May 3rd, 2024

Homework #3:

Semaphore and shared memory

Objective:

Two parking attendants supervise a parking lot with a capacity of **four places for pickups** and **eight places for automobiles**. Various vehicles, such as automobiles and pickups, arrive at the parking lot entrance. Categories of vehicles arriving at the parking entrance should be contingent upon **a random generator** (For instance, a singular numerical value can symbolize an automobile, whereas a dual numerical value signifies a pickup). Only one vehicle can enter the parking system. Upon the driver's arrival at the parking lot entrance, the valets are present and ready to park the vehicle in the designated parking spot. Each parking officer is assigned to a specific type of vehicle. There is a person responsible for overseeing the pickup park. The other individual is responsible for managing the automobile park. They would personally retrieve several cars from a 'temporary parking lot' with four places for pickups and eight places for automobiles, where car owners leave their vehicles with the keys inside. Two threads must be created to fulfil the task for every vehicle type: **carOwner()** and **carAttendant()**. The **carOwner()** thread will represent the car owner, while the **carAttendant()** thread will represent the valet. The synchronization of these threads should be achieved by utilizing semaphores. Every owner must confirm the presence of a temporary parking space for their automobile or pickup while waiting for the parking attendant to park it in the designated parking area. The owner will exit the parking lot if no temporary parking space is available. To accurately track the temporary spots, one must utilize two counter variables called **mFree_automobile** and **mFree_pickup** and ensure their appropriate protection. The parking attendant will be stationed at the 'temporary parking lot' to wait for car owners and will also update the availability status of parking spaces. Please ensure that the synchronization process utilizes semaphores named **"newPickup"**, **"inChargeforPickup"**; **"newAutomobile"**, **"inChargeforAutomobile"** and that these semaphores are initialized correctly.

Test Scenario (Expected Results):

1. **Scenario Overview:**

- There are separate spots in the parking lot for cars and pickups.
- A random number is generated at the entrance (odd: car, even: pickup).
- Only one vehicle can enter the system at a time.

2. **Workflow:**

- `carOwner()` represents the vehicle owner, and `carAttendant()` represents the valet.
- Threads are synchronized using semaphores.

3. **Details:**

- Owners must confirm their vehicle's parking availability.
- The occupancy of the temporary parking area and vehicle types are handled correctly.

- Semaphores are initialized and used correctly.

Grading Criteria:

1) No compilation: -100 points
2) Missing Makefile or Makefile without "make clean": -30 points
3) Each memory leak that we encounter: -20 points
4) No report was submitted: -100 points (You must test and demonstrate every step.)
5) Failure in one of the tasks: -10 points

Late submissions will not be accepted.
Deadline: May 20th at 09:00
Good Luck!