Operating Systems II

Car-Passenger-Syncronization

- Bhogalapalli Sahishnu

Following is the low level design information regarding the program that simulates passengers and cars at Jurassic park.

- Information regarding the passengers count, car count, respective average values for wander time and travel time distributions in milliseconds, and number of rounds are taken through the input file given as an argument while running the executable file.

- Two counting semaphores ‘car’ and ‘passenger’ are initialized externally to 0 along with a binary semaphore ‘current_active_passenger’ to 1.

- According to the information received, passenger threads and car threads are created and corresponding information is passed.

- Externally ‘struct cpass’ and ‘struct pass’ structures are created to pass variables to the corresponding thread functions.

- To a passenger thread, its index, average value for the wander time distribution, number of iterations, a dynamically generated array pointer ‘current_passenger_of_car’ and another dynamically generated array pointer, ‘passenger_total_time’ are passed.

- To a car thread, the same variables as above except in place of average value for wander time distribution, average value for travel time distribution are passed.

- Inside a car thread, every thread waits on externally created car semaphore, waiting for it to be posted by any of the passenger semaphores.

- Meanwhile in the passenger thread, it enters a for loop corresponding to the total number of iterations and sleeps for a randomly generated amount of time corresponding to the time wandering in the museum.

- After any passenger thread comes out of the sleep function, it encounters a sem_post function which triggers any one of the waiting cars to leave busy waiting and move forward.

- As soon as the passenger thread makes a sem_post on the car, it makes a sem_wait on the binary semaphore and updates a global variable named ‘current active_passenger’ to its index and hits the wait function on a passenger semaphore. (sem_wait is done because only one passenger thread can move forward as the binary semaphore was initialized to 1).

- Meanwhile the car thread busy waits in a while loop while the ‘current_active_passenger’ is being updated, where it exits the loop once the update is made.

- On exiting the busy wait, the car thread updates the ‘current_passenger_of_car’ array at its index to the ‘current_active_passenger’ and posts the binary semaphore to let the next passenger access the global variable.

- Now the car thread has access to the passenger id who is in the particular car. The car thread is made to sleep for a randomly generated amount of time corresponding to the travel time.

- Once the car thread is awake, it posts on the passenger semaphore to wake the corresponding passenger thread that is waiting on the passenger semaphore in its thread.

- This way, the particular passenger thread that stopped busy waiting completes its first round around the park in the corresponding car and it re-enters the for loop to complete the remaining k-1 rounds.

- For calculating the average time taken by a passenger to finish his k tours, we find the time before and at the end of the for loop using a high precision chrono clock. The time difference gives total time taken by a passenger to finish his k rides including the waiting time in the process.

- These values are stored in a dynamically declared array variable called ‘passenger_total_time’ which is passed to the passenger function in the beginning as mentioned earlier.

- Once every passenger finishes all his tours, the array is full and we average it out to find out the average time taken by a passenger to take all tours around the park.

- For calculating the average time taken by a car to ride its passengers, we use the randomly generated travel_time variable from earlier.

- Whenever we generate a random travel_time corresponding to the car’s travel time with its passenger, we add it to a global variable called ‘c_average’ using a binary semaphore called ‘c_average_binary_semaphore’ to avoid race conditions.

- Finally when all the passengers complete all their tours, that is when we find the passenger average, we also find the car average by dividing the c_average term by total number of cars to give us the average time taken by a car to ride all its passengers around the park.

- The program is set to output the average time taken to complete all the rides by a passenger and also the average time taken by a car to finish all the requested rides.

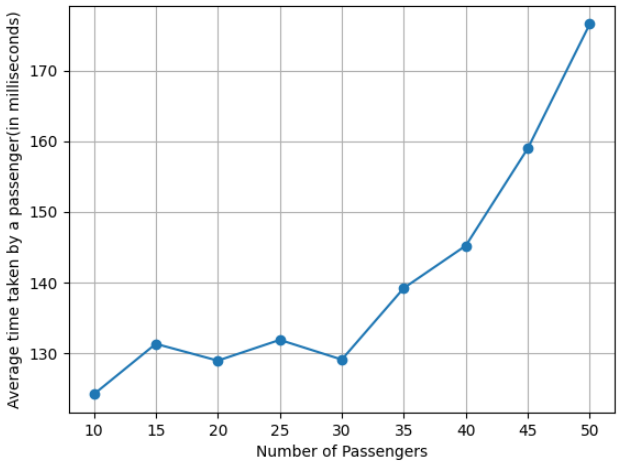
- To record time at any given instant, the timespec structure and time_t data types have been used.

- To calculate a time difference between two incidents, high resolution clocks from chrono library have been placed to record time and subtract them to get the result.

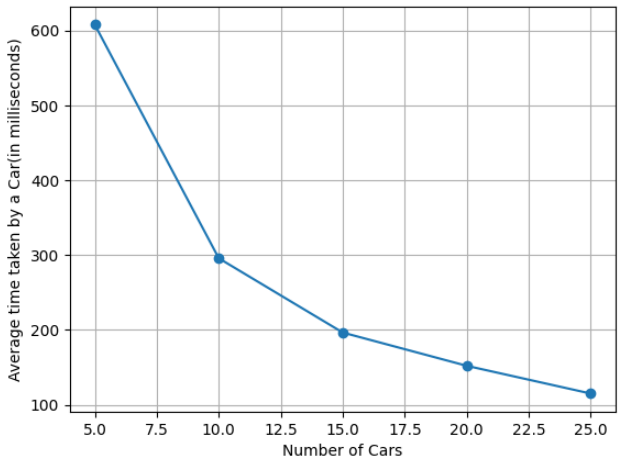
- To generate random numbers, ‘random engine constructor’ and ‘exponential distribution’ functions have been used. To make sure that different numbers are generated at every iteration, seed is varied using the ‘time_since_epoch()’ function from chrono library.

Graphs:

Following is the graph depicting the average passenger time fixing the number of cars and number of rounds to 25 and 5 respectively while changing the number of passengers.



Following is the graph depicting the average car time fixing the number of passengers and number of rounds at 50 and 3 respectively while changing the number of cars.



Observation and Analysis :

- We can see from the first graph that on increasing the number of passengers, there is no significant increase in the average time taken until the number of cars equals the number of passengers. Once the number of passengers overtakes the number of cars we see a gradual increase in average time taken for a passenger to finish all his rides. This is expected because if we keep the number of cars the same and increase the number of passengers above the number of cars, time taken by a passenger to finish all the rides is bound to increase.

- From the second graph, it is evident that there is a decrease in the amount of average time taken by a car to ride all its requested passengers with an increase in total number of cars. This also is expected because if the number of cars is increased keeping the number of passengers the same, the average time taken by a car to finish all the requested rides is bound to decrease.