CENG334 - Introduction to Operating Systems

Spring 15/16

Programming Assignment 2

Synchronization – Crossroad Simulation

Özcan Dülger – odulger(at)ceng.metu.edu.tr

Due To: 05.05.2016 - 23:55:50

1. Objectives

In this assignment you are going to practice on synchronization between threads using C++. Your task is to implement a four-way crossroads simulation system to synchronize the movements of vehicles on different directions. You are allowed to use semaphores, condition variables and mutexes for the synchronization between threads. You can use one of them or all of them.

Keywords: Four way *Crossroad Simulation, Thread, Semaphore, Mutex, Condition Variable*

2. Problem Definition

In four-way crossroad simulation, there are vehicles that come from a direction to the intersection and go to another direction. There may be more than one intersection. The vehicles can pass through one or more intersections on their simulation path. Assume the connection between the intersections is not relevant. The outgoing direction of the vehicle cannot be same as the incoming direction. The vehicles travel through each intersection in four different directions which are west, south, east and north. The illustration of a single intersection is given in Figure 1.

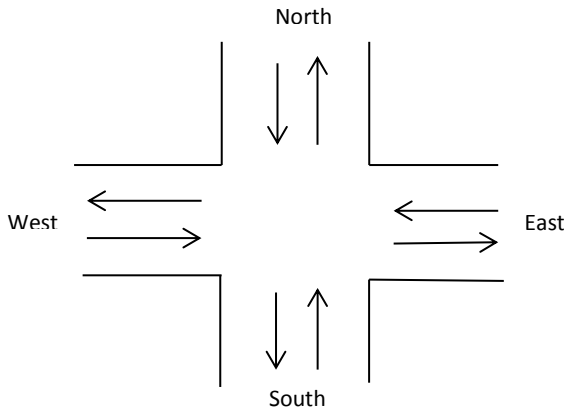

Figure 1: An illustration of a single intersection

There are two modes in the simulation which are day and night modes.

Day Mode:

In the day mode, there are traffic lights on each incoming direction. When the red light is active the vehicles must stop and wait until the green light is active. When the green light is active, the vehicles can enter and pass through the intersection if there is no vehicle in the intersection (remaining

vehicles from previous green directions). Only one green light is active at a time. The turning on order of the green lights changes in the anti-clockwise manner, which will be controlled by a specific thread. The illustration of the day mode is given in Figure 2.

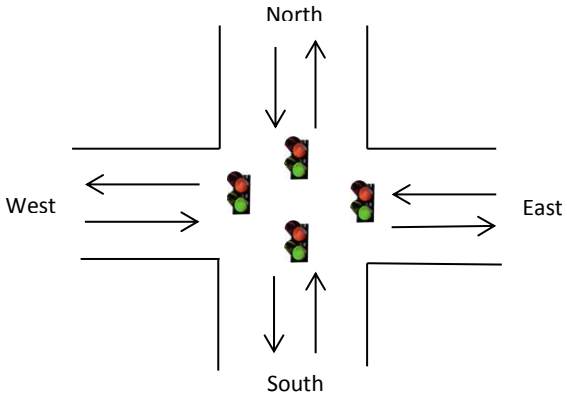

Figure 2: An illustration of the day mode

Night Mode:

In the night mode, there is no traffic light on the intersection but there are some rules about the movements of the vehicles on the intersection. A vehicle can enter an intersection only if there is currently no vehicle on the intersection with a conflicting move. Conflicting rules described as follows:

Rule 0 (vehicles coming from same direction):

Vehicles entering the intersection from the same direction do not conflict.

Rule 1 (vehicles turning right):

Vehicles turning right only conflict with vehicles coming from another direction and outgoing direction is the same. An example illustration of this rule is given in Figure 3. In this example, the vehicle comes from south and goes to the east direction. The arrows show the movements of the vehicles in the intersection. The red ones must not enter the intersection but one of the green ones can enter. The blue one is the vehicle which turns right in the example.

