192311035

M.RAVI SAI VINAY

EXPERIMENT-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.

ALGORITHM:-

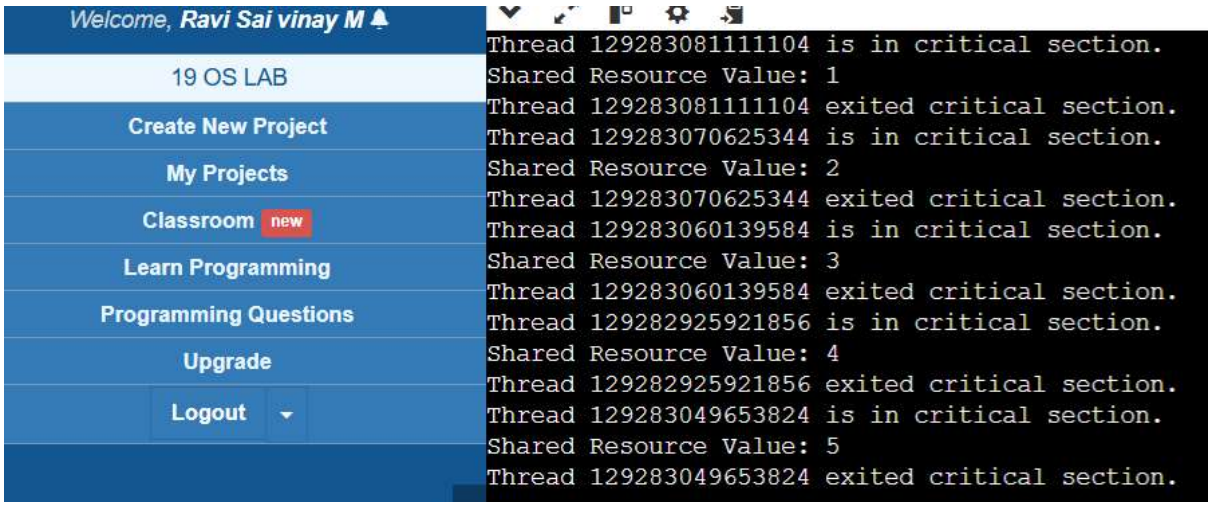
1. **Initialize Mutex**:
    - Use pthread_mutex_t to create a mutex.
    - Initialize it using pthread_mutex_init().
2. **Critical Section**:
    - Lock the mutex using pthread_mutex_lock() before entering the critical section.
    - Perform operations in the critical section.
    - Unlock the mutex using pthread_mutex_unlock() after completing the operations.
3. **Create Threads**:
    - Create multiple threads using pthread_create().
    - Each thread accesses the shared resource (critical section) in a synchronized manner.
4. **Wait for Threads**:
    - Use pthread_join() to wait for all threads to complete execution.
5. **Destroy Mutex**:
    - Use pthread_mutex_destroy() to release the mutex resources.

CODE:-

```
#include <stdio.h>

#include <pthread.h>

#include <unistd.h>


pthread_mutex_t mutex;

int shared_resource = 0;


void *thread_function(void *arg) {

    pthread_mutex_lock(&mutex);

    printf("Thread %ld is in critical section.\n", pthread_self());

    shared_resource++;

    printf("Shared Resource Value: %d\n", shared_resource);

    sleep(1);

    pthread_mutex_unlock(&mutex);

    printf("Thread %ld exited critical section.\n", pthread_self());

    return NULL;

}


int main() {

    pthread_t threads[5];


    pthread_mutex_init(&mutex, NULL);
```

```
    for (int i = 0; i < 5; i++) {
        pthread_create(&threads[i], NULL, thread_function, NULL);
    }

    for (int i = 0; i < 5; i++) {
        pthread_join(threads[i], NULL);
    }

    pthread_mutex_destroy(&mutex);
    return 0;
}
```

OUTPUT:-



RESULT:-

The program successfully implemented process synchronization using mutex locks. Each thread accessed the critical section in a synchronized manner, ensuring no race conditions occurred.