);  
    printf("Thread %ld in critical section.\n", pthread_self());  
    sleep(1);  
    printf("Thread %ld exiting critical section.\n", pthread_self());  
    pthread_mutex_unlock(&mutex);  
    return NULL;  
}
```

```
int main() {  
    pthread_t thread1, thread2;  
    pthread_mutex_init(&mutex, NULL);  
    pthread_create(&thread1, NULL, critical_section, NULL);
```

```
pthread_create(&thread2, NULL, critical_section, NULL);

pthread_join(thread1, NULL);

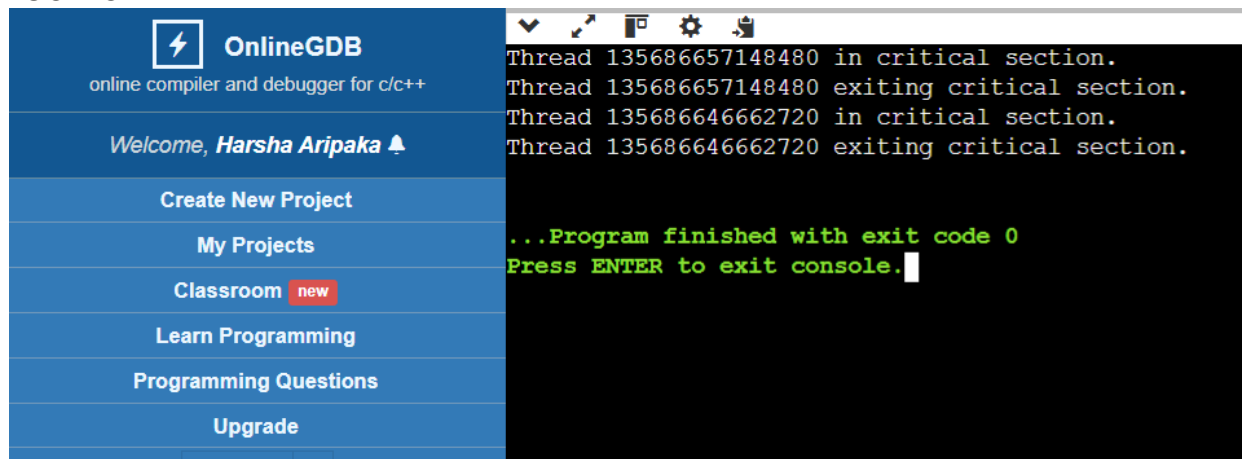
pthread_join(thread2, NULL);

pthread_mutex_destroy(&mutex);

return 0;

}
```

OUTPUT:

The screenshot shows the OnlineGDB web interface. On the left is a blue sidebar with navigation links: 'Create New Project', 'My Projects', 'Classroom' (with a 'new' badge), 'Learn Programming', 'Programming Questions', and 'Upgrade'. The main area on the right has a dark background and displays the program's output in a monospaced font. The output shows two threads, 135686657148480 and 135686646662720, each entering and then exiting a critical section. The program concludes with the message '...Program finished with exit code 0' and 'Press ENTER to exit console.' with a cursor. The OnlineGDB logo and name are visible at the top left of the interface.

```
Thread 135686657148480 in critical section.
Thread 135686657148480 exiting critical section.
Thread 135686646662720 in critical section.
Thread 135686646662720 exiting critical section.

...Program finished with exit code 0
Press ENTER to exit console.
```

## RESULT

The program ensures mutual exclusion in the critical section using mutex locks.