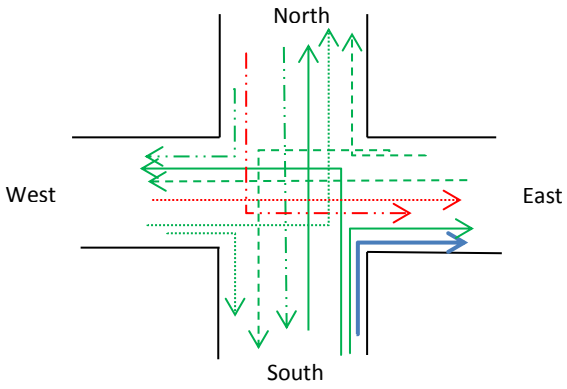

Figure 3: An example illustration for the first rule

Rule 2 (straight passing vehicles):

Straight passing vehicles conflict with vehicles in orthogonal direction, vehicles with left turns, vehicles with right turn to the same direction. An example illustration of this rule is given in Figure 4.

In this example, the vehicle comes from south and goes to the north direction. The meanings of the arrows are as in the previous rule.

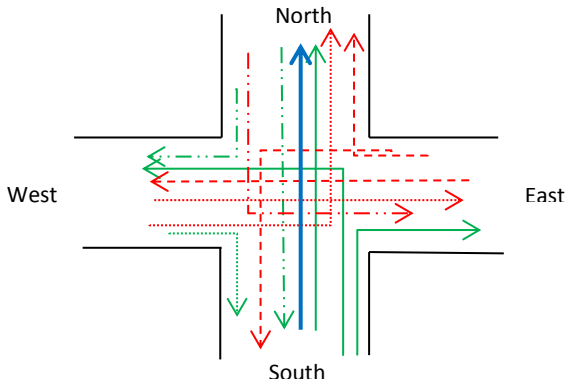

Figure 4: An example illustration for the second rule

Rule 3 (vehicles turning left):

Vehicles turning left conflict with every vehicle, except the vehicles turning right in opposite direction. An example illustration of this rule is given in Figure 5. In this example, the vehicle comes from south and goes to the west direction. The meanings of the arrows are as in the previous rules.

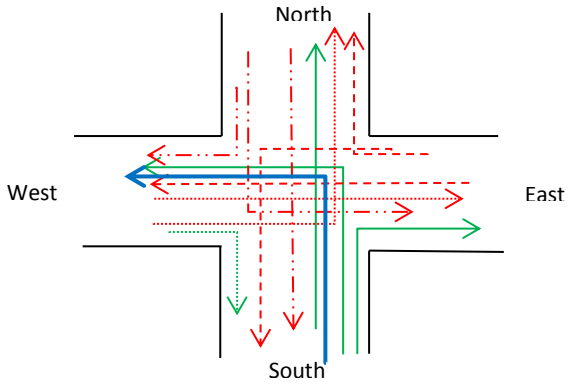

Figure 5: An example illustration for the third rule

There may be more than one vehicle that waits intersection to be available on different incoming directions. If there are more than one vehicle are available to enter the intersection, the vehicles must enter according to the first come first serve (FCFS) rule.

The following is a summary of checks for entering and exiting intersection for a vehicle on the day mode:

for enter():

1. if red light turns on → wait until the green light turns on
2. else if green light turns on and vehicle(s) on intersection→ wait until the intersection becomes free and enter if green light is still active. (Note that vehicles on intersection here are from previous green lights. Once intersection becomes free of them, all vehicles on current direction can enter the intersection)
3. else → enter

for exit():

1. inform → exit

The following is a summary of checks for entering and exiting intersection for a vehicle on the night mode:

for enter():

1. if intersection is empty → enter
2. if vehicle(s) on intersection →
    2.1. if all vehicles on its incoming direction → enter
    2.2. else →
        2.2.1. if intersection is not available according to the movement rules → wait until it becomes available and enter according to the FCFS rule (among all vehicles that can enter the intersection)
        2.2.2. else → enter

for exit()

1. inform → exit

3. Tasks

Your task is to write a four-way crossroads simulation. In this simulation, the group of vehicle threads on the crossroads will be synchronized. You are allowed to use only pthread.h library and semaphore.h library for the threads, semaphores, condition variables and mutexes. Your implementation should include the following:

