NAME: G. GANESH

REG NO: 192373008

EXERICSE-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 to ensure mutual exclusion in a critical section.

Algorithm:

1. **Initialize Mutex Lock:**
    - Declare and initialize a pthread_mutex_t variable for mutual exclusion.
2. **Create Threads:**
    - Define two or more threads representing processes that need to access a shared resource.
3. **Access Critical Section:**
    - Use pthread_mutex_lock() to acquire the mutex lock before entering the critical section.
    - Perform the required operation on the shared resource.
    - Use pthread_mutex_unlock() to release the mutex lock after exiting the critical section.
4. **Execute Threads:**
    - Create threads using pthread_create() and assign them the function that accesses the critical section.
    - Use pthread_join() to wait for threads to complete execution.
5. **Clean Up:**
    - Destroy the mutex lock using pthread_mutex_destroy() after all threads finish execution.

Procedure:

1. Initialize a mutex lock.
2. Define a shared resource, such as a counter variable.
3. Implement thread functions that access the shared resource, ensuring mutual exclusion using mutex locks.
4. Create and execute threads to simulate concurrent access to the shared resource.
5. Observe the results to ensure proper synchronization.
6. Destroy the mutex lock after thread execution.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_THREADS 5
int shared_counter = 0;
pthread_mutex_t mutex;
void *increment_counter(void *param) {
    for (int i = 0; i < 5; i++) {
        pthread_mutex_lock(&mutex);
        shared_counter++;
        printf("Thread %ld incremented counter to %d\n", pthread_self(), shared_counter);
        pthread_mutex_unlock(&mutex);
        sleep(1);
    }
    return NULL;
}
int main() {
    pthread_t threads[NUM_THREADS];
    pthread_mutex_init(&mutex, NULL);
    for (int i = 0; i < NUM_THREADS; i++) {
pthread_create(&threads[i], NULL, increment_counter, NULL);
    }
    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }
    pthread_mutex_destroy(&mutex); // Destroy the mutex
    printf("Final counter value: %d\n", shared_counter);
    return 0;
```

}

Result:

The program demonstrates process synchronization using mutex locks, ensuring that threads access the shared resource sequentially without interference, maintaining data integrity.

Output: