CSE 344 – SYSTEM PROGRAMMING – HW 3 REPORT

REŞİT AYDIN – 200104004019

1. Introduction

This homework simulates a parking lot management system where car owners arrive at the parking lot entrance with either automobiles or pickups, and parking attendants are responsible for parking these vehicles in designated spots. The parking lot has separate spots for pickups and automobiles, with a specified capacity for each. The program utilizes multithreading and synchronization mechanisms such as mutexes and semaphores to ensure thread safety and proper management of the parking process.

2. Problem-Solving Approach

The problem-solving approach involves several key components:

2.1 Threading and Synchronization: The program uses pthreads (POSIX threads) to implement concurrent execution. Mutexes (pthread_mutex_t) are employed to provide mutual exclusion and protect shared resources, specifically the counters for available parking spots for pickups and automobiles. Semaphores (sem_t) are used for thread synchronization, allowing threads to signal each other when they are ready to proceed.

2.2 Car Owner Functionality: The carOwner function represents the behavior of car owners arriving at the parking lot entrance. Depending on the type of vehicle (pickup or automobile), the function attempts to acquire a parking spot. If a spot is available, the corresponding counter is decreased, and the owner is notified of successful parking. If no spots are available, the owner leaves, and a message indicating the unavailability of temporary spots is printed.

2.3 Car Attendant Functionality: The carAttendant function simulates the behavior of parking attendants responsible for parking the vehicles. It waits for signals from car owners indicating the arrival of new vehicles and then updates the availability of parking spots accordingly.

2.4 Main Function: The main function initializes semaphores, creates threads for car attendants, and simulates the arrival of vehicles using a loop. Inside the loop, random vehicles (pickups or automobiles) are generated, and threads representing car owners are created to handle the parking process. The program runs indefinitely until manually terminated.

2.5 (Extra Test Function) - Simulation of No Temporary Spots: The simulateNoTemporarySpots function simulates the scenario where all temporary parking spots are occupied. It fills the parking lot with vehicles step by step until it reaches its full capacity, ensuring that the program terminates after all spots are taken.

3. Program Utilization

The program is designed to be executed as a standalone application. After initializing the semaphores and creating threads for car attendants, it simulates the scenario of filling the parking lot with vehicles using a loop. Inside the loop, random vehicles (pickups or automobiles) are generated, and threads representing car owners are created to handle the parking process. The program runs indefinitely until manually terminated.

To compile and run the program:

- make

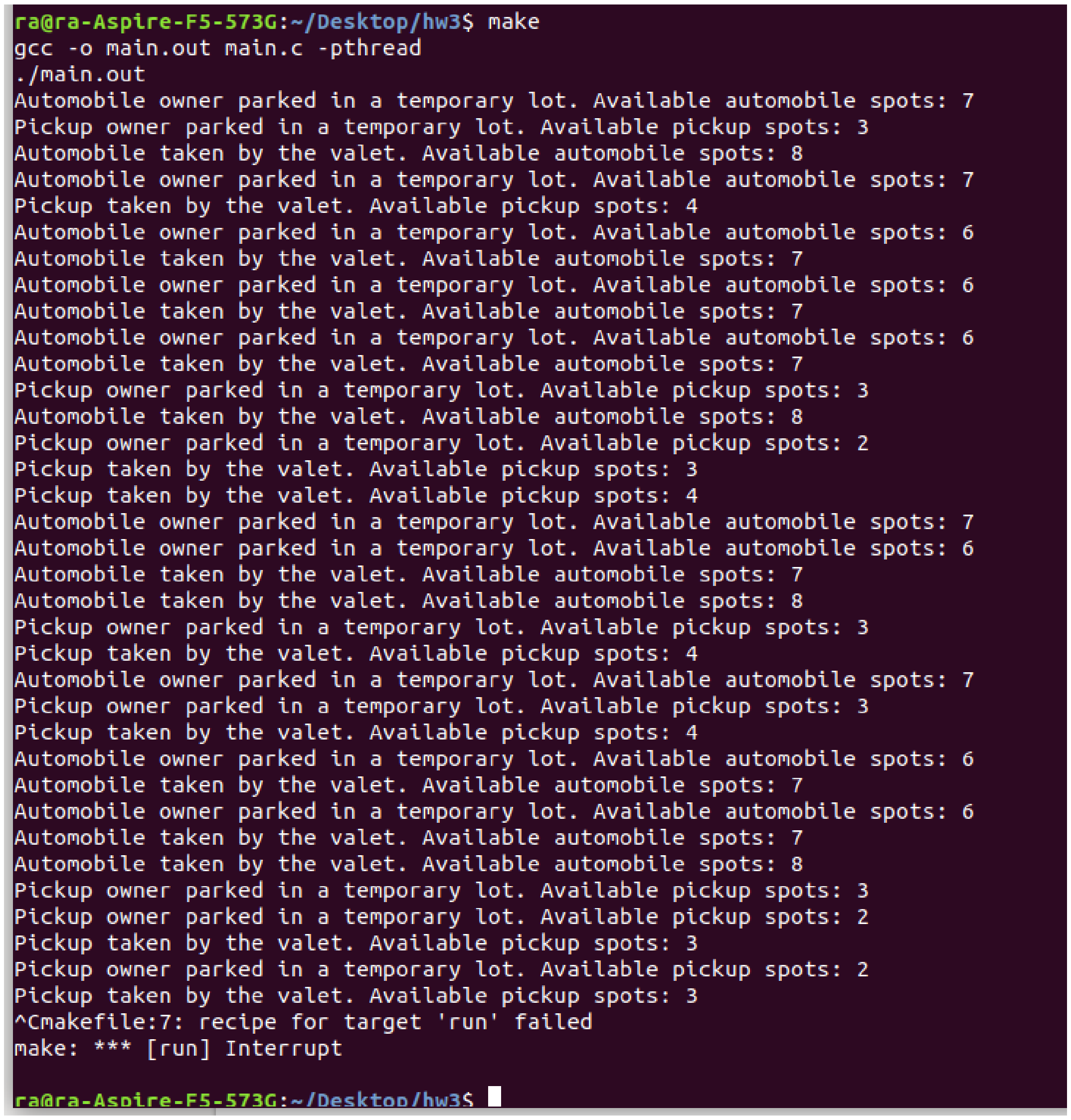
To clean up the program generated files:

- make clean

Commands can be executed.

4. Tests and Outputs

4.1 Test with random vehicle types



```
ra@ra-Aspire-F5-573G:~/Desktop/hw3$ make
gcc -o main.out main.c -pthread
./main.out
Automobile owner parked in a temporary lot. Available automobile spots: 7
Pickup owner parked in a temporary lot. Available pickup spots: 3
Automobile taken by the valet. Available automobile spots: 8
Automobile owner parked in a temporary lot. Available automobile spots: 7
Pickup taken by the valet. Available pickup spots: 4
Automobile owner parked in a temporary lot. Available automobile spots: 6
Automobile taken by the valet. Available automobile spots: 7
Automobile owner parked in a temporary lot. Available automobile spots: 6
Automobile taken by the valet. Available automobile spots: 7
Automobile owner parked in a temporary lot. Available automobile spots: 6
Automobile taken by the valet. Available automobile spots: 7
Pickup owner parked in a temporary lot. Available pickup spots: 3
Automobile taken by the valet. Available automobile spots: 8
Pickup owner parked in a temporary lot. Available pickup spots: 2
Pickup taken by the valet. Available pickup spots: 3
Pickup taken by the valet. Available pickup spots: 4
Automobile owner parked in a temporary lot. Available automobile spots: 7
Automobile owner parked in a temporary lot. Available automobile spots: 6
Automobile taken by the valet. Available automobile spots: 7
Automobile taken by the valet. Available automobile spots: 8
Pickup owner parked in a temporary lot. Available pickup spots: 3
Pickup taken by the valet. Available pickup spots: 4
Automobile owner parked in a temporary lot. Available automobile spots: 7
Pickup owner parked in a temporary lot. Available pickup spots: 3
Pickup taken by the valet. Available pickup spots: 4
Automobile owner parked in a temporary lot. Available automobile spots: 6
Automobile taken by the valet. Available automobile spots: 7
Automobile owner parked in a temporary lot. Available automobile spots: 6
Automobile taken by the valet. Available automobile spots: 7
Automobile taken by the valet. Available automobile spots: 8
Pickup owner parked in a temporary lot. Available pickup spots: 3
Pickup owner parked in a temporary lot. Available pickup spots: 2
Pickup taken by the valet. Available pickup spots: 3
Pickup owner parked in a temporary lot. Available pickup spots: 2
Pickup taken by the valet. Available pickup spots: 3
^Cmakefile:7: recipe for target 'run' failed
make: *** [run] Interrupt

ra@ra-Aspire-F5-573G:~/Desktop/hw3$
```