- Create a thread for each vehicle and synchronize the vehicles ensuring the intersection rules defined in the day and night mode. Create the threads in detached mode in order to release their resources without waiting all the threads to finish their job.
- Read the number of intersection, mode and the duration of the green lights from the command line as *./hw2 N mode duration*, where N is the number of intersection, mode is 0 (day mode) or 1 (night mode) and duration is the coefficient for the duration of the green lights on the intersection.
- In case mode is day mode, create a traffic light thread for each intersection and synchronize the traffic lights with a proper way. Create the threads in detached mode.
- Read the information of the vehicles from stdin (standard input) each on a separate line. Its content is explained in section 4 in detail.
- Write the output of each operation to the stdout (standard output). Functions namely writeOutput will be given in writeOutput.h file to write the outputs of the simulation. Do not modify this file. Read the comment lines in the function carefully and set the parameters for the appropriate output.
- The content template for hw2.cpp files will be provided. There is CrossRoad class which consists of data members and procedures for the crossroad operations on the day and night mode. The vehicles will call Enter function before entering the intersection. The vehicles will call Exit function before they exit the intersection. SetGreenLight function turns green light on according to the rules in the day mode. *N* is the number of intersections and *lightState* represents the direction of the active green lights on each intersection. The value of directions is given as an enumeration in the code. Set *lightState* as -1 in the night mode. Do not modify the definitions of public functions and data members of the class. But you can add any function or data member as a private member to the class.

- Arriving operation of the vehicles to any intersection will take (arriveTime*1000) microseconds. Moving operation on the intersection will take (moveTime*1000) microseconds. Use usleep function that is defined in unistd.h library. It gets its input as a microsecond value.
- On the day mode, initially the green light on the west incoming direction is active. The duration of the green traffic lights is (duration*1000).
- Read and use the underlined information carefully; otherwise your evaluation may end up with unexpected results.
- Pseudo-code for simulation:

traffic light thread:

```
while (true)
{
    usleep(duration*1000)
    SetGreenLight(intersectionCode)
}
```

vehicle thread:

```
for all paths of the vehicle
{
    usleep(arriveTime*1000)
    writeOutput(0, vehicleCode, intersectionCode, From, To)
    Enter(vehicleCode,  intersectionCode, From, To)
    usleep(moveTime *1000)
    Exit(vehicleCode,  intersectionCode, From, To)
}
finish thread
```

main thread:

```
read intersection number, mode and duration of green lights
if (mode is 0)
{
    start a traffic light thread for each intersection
}
while (read vehicle request r)
{
    start a vehicle thread for r
}
end of request input
wait until last vehicle finishes
exit
```

4. Inputs

You will read the vehicle information from stdin (standard input) as a single line. An example vehicle input is as follows:

```
1 0 10
1 EAST WEST
3 SOUTH NORTH
2 WEST EAST
NR
2 10 10
1 SOUTH WEST
2 EAST NORTH
3 EAST SOUTH
NR
3 21 20
1 NORTH EAST
NR
4 33 30
1 EAST SOUTH
3 EAST WEST
NR
5 13 10
2 EAST NORTH
1 WEST NORTH
NR
```

The parameters are separated with a single space character. The data of a vehicle are given in multiple lines. In the first line, the first parameter is the vehicle code of the vehicle. This is a unique integer value. The second parameter is the relative arrival time (arriveTime) of the vehicle to any intersection. The third parameter is the moving time (moveTime) of the vehicle on the intersection. In the following lines, the paths of the vehicle are given up to the first following line which contains NR. In these lines, the first parameter is the intersection code. The second parameter is the incoming direction of the vehicle. The third parameter is the outgoing direction of the vehicle. Intersection codes are unique integer value.

5. Outputs

Write the output of each operation to the stdout (standard output). A function namely writeOutput will be given in homework files to write the outputs of the simulation. Read the comment lines in the function carefully and set the parameters for the appropriate output. Output order is important, so be sure that you call the writeOutput function in the correct step with the appropriate parameters. Do not modify writeOutput.h file. An example output of the input that is given section 4 for the day mode is as follows (The value of duration is 10):

```
Vehicle = 1 arrived the intersection 1 from east towards the west
Green Light becomes active on the south incoming direction of the intersection 1
Green Light becomes active on the south incoming direction of the intersection 3
Green Light becomes active on the south incoming direction of the intersection 2
Vehicle = 2 arrived the intersection 1 from south towards the west
Vehicle = 2 entered the intersection 1 from south towards the west
Vehicle = 5 arrived the intersection 2 from east towards the north
```

