Operating Systems
INF333

TP06
Mutexes

Eda Bahar 02/05/2025
ebahar@gsu.edu.tr
edaabahar@gmail.com

About this TP

- In this TP, you will be learning:
  - Mutexes

Introduction

- What is concurrency? Concurrency refers to the execution of multiple tasks at the same time. This can occur in **multithreaded programs** or in **multi-process environments**.
- Why do we need synchronization? When multiple threads access shared resources (e.g., memory, files, or databases) without coordination, it can lead to:
  - **Race conditions** (unpredictable behavior)
  - **Data corruption**
  - **Inconsistent states**
- **The concept of mutual exclusion:** Mutual exclusion (mutex) ensures that only one thread can access a shared resource at a time, preventing conflicts.

Mutex?

- A **mutex** (short for **"mutual exclusion"**) is a synchronization primitive that allows only one thread to access a resource at a time.
- How does a mutex work?
  - A thread **locks** the mutex before accessing a shared resource.
  - Other threads trying to access the same resource **must wait** until the mutex is **unlocked**.
  - The original thread **unlocks** the mutex when it is done.

Mutex Implementation

- lock() -> Locks the mutex, blocking other threads.
- unlock() -> Releases the mutex, allowing other threads to proceed.
- try_lock() -> Attempts to lock the mutex without blocking.

Example

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

pthread_mutex_t lock; // Declare a mutex
int counter = 0;

void* increment(void* arg) {
    for (int i = 0; i < 1000000; i++) {
        pthread_mutex_lock(&lock); // Lock the mutex
        counter++;                 // Critical section
        pthread_mutex_unlock(&lock); // Unlock the mutex
    }
    return NULL;
}

int main() {
    pthread_t t1, t2;
    pthread_mutex_init(&lock, NULL); // Initialize the mutex

    pthread_create(&t1, NULL, increment, NULL);
    pthread_create(&t2, NULL, increment, NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);

    pthread_mutex_destroy(&lock); // Destroy the mutex

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

Problems with Mutexes

- **Deadlocks**: Occur when two or more threads wait for each other to release a lock.
- **Solution**: Use lock ordering. Always lock mutexes in the same order.