192311035

M.RAVI SAI VINAY

EXPERIMENT-20

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

AIM:-

To implement process synchronization using mutex locks in a C program to ensure that multiple threads access shared resources without causing race conditions

ALGORITHM:-

1. Initialize Mutex:
2. Declare a mutex variable and initialize it using pthread_mutex_init().
3. Create Threads:
4. Use pthread_create() to create multiple threads.
5. Critical Section:
6. Each thread enters the critical section after locking the mutex using pthread_mutex_lock().
7. Modify shared resources inside the critical section.
8. Once done, the mutex is unlocked using pthread_mutex_unlock() to allow other threads access.
9. Wait for Threads:
10. Use pthread_join() to wait for all threads to finish execution.
11. Destroy Mutex:
12. Use pthread_mutex_destroy() to clean up the mutex.

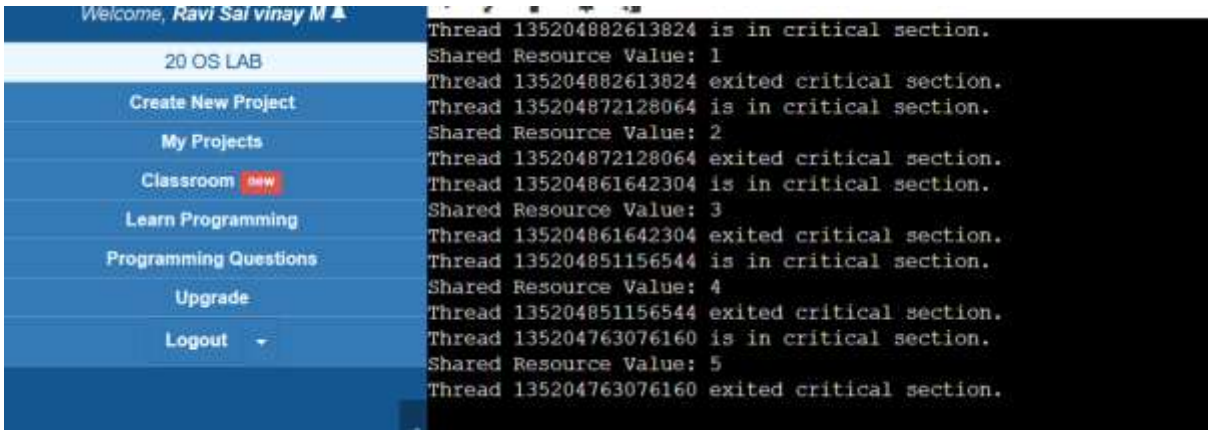
CODE:-

```
#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. Multiple threads accessed the critical section without causing any race conditions, ensuring proper synchronization and mutual exclusion.