Programming Assignment (PA) -3

(Riding to a Soccer Game)

CS307 – Operating Systems

FALL 2021/2022

Submission Report

by

Furkan Kerim Yıldırım

28138

Problem Description

In this programming assignment we were expected to implement simple, Unix command line based rideshare application for the fans of soccer clubs. For simplicity, it was assumed that there were two rival football clubs, and the supporters of that club were expected to be placed in the vehicles according to certain rules.

C Program Implementation

First of all, in rideshare.c file, the supporters of both teams were taken from the user in order for the program to work correctly. Then, the number of supporters taken from the user was checked so that they could be placed on the vehicles correctly, and the program was terminated if the condition given in the homework was not met. If the given condition is met, a mutex is defined to be able to access global data correctly and a mutex barrier is created so that insertion into vehicles can be done correctly. Next, a list of threads equal to the total number of fans was created, and a list of structs with randomly placed fans so that fans could find queues to buses more fairly and send data to threads. After the data structures were created, two semaphores were initialized so that the fans could get on the vehicles according to the given rule and the driver selection was correct. After the semahores were initialized, all the fans were sent to the threads and started to be placed in the vehicles.

In thread functions, a boolean variable is defined to control whether the driver is selectable, and the team information sent from the main function is determined. In order for the data to be processed to be processed correctly, the mutex started in the main function is locked and the first message is printed to the console. According to incoming fan information, sem_A and sem_B semaphores are kept with sem_T(Team) and sem_O(Other team) pointer semaphores. In order to get the values inside the semaphores, the sem_getvalue function is called and according to the values obtained from the semaphore pointers, it is determined whether the semaphore will continue or not. After determining whether the semaphore will continue or not, if the semaphore will continue, the information that the team has found a vehicle is printed on the console and the barrier created waits until it is sure that the other fans are also placed. After making sure that the vehicle is full, if a driver has not been selected yet, a driver is selected from the fans who get into the vehicle and the barrier used is destroyed and re-initialized. Finally, ensuring that all fans get on the vehicles and a driver is selected for each vehicle, the threads join the main function, and the program is finished.

A global mutex is used to access and process data correctly. A 4-way mutex barrier was used to select the driver from the fans who got on the vehicles, and after each selected driver, the 4-way mutex barrier was destroyed and re-initialized. In addition, semaphores were used in order for the fans to be placed on the buses in a suitable way, and the semaphore functions necessary for semaphore operations were re-defined.