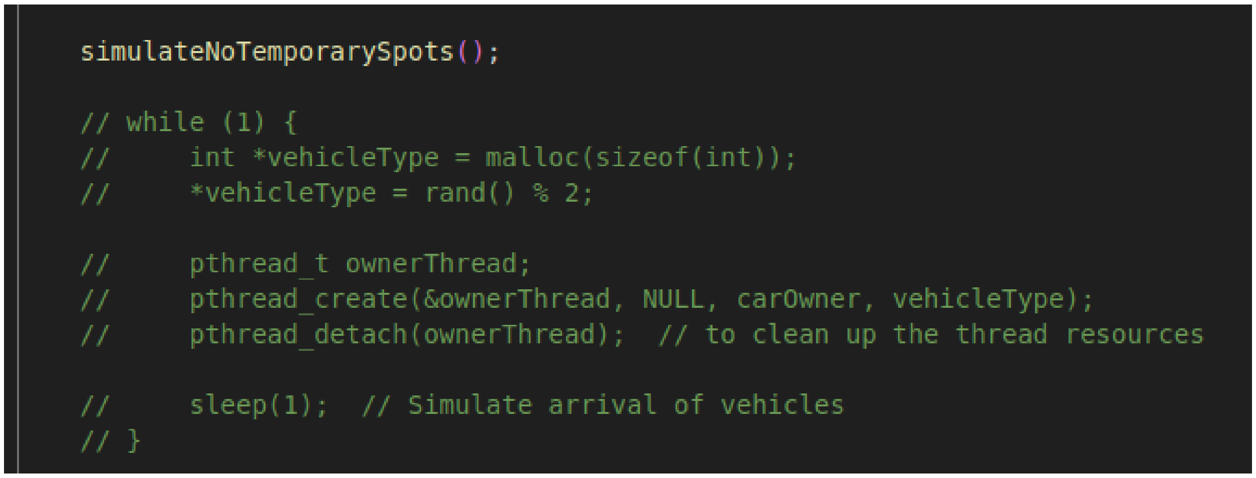
Explanation of output: The random generated vehicles are parked into the temporary parking lots and then valets take them randomly.

- Synchronization is ensured since only one vehicle can enter the parking lot at a time.
- The available spot number cannot exceed 8 for automobile vehicle type and 4 for pickup vehicle type.
- 1 second of delay has been used in between parkings to simulate reality.

4.2 Test with a custom noTemporarySlots function

I came up with a simulation idea to test my code. When the simulation starts, all the parking lots are available to the owners. As the simulation goes on, the parking slots are taken from the vehicle owners until there is no available spot left. With that, valets also take vehicles from the lots. After all of the vehicles are taken from the parking lots, the program halts.

- We can execute the test function in main by commenting out existing logic and calling the function.

