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

Department of Computer Science & Engineering

Programme Name: B.Tech Course Code: CS17201

Semester: VII Branch: Computer Science & Engg. Course Name: Distributed Systems (Lab)

**Lab Assignment 4**

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

#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);

}

// 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;

}

