EXP 19: Design a C program to implement process synchronization using mutex locks.

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

#define NUM_THREADS 5
#define NUM_ITERATIONS 100000

int counter = 0;            // Shared resource
pthread_mutex_t lock;       // Mutex lock

// Function to be executed by threads
void* increment_counter(void* arg) {
  for (int i = 0; i < NUM_ITERATIONS; i++) {
    pthread_mutex_lock(&lock);   // Enter critical section
    counter++;                   // Critical section
    pthread_mutex_unlock(&lock); // Exit critical section
  }
  return NULL;
}

int main() {
  pthread_t threads[NUM_THREADS];

  // Initialize mutex
  pthread_mutex_init(&lock, NULL);
```

```
    // Create threads
    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_create(&threads[i], NULL, increment_counter, NULL);
    }

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

    // Destroy mutex
    pthread_mutex_destroy(&lock);

    printf("Final Counter Value: %d\n", counter);
    return 0;
}
```

Sample Output

```
Final Counter Value: 500000


=== Code Execution Successful ===
```