Motilal Nehru National Institute of TechnologyAllahabad
Prayagraj-211004 [India]

Department of Computer Science & Engineering

Programme Name: B.Tech **Semester: VII** **Branch: Computer Science & Engg.**

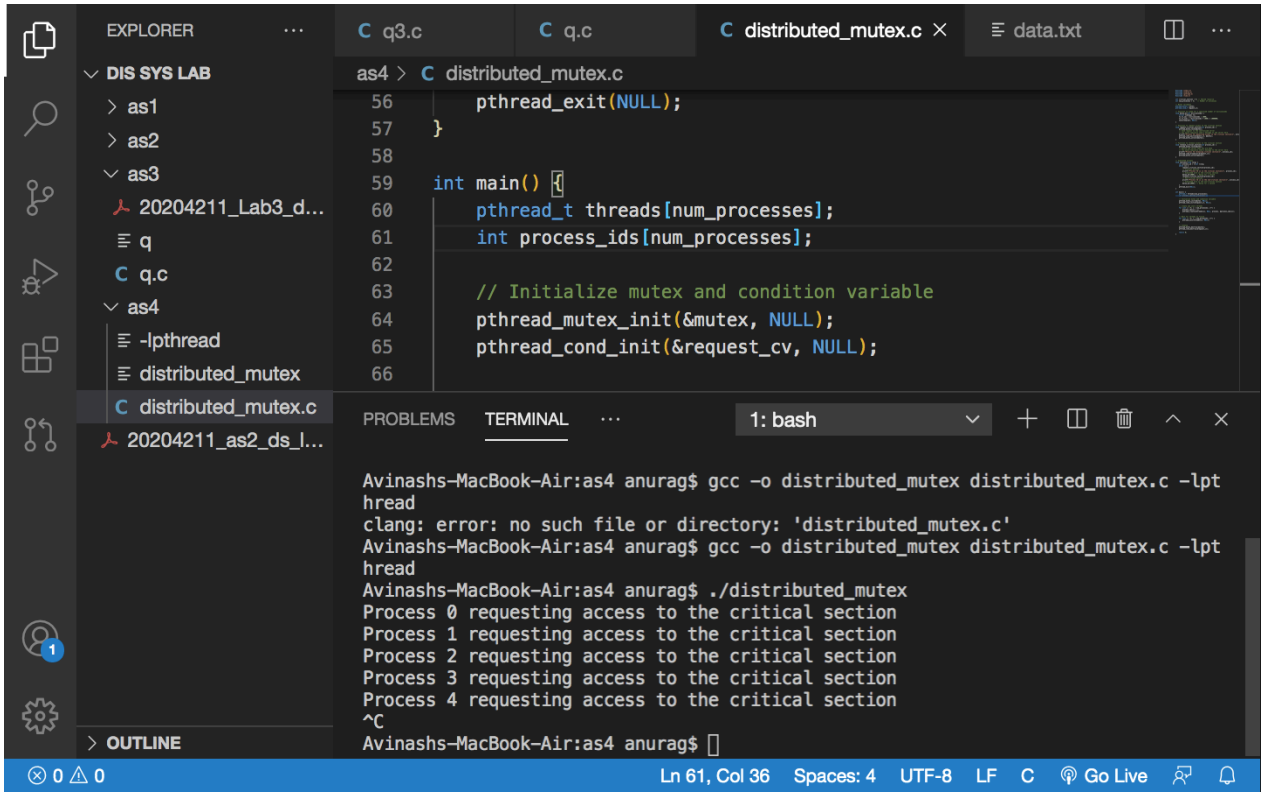
Course Code: CS17201 **Course Name: Distributed Systems (Lab)**

Lab Assignment 4

Lab #	Name of Experiment
4	Simulate the Distributed Mutual Exclusion.

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <time.h>

int critical_section = 0; // Shared resource
int num_processes = 5;   // Number of processes

// Mutex variables
pthread_mutex_t mutex;
pthread_cond_t request_cv;

// Function to sleep for a specified number of milliseconds
void delay_ms(int milliseconds) {
    struct timespec ts;
    ts.tv_sec = milliseconds / 1000;
    ts.tv_nsec = (milliseconds % 1000) * 1000000;
    nanosleep(&ts, NULL);
}

// Function to request access to the critical section
void request_critical_section(int process_id) {
    pthread_mutex_lock(&mutex);
    // Send request to the centralized server
    // You would typically send a message to the server here
    printf("Process %d requesting access to the critical section\n", process_id);
    pthread_cond_wait(&request_cv, &mutex);
    pthread_mutex_unlock(&mutex);
}

// Function to release access to the critical section
void release_critical_section(int process_id) {
    pthread_mutex_lock(&mutex);
    // Notify the server that you are done
    // You would typically send a message to the server here
    printf("Process %d releasing critical section\n", process_id);
    pthread_cond_broadcast(&request_cv);
    pthread_mutex_unlock(&mutex);
}
```

```c
// Simulated process
void *process(void *arg) {
    int process_id = *(int *)arg;
    while (1) {
        request_critical_section(process_id);
        // Critical Section
        printf("Process %d is in the critical section\n", process_id);
        // Simulated work in the critical section
        delay_ms(1000); // Delay for 1 second
        release_critical_section(process_id);
        // Non-critical Section
        printf("Process %d is in the non-critical section\n", process_id);
        // Simulated work in the non-critical section
        delay_ms(1000); // Delay for 1 second
    }
    pthread_exit(NULL);
}

int main() {
    pthread_t threads[num_processes];
    int process_ids[num_processes];

    // Initialize mutex and condition variable
    pthread_mutex_init(&mutex, NULL);
    pthread_cond_init(&request_cv, NULL);

    // Create and start threads
    for (int i = 0; i < num_processes; i++) {
        process_ids[i] = i;
        pthread_create(&threads[i], NULL, process, &process_ids[i]);
    }

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

    // Cleanup
    pthread_mutex_destroy(&mutex);
    pthread_cond_destroy(&request_cv);

    return 0;
}
```


EXPLORER
DIS SYS LAB
as1
as2
as3
20204211_Lab3_d...
q
q.c
as4
-lpthread
distributed_mutex
distributed_mutex.c
20204211_as2_ds_l...
OUTLINE
q3.c
q.c
distributed_mutex.c
data.txt
as4 > distributed_mutex.c
56 pthread_exit(NULL);
57 }
58
59 int main() {
60 pthread_t threads[num_processes];
61 int process_ids[num_processes];
62
63 // Initialize mutex and condition variable
64 pthread_mutex_init(&mutex, NULL);
65 pthread_cond_init(&request_cv, NULL);
66
PROBLEMS
TERMINAL
1: bash
Avinashs-MacBook-Air:as4 anurag$ gcc -o distributed_mutex distributed_mutex.c -lpthread
clang: error: no such file or directory: 'distributed_mutex.c'
Avinashs-MacBook-Air:as4 anurag$ gcc -o distributed_mutex distributed_mutex.c -lpthread
Avinashs-MacBook-Air:as4 anurag$ ./distributed_mutex
Process 0 requesting access to the critical section
Process 1 requesting access to the critical section
Process 2 requesting access to the critical section
Process 3 requesting access to the critical section
Process 4 requesting access to the critical section
^C
Avinashs-MacBook-Air:as4 anurag$
0 0
Ln 61, Col 36
Spaces: 4
UTF-8
LF
C
Go Live