GEBZE TECHNİCAL UNIVERSITY  
  
CSE344

## System Programming

# Homework 3 Report

### How to Run?

Open the terminal and navigate to the source directory. Then, compile the program by typing “**make**” and execute it using “**./parking\_lot\_simulation**”. Once executed, the program will produce output as specified by its functionality. To clean up generated files, use “**make clean**”, which will remove the executable file.

### Methods and Components Used in the Code

#### Mutexes

We use two mutexes, **carSpotLock** and **pickupSpotLock**, to protect the shared counters **availableCarSpots** and **availablePickupSpots**, respectively. This ensures that only one thread can access and modify these counters at a time, preventing race conditions and ensuring data consistency.

*// Mutexes for accessing shared variables*

pthread\_mutex\_t automobile\_lock;

pthread\_mutex\_t pickup\_lock;

#### Semaphores

Four semaphores are used:

* **pickupSemaphore** and **carSemaphore**: Signal when a pickup or car owner has arrived and is ready for the attendant to park the vehicle.
* **pickupHandlerSemaphore** and **carHandlerSemaphore**: Signal when the attendant has finished parking the pickup or car.

*// Semaphores*

sem\_t newPickup;

sem\_t inChargeforPickup;

sem\_t newAutomobile;

sem\_t inChargeforAutomobile;

#### Thread Creation and Management

We create two threads for each vehicle: one for the vehicle owner (**vehicleOwner**) and one for the attendant (**parkingAttendant**). These threads are synchronized using semaphores and mutexes to ensure that they operate correctly without interfering with each other.

*// Initialize mutexes*

pthread\_mutex\_init(&automobile\_lock, NULL);

pthread\_mutex\_init(&pickup\_lock, NULL);

/\*\*\*\* mutex and semaphore operations \*\*\*\*/

*// Destroy mutexes*

pthread\_mutex\_destroy(&automobile\_lock);

pthread\_mutex\_destroy(&pickup\_lock);

#### Shared Counters

Two integer variables, **availableCarSpots** and **availablePickupSpots**, are used to track the available parking spots for cars and pickups. These counters are protected by mutexes to ensure thread-safe access and modification.

*// Shared counters for free spots*

int mFree\_automobile = 8;

int mFree\_pickup = 4;

### Output

A screenshot of a computer

Description automatically generated