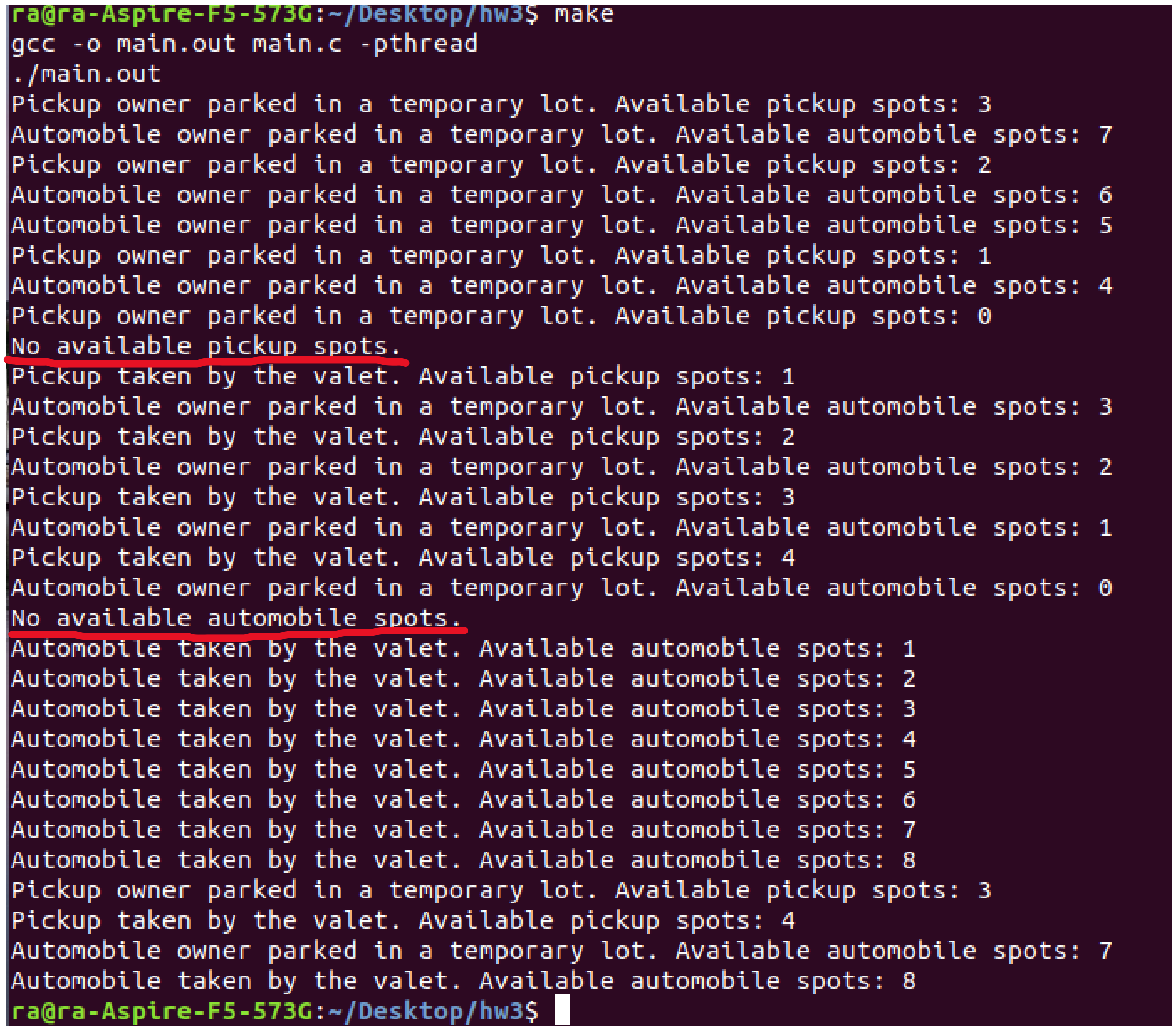


```
    simulateNoTemporarySpots();

    // while (1) {
    //     int *vehicleType = malloc(sizeof(int));
    //     *vehicleType = rand() % 2;

    //     pthread_t ownerThread;
    //     pthread_create(&ownerThread, NULL, carOwner, vehicleType);
    //     pthread_detach(ownerThread);  // to clean up the thread resources

    //     sleep(1);  // Simulate arrival of vehicles
    // }
```



```
ra@ra-Aspire-F5-573G:~/Desktop/hw3$ make
gcc -o main.out main.c -pthread
./main.out
Pickup owner parked in a temporary lot. Available pickup spots: 3
Automobile owner parked in a temporary lot. Available automobile spots: 7
Pickup owner parked in a temporary lot. Available pickup spots: 2
Automobile owner parked in a temporary lot. Available automobile spots: 6
Automobile owner parked in a temporary lot. Available automobile spots: 5
Pickup owner parked in a temporary lot. Available pickup spots: 1
Automobile owner parked in a temporary lot. Available automobile spots: 4
Pickup owner parked in a temporary lot. Available pickup spots: 0
No available pickup spots.
Pickup taken by the valet. Available pickup spots: 1
Automobile owner parked in a temporary lot. Available automobile spots: 3
Pickup taken by the valet. Available pickup spots: 2
Automobile owner parked in a temporary lot. Available automobile spots: 2
Pickup taken by the valet. Available pickup spots: 3
Automobile owner parked in a temporary lot. Available automobile spots: 1
Pickup taken by the valet. Available pickup spots: 4
Automobile owner parked in a temporary lot. Available automobile spots: 0
No available automobile spots.
Automobile taken by the valet. Available automobile spots: 1
Automobile taken by the valet. Available automobile spots: 2
Automobile taken by the valet. Available automobile spots: 3
Automobile taken by the valet. Available automobile spots: 4
Automobile taken by the valet. Available automobile spots: 5
Automobile taken by the valet. Available automobile spots: 6
Automobile taken by the valet. Available automobile spots: 7
Automobile taken by the valet. Available automobile spots: 8
Pickup owner parked in a temporary lot. Available pickup spots: 3
Pickup taken by the valet. Available pickup spots: 4
Automobile owner parked in a temporary lot. Available automobile spots: 7
Automobile taken by the valet. Available automobile spots: 8
ra@ra-Aspire-F5-573G:~/Desktop/hw3$
```

Explanation of output: The parking slots are taken from the vehicle owners until there is no available spot left. With that, valets also take vehicles from the lots. After all of the vehicles are taken from the parking lots, the program halts.

- Synchronization is ensured since only one vehicle can enter the parking lot at a time.
- The available spot number cannot exceed 8 for automobile vehicle type and 4 for pickup vehicle type.
- If there is no available slot to park a vehicle, an error message is displayed. (indicated red in photo)
- When the program reaches maximum number of counts for both vehicle types (8 for car and 4 for pickup), it terminates.
- 1 second of delay has also been used here to simulate parking.