Green Light becomes active on the east incoming direction of the intersection 1
Green Light becomes active on the east incoming direction of the intersection 2
Vehicle = 2 exited the intersection 1 from south towards the west
Green Light becomes active on the east incoming direction of the intersection 3
Vehicle = 1 entered the intersection 1 from east towards the west
Vehicle = 5 entered the intersection 2 from east towards the north
Vehicle = 3 arrived the intersection 1 from north towards the east
Vehicle = 2 arrived the intersection 2 from east towards the north
Green Light becomes active on the north incoming direction of the intersection 2
Green Light becomes active on the north incoming direction of the intersection 1
Green Light becomes active on the north incoming direction of the intersection 3
Vehicle = 5 exited the intersection 2 from east towards the north
Vehicle = 1 exited the intersection 1 from east towards the west
Vehicle = 3 entered the intersection 1 from north towards the east
Vehicle = 1 arrived the intersection 3 from south towards the north
Vehicle = 4 arrived the intersection 1 from east towards the south
Green Light becomes active on the west incoming direction of the intersection 2
Green Light becomes active on the west incoming direction of the intersection 3
Green Light becomes active on the west incoming direction of the intersection 1
Vehicle = 5 arrived the intersection 1 from west towards the north
Vehicle = 3 exited the intersection 1 from north towards the east
Green Light becomes active on the south incoming direction of the intersection 3
Green Light becomes active on the south incoming direction of the intersection 2
Vehicle = 1 entered the intersection 3 from south towards the north
Green Light becomes active on the south incoming direction of the intersection 1
Green Light becomes active on the east incoming direction of the intersection 3
Green Light becomes active on the east incoming direction of the intersection 2
Vehicle = 1 exited the intersection 3 from south towards the north
Green Light becomes active on the east incoming direction of the intersection 1
Vehicle = 4 entered the intersection 1 from east towards the south
Vehicle = 1 arrived the intersection 2 from west towards the east
Vehicle = 2 entered the intersection 2 from east towards the north
Green Light becomes active on the north incoming direction of the intersection 3
Green Light becomes active on the north incoming direction of the intersection 2
Green Light becomes active on the north incoming direction of the intersection 1
Vehicle = 2 exited the intersection 2 from east towards the north
Green Light becomes active on the west incoming direction of the intersection 3
Green Light becomes active on the west incoming direction of the intersection 2
Green Light becomes active on the west incoming direction of the intersection 1
Vehicle = 1 entered the intersection 2 from west towards the east
Vehicle = 2 arrived the intersection 3 from east towards the south
Vehicle = 4 exited the intersection 1 from east towards the south
Vehicle = 5 entered the intersection 1 from west towards the north
Green Light becomes active on the south incoming direction of the intersection 3
Green Light becomes active on the south incoming direction of the intersection 2
Green Light becomes active on the south incoming direction of the intersection 1
Vehicle = 1 exited the intersection 2 from west towards the east
Vehicle = 5 exited the intersection 1 from west towards the north
Green Light becomes active on the east incoming direction of the intersection 3
Vehicle = 2 entered the intersection 3 from east towards the south
Green Light becomes active on the east incoming direction of the intersection 2
Green Light becomes active on the east incoming direction of the intersection 1

Green Light becomes active on the north incoming direction of the intersection 3
Green Light becomes active on the north incoming direction of the intersection 1
Vehicle = 2 exited the intersection 3 from east towards the south
Green Light becomes active on the north incoming direction of the intersection 2
Green Light becomes active on the west incoming direction of the intersection 3
Green Light becomes active on the west incoming direction of the intersection 1
Green Light becomes active on the west incoming direction of the intersection 2
Vehicle = 4 arrived the intersection 3 from east towards the west
Green Light becomes active on the south incoming direction of the intersection 3
Green Light becomes active on the south incoming direction of the intersection 1
Green Light becomes active on the south incoming direction of the intersection 2
Green Light becomes active on the east incoming direction of the intersection 1
Green Light becomes active on the east incoming direction of the intersection 3
Green Light becomes active on the east incoming direction of the intersection 2
Vehicle = 4 entered the intersection 3 from east towards the west
Green Light becomes active on the north incoming direction of the intersection 1
Green Light becomes active on the north incoming direction of the intersection 2
Green Light becomes active on the north incoming direction of the intersection 3
Green Light becomes active on the west incoming direction of the intersection 1
Green Light becomes active on the west incoming direction of the intersection 3
Green Light becomes active on the west incoming direction of the intersection 2
Vehicle = 4 exited the intersection 3 from east towards the west

