

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 <stdlib.h>
#include <pthread.h>

#define NUM_THREADS 5

pthread_mutex_t lock;
int shared_resource = 0;

void* thread_function(void* arg) {
    pthread_mutex_lock(&lock);
    int thread_id = *((int*)arg);
    printf("Thread %d: Accessing shared resource.\n", thread_id);
    shared_resource++;
    printf("Thread %d: Updated shared resource to %d.\n", thread_id, shared_resource);
    pthread_mutex_unlock(&lock);
    return NULL;
}

int main() {
    pthread_t threads[NUM_THREADS];
    int thread_ids[NUM_THREADS];

    pthread_mutex_init(&lock, NULL);

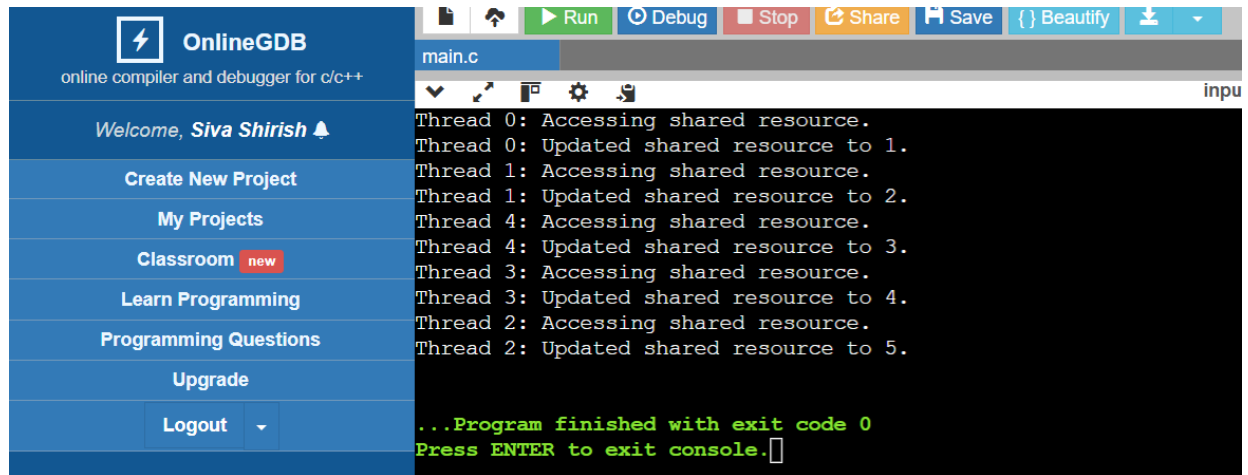
    for (int i = 0; i < NUM_THREADS; i++) {
        thread_ids[i] = i;
        pthread_create(&threads[i], NULL, thread_function, &thread_ids[i]);
    }

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

    pthread_mutex_destroy(&lock);
}
```

```
    return 0;  
}
```

OUTPUT:



The screenshot displays the OnlineGDB web interface. On the left is a sidebar with navigation links: 'Welcome, Siva Shirish', 'Create New Project', 'My Projects', 'Classroom' (marked 'new'), 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area shows a terminal window for 'main.c'. The output of the program is as follows:

```
Thread 0: Accessing shared resource.  
Thread 0: Updated shared resource to 1.  
Thread 1: Accessing shared resource.  
Thread 1: Updated shared resource to 2.  
Thread 4: Accessing shared resource.  
Thread 4: Updated shared resource to 3.  
Thread 3: Accessing shared resource.  
Thread 3: Updated shared resource to 4.  
Thread 2: Accessing shared resource.  
Thread 2: Updated shared resource to 5.  
...Program finished with exit code 0  
Press ENTER to exit console.
```

RESULT

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