PROCESS SYNCHRONIZATION

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#define NUM_THREADS 5
#define NUM_ITERATIONS 3
int shared_counter = 0;
pthread_mutex_t lock;
void* thread_function(void* arg) {
    int thread_id = *((int*)arg);
    for (int i = 0; i < NUM_ITERATIONS; i++) {
        pthread_mutex_lock(&lock);
        printf("Thread %d entering critical section.\n", thread_id);
        int temp = shared_counter;
        temp++;
        sleep(1);
        shared_counter = temp;
        printf("Thread %d updated counter to %d\n", thread_id, shared_counter);
        printf("Thread %d leaving critical section.\n\n", thread_id);
        pthread_mutex_unlock(&lock);
        sleep(1);
    }
    pthread_exit(NULL);
}
int main() {
    pthread_t threads[NUM_THREADS];
    int thread_ids[NUM_THREADS];
    if (pthread_mutex_init(&lock, NULL) != 0) {
        printf("Mutex initialization failed!\n");
        return 1;
    }
    for (int i = 0; i < NUM_THREADS; i++) {
        thread_ids[i] = i + 1;
        if (pthread_create(&threads[i], NULL, thread_function, &thread_ids[i]) != 0) {
            printf("Error creating thread %d\n", i + 1);
            return 1;
        }
    }
    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }
    pthread_mutex_destroy(&lock);
    printf("\nFinal value of shared counter: %d\n", shared_counter);
    return 0;
}
```