An example output of the input that is given section 4 for the night mode is as follows:

Vehicle = 1 arrived the intersection 1 from east towards the west
Vehicle = 1 entered the intersection 1 from east towards the west
Vehicle = 2 arrived the intersection 1 from south towards the west
Vehicle = 1 exited the intersection 1 from east towards the west
Vehicle = 2 entered the intersection 1 from south towards the west
Vehicle = 1 arrived the intersection 3 from south towards the north
Vehicle = 1 entered the intersection 3 from south towards the north
Vehicle = 5 arrived the intersection 2 from east towards the north
Vehicle = 5 entered the intersection 2 from east towards the north
Vehicle = 2 exited the intersection 1 from south towards the west
Vehicle = 1 exited the intersection 3 from south towards the north
Vehicle = 1 arrived the intersection 2 from west towards the east
Vehicle = 1 entered the intersection 2 from west towards the east
Vehicle = 3 arrived the intersection 1 from north towards the east
Vehicle = 3 entered the intersection 1 from north towards the east
Vehicle = 5 exited the intersection 2 from east towards the north
Vehicle = 2 arrived the intersection 2 from east towards the north
Vehicle = 2 entered the intersection 2 from east towards the north
Vehicle = 1 exited the intersection 2 from west towards the east
Vehicle = 4 arrived the intersection 1 from east towards the south
Vehicle = 5 arrived the intersection 1 from west towards the north
Vehicle = 2 exited the intersection 2 from east towards the north
Vehicle = 3 exited the intersection 1 from north towards the east
Vehicle = 4 entered the intersection 1 from east towards the south
Vehicle = 2 arrived the intersection 3 from east towards the south
Vehicle = 2 entered the intersection 3 from east towards the south
Vehicle = 2 exited the intersection 3 from east towards the south

Vehicle = 4 exited the intersection 1 from east towards the south
Vehicle = 5 entered the intersection 1 from west towards the north
Vehicle = 5 exited the intersection 1 from west towards the north
Vehicle = 4 arrived the intersection 3 from east towards the west
Vehicle = 4 entered the intersection 3 from east towards the west
Vehicle = 4 exited the intersection 3 from east towards the west

6. Specifications

- Please read the specifications carefully!
- Your homework must be written in C++ on Linux. No other platforms are accepted.
- You are allowed to use only pthread.h library and semaphore.h library for the threads, semaphores, condition variables and mutexes. Your solution should not busy wait.
- Your solutions will be evaluated as a black box technique by using differently many inputs. Output order is important, so be sure that you call writeOutput function in the correct step with the appropriate parameters.
- The grades of the bad solutions will cut off. The non-terminated solutions will get zero from the corresponding input.
- Partial grades will be given as:
    1. Only day mode is implemented → Max 40 pts
    2. In addition, night mode implemented without FCFS rule → Max 75 pts
    3. In addition to 1 or 2, FCFS rule is implemented → + 25 pts.
- Everything you submit should be your own work.
- Please follow the course page on newsgroup (cow) for any update and clarification.
- Please ask your questions related to the homework on cow instead of email in order to your friends, who may face with same problem, can see your questions and answers.
- Your programs will be compiled with g++ and run on the department inek machines. No other platforms/g++ versions etc. will be accepted so check that your code works on ineks before submitting it.
- Sharing and copying any piece of code from each other and INTERNET is strictly forbidden. Both the sharing one and the copying one will be counted as cheated.
- Use sufficient comment lines in your algorithm in order to explain your solution clearly.

7. Submission

Submission will be done via COW. Create a tar.gz file named hw2.tar.gz that contains hw2.cpp and a makefile. Makefile should create executables named hw2. Your implementations will be compiled using the command chain:

```
$ tar -xf hw2.tar.gz
$ make
```

The tar file should not contain any directories!!

May it be easy.