**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 :

Step 1: Include Necessary Libraries: Include the required header files for pthreads and mutex locks.

Step 2: Declare Global Variables: Declare any global variables needed for synchronization, such as mutex variables.

Step 3: Initialize Mutex: In the main function or initialization function, initialize the mutex using pthread\_mutex\_init function.

Step 4: Define Functions: Define functions that represent the actions of threads. These functions should include the critical sections where the mutex lock is acquired and released.

Step 5: Create Threads: In the main function or any other appropriate function, create threads and assign the functions to execute for each thread. Pass NULL or any necessary data as arguments to the functions.

Step 6: Implement Mutex Synchronization: Inside the functions that represent the actions of threads, use pthread\_mutex\_lock to acquire the mutex lock and pthread\_mutex\_unlock to release the lock. This ensures that only one thread can execute the critical section at a time.

Step 7: Join Threads and Cleanup: In the main function or any other appropriate function, wait for the threads to finish using pthread\_join. After the threads have finished their execution, destroy the mutex using pthread\_mutex\_destroy function.

Step 8: Compile and Run: Compile the program using a C compiler with the appropriate flags (for example, -pthread for GCC) to link the pthread library. Then, run the compiled executable to observe the synchronized behavior of threads due to mutex locks.

## OUTPUT :

A screenshot of a computer

Description automatically generated