Name: Muhammed Orhun Gale ID: 26754

CS307 PA3 REPORT

Implementation of Threads

In this assignment, in order to imitate a carpool scenario with multiple threads that are classified as ‘A Fans’ and ‘B Fans’, I used 3 type of threads.

One is the main thread that handles ‘Fan thread’ creation according to the number of each teams’ fans given as input and checks if the given inputs are correct or not.

Remaining two types are ‘Fan A thread’ and ‘Fan B thread’ which perform fundamentally the same operations to imitate randomly arriving fans that support two different teams. In the threads that I employed to imitate fans, I used ‘fanA_handler’ and ‘fanB_handler’ methods.

Implementation of Methods and Synchronization

In the fanA/B_handler methods, initially I employed a ‘mutex’ to put every arriving fan to the carpool queue without encountering a race condition.

After a fan declares that s/he arrives and waits for a car, if s/he is not the one that arrives last, who is the person that makes a car full, s/he simply release the mutex and goes to the sem_wait that puts it to the sleep.

If the arriving fan is the fan, who makes the car full, after s/he obtained the mutex, s/he does not release the mutex until the end of her/his execution. The aim to do this is blocking the formation of new cars until the current car is declared to be filled by all the fans that got a seat from the car and the latest arriving fan by saying that s/he is the captain. The latest arriving fan, after getting the mutex, checks that if there exist 4 fans from the same team or 2-2 from different teams. After deciding this, s/he claims that s/he is the captain and signals waiting fans. ,

To let them declaring that they find a spot on the car, I used pthread_barrier which makes the threads until a certain number (4 in this case) of them calls the pthread_barrier_wait function. After all of them declared that they find the car, lastly the captain declares s/he is the captain and releases the mutex to enable other fans to form new cars.