Topic:

Traffic Intersection

Subject:

Operating System

Submitted by:

Anns Shahzad (2019-CS-601)

Muhammad Hassan Ijaz (2019-CS-617)

Submitted to:

Mr. Nadeem Iqbal

Department of Computer Science

University of Engineering & Technology, Lahore (New Campus)

Introduction:

Intersection is an area shared by two or more roads. This area is designated for the vehicles to turn to different directions to reach their desired destinations. Traffic intersections are complex locations on any highway. This is because vehicles moving in different direction wan to occupy same space at the same time.

Description:

This program runs a simulation of an intersection with emphasis on the use of threads and semaphores. The program will take in user input to determine how long to run and how many cars to simulate, and then create a unique thread for each car.

As these cars run through various stages of an intersection to either turn right, go straight, or turn left, the status of each car is printed out to monitor correct activity.

At the end of the simulation the minimum, maximum, average, and total time spent at the intersection is printed out.

Objective:

The main objective of this project is to calculate the minimum, maximum, average, and total time spent at the traffic intersection. This will help in future constructions or modifications of traffic intersection. Like, how large must they be built so the average amount of cars can pass through there easily with less probability of traffic blockage and accidents.

How Program Will Work?

This software is run via the command line and takes in two required arguments:

Argument 1: Time to live in seconds (how long the intersection simulation will run).

Argument 2: The number of cars to create and run through the intersection.

Features of this Program:

Following is some features that will be provided by this program:

⇒ **Avoiding deadlock:**

- Break circular wait by forcing each car to obtain all its resources before moving through the intersection.

- Four things must occur for deadlock to be possible:
  - Mutual exclusion.
  - Hold and wait.
  - No Preemption.
  - Circular Wait.

⇒ **Prevent Accidents:**

- A car must be able to move into all the quadrants of the intersection it needs before it starts to go.
- If it cannot grab all these sections, it will try again later.
- This ensures that no two cars are in the same position at the same time.

⇒ **Improve traffic flow:**

- Multiple cars can be within the intersection assuming they do not collide when trying to lock all their resources. at worst case only one car will be in the intersection under a rare circumstance.
- At best case there can be up to 4 fours in the intersection at a given time.

⇒ **Preserve car Order**:

- We will use a Queue to accomplish this.
- There is a big decision in deciding if we want to implement four queues for each possible direction of arrival or having one large queue that all the cars waited in.
- We will go with one large queue because of fairness.

Tools to Implement:

⇒ Threads.
⇒ Semaphore.
⇒ Processes.
⇒ Makefile.