Assignment 4

Ali Asghar Yousuf

ay06993

1 Documentation

My implementation uses 1 mutex locks and 5 condition variables to ensure maximum concurrency. No more than 3 cars should be present on the street at any instance; let us say it is illegal. Subsequent cars can only wait until any one of the 3 cars leaves:

I have done this by using the `cars_on_street` variable. The conditions are present in `incoming_enter` and `outgoing_enter` function that put the entering car on wait if there are more than three cars on the street at a given time. Then in `incoming_leave` and `outgoing_leave`, I have put a signal that notifies the incoming cars that they can now enter if `cars_on_street` < 3.

We assume the street is too narrow for cars to be incoming and outgoing simultaneously. So there will be either only incoming cars or only outgoing cars travelling at a time. However, the streaming should keep running without deadlock in either direction irrespective of how cars arrive.

In `incoming_enter` function, I have added a variable that checks if there are outgoing cars present on the street at the moment. If there are, the incoming car would go to wait. In `outgoing_leave` function, I have added a signal that checks if all outgoing cars have left the street and signals the waiting incoming thread. All this is done between mutex calls to prevent race condition. Similar mechanism is applied for outgoing cars.

After every 7th car leaves, the street becomes unusable and has to be repaired. Cars do not enter the street unless it is ready to use. Only the street thread is allowed to repair the street.

In the `street_thread` function there's a wait condition that checks if `cars_since_repair` is less than 7, if it is not, then it calls repair. In `incoming_leave` and `outgoing_leave`, there is a condition that checks if `cars_since_repair` are 7, then it signals the repair function. In `incoming_enter` and `incoming_leave` also there is a condition that put the arriving car to wait if `cars_since_repair` > 7.