Pset 2 Report

By Codrin Oneci

Question 3 Basic Threading

Average iteration time for the run() method is 10727 microseconds on my machine.

Average iteration time for the run_threaded() method is 11856 microseconds on my machine.

My Ubuntu 18.04 virtual machine has only two threads, and thus for three vehicles the run_threaded() method is not very efficient. One would expect the threading program to be efficient for machines with many threads (such as 4 or 8) and number of vehicles smaller that the maximum hardware threads number.

Thus in our case (three vehicles) on my machine the performances of the two implementations are close from the perspective of the execution time.

Question 4 Writing a Thread safe specification

I will use threading in such a way that the critical region of the system is void. Each vehicle will be run as a distinct thread and no common resources will be used to avoid synchronization issues.

1. *run_threaded* function specification
   1.1. The **simulator.c** file shall include a definition for a function named *run_threaded* of the form:
   *void run_threaded(struct t_simulator * sim)*
   1.2. The *run_threaded* function shall use threading to simulate the behavior of each vehicles from the array *sim->vehicles*.
   1.3. The execution time of the simulation shall be bounded by the condition that sim->current time is smaller than *sim->max_time*. The simulation updates shall be done inside a while loop.
   1.4. In the loop, for each simulation time, an *ids* array of type *pthread_t* of length *sim->n_vehicle* containing thread identifiers shall be created.
   1.5. In each iteration of the main loop, a thread should be created for each vehicle. The thread should call a function named *thr_function* of the type *void** that has as an argument a pointer to a structure describing the thread parameters, specifically the vehicle pointer and the time increment requested by the *update_state* function (which is defined in *vehicle.c*). The *thr_function* shall update the vehicle state and control law.
   1.6. The critical region of the code shall be void, in the sense that no vehicle iterative update thread is allowed to modify variables that are not directly related to the thread function arguments.
   1.7. After the execution of every iteration through vehicles in the main while loop, the *run_threaded* function shall print to the terminal the average execution time of an iteration for a vehicle thread update.

   Since the threads don't share resources, a special termination condition regarding threads is not needed: each thread will simply terminate after updating its specific vehicle state and control law, when the current iteration of the main while loop function finishes. This is usual C pthread behavior. Also I don't use any condition variables in my code because the implementation of threads is as simple as possible and they operate with distinct object and variables instances.

My implementation ticks properly because the threads are created inside the main loop, and I use pthread_join and give the second argument as a NULL to make sure that they all thick concomitantly in the simulation. Thus all my threads run in parallel, assuring proper tick such that vehicles positions can be shown on the display in real time moving in the bounded domain.

Question 5 The Github log

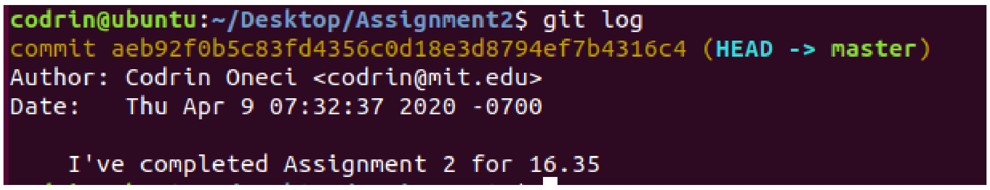
codrin@ubuntu:~/Desktop/Assignment2$ git log

commit aeb92f0b5c83fd4356c0d18e3d8794ef7b4316c4 (HEAD -> master)

Author: Codrin Oneci <codrin@mit.edu>

Date: Thu Apr 9 07:32:37 2020 -0700

I've completed Assignment 2 for 16.35



```
codrin@ubuntu:~/Desktop/Assignment2$ git log
commit aeb92f0b5c83fd4356c0d18e3d8794ef7b4316c4 (HEAD -> master)
Author: Codrin Oneci <codrin@mit.edu>
Date:   Thu Apr 9 07:32:37 2020 -0700

    I've completed Assignment 2 for 16.35
```

Question 6 Time spent

1- 25 hours
2- 8 hours
3- 20 hours
4- 30 min
5- 5 min
6- 3 minutes