

# Project 3 – Traffic Control Simulation

**Course:** Operating Systems

**Student:** Abdultawwab Tello

## 1. Overview

This project implements a multi-threaded traffic-control simulation using POSIX threads (pthread).

Each car is represented by a thread and follows these steps:

1. ArriveIntersection()
2. CrossIntersection()
3. ExitIntersection()

Synchronization is enforced using:

- A **head-of-line lock** per direction (only front car may proceed)
- **Quadrant locks** (modeled with a direction-sharing readers/writers design)
- A **global state lock** for coordination of waiting rules
- A **print lock** to serialize console output

The simulation prints timestamps for:

- Arrival
- Crossing
- Exiting

## 2. Traffic Rules Implemented

### Stop Sign Rule

Each car must:

- Stop for **2 seconds**
- Wait until it reaches the front of its lane
- Yield to any earlier-arriving car from other directions that is still waiting

### Quadrant Rule

Each car acquires the required intersection quadrants based on its direction and turn:

- Right turn -> 1 quadrant
- Straight -> 2 quadrants
- Left turn -> 3 quadrants

Cars from the **same direction** may share quadrants (readers-writers style), which improves throughput.

## No Deadlock

Quadrants are always acquired in fixed order (0 -> 1 -> 2 -> 3).

## 3. Notes

- Timing is simulated using `gettimeofday()` and `usleep()`.
- Shared state updates use mutex + condition variable.
- Printing is synchronized to avoid interleaving.
- Each car runs in its own thread and follows realistic timing constraints.