K NARENDRA-192321162

1. 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);

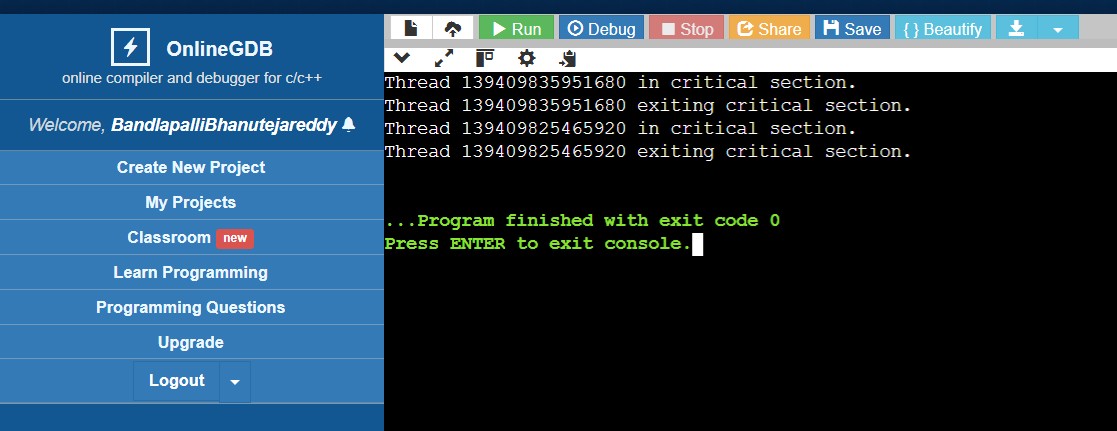
pthread\_create(&thread1, NULL, critical\_section, NULL);

pthread\_create(&thread2, NULL, critical\_section, NULL); pthread\_join(thread1, NULL);

pthread\_join(thread2, NULL); pthread\_mutex\_destroy(&mutex); return 0;

}

OUTPUT:



RESULT

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