Larbi Ben M'hidi University – Oum El Bouaghi

Faculty of Exact Sciences and Nature and Life Sciences

Department of Mathematics and Computer Science

Module: Operating Systems II

Lecturer: Dr. Moustafa Sadek KAHIL

2025 – 2026

Practical Work Series 2

Exercise 1:

Write a C program that creates three parallel threads, each performing a specific operation on a given array T of size 10. The operations are as follows:

- **Thread 1**: Calculates the average of the array's element.
- **Thread 2**: Searches for a specific element in array T and returns its position (if it exists).
- **Thread 3**: Multiplies all of the array's elements by a constant and displays the resulting array.

Exercise 2:

The goal of this exercise is to simulate a mutual exclusion system between three threads using a global lock. The lock is represented by a global variable `int lock` which can be 0 if the critical resource is free and 1 if it is occupied.

Each thread must perform the following steps:

1. **Test and wait:** Check if the lock is 0. If not, wait for it to become free.
2. **Take the lock:** Once the lock is 0, the thread must set it to 1 to signal that it is entering the critical section.
3. **Execute the critical section:** Simulate a task in the critical section (display a message indicating that the thread is in the critical section).
4. **Release the lock:** Set the lock to 0 to allow other threads to enter the critical section.

Exercise 3 :

Simulate mutual exclusion between two threads using a binary semaphore. Each thread must perform the following actions:

1. Display a message requesting access to the critical section.
2. Wait for the semaphore to become available.
3. Enter the critical section, display a message, and perform a simulated task (e.g., sleep).
4. Exit the critical section, display a message, and release the semaphore.

Exercise 4:

Using a counting semaphore, simulate a system where 5 threads simultaneously access a shared resource, but with a limit of 2 threads at a time in the critical section.