Assignment 2

Georgiy Krylov

January 31, 2022

ASSIGNMENT IS TO BE COMPLETED INDIVIDUALLY BY ALL STUDENTS!

1 Description

This assignment to introduce the class to the process of POSIX threads creation, parameters passing, barrier synchronization. **The assignment is Due by 11:59 a.m. (noon) on Wednesday, 9th of February 2022.**

2 Task

In this assignment you're working on moving company simulation (multi-buffer producer-consumer problem). For this assignment you should use pipes as your buffers. At the beginning of the program, your code should ask for the number of people living in the house, the number of movers and the number of truck drivers. You will create three types of threads:

- A thread that simulates a house dweller. A house dweller thread introduces itself once created, produces an item (it takes random number of seconds, should be facilitated by **sleep** function and pre-defined macros for random numbers , see later see later in Subsection 2.1) of certain weight (Random weight, see later in Subsection 2.1), puts it on the floor (i.e. writes it to the first pipe), for the mover (i.e., mover thread) to pick up and bring downstairs. The house dweller needs to report its id, package weight and how much it took to report.The thread can only produce so many items (defined in sample code). After thread produced the maximum number of elements the thread should report it is done moving and terminate.
- A thread that simulates a mover's behavior. A mover picks up a package from the house floor and brings it next to the truck (i.e. sends a message through the second pipe). It takes the mover random amount of time. Once the box is delivered, the mover should report its id and the weight of the box, as well as how long did it take the mover to finish the task.

Once the mover is done bringing all the boxes from the house to the land next to the truck, the mover should report that it is done and terminate.

- A thread that simulates a truck driver. A truck driver loads up a package from the land next to the truck over a random interval of time. The trucks can carry as much weight as needed, but can fit only defined number of crates (i.e the volume is limited, the mass is not). After loading a crate, the thread should report truck thread id, the weight of the box just loaded, the time it took to load the box, as well as how much room is left in the truck. Once the truck is full, the truck should take a round trip to the new house, reporting that the truck is fully loaded, the total weight and the time it takes for a round-trip (This is just a print statement followed by the number of trips, no further communication is required). If there are no elements left on the floor next to the truck, the truck should do a final trip, reporting that the truck is not full, the total weight and the time it takes for a one way trip (same formula as for the round trip, just different print statement). Once the truck is done, it should report its id and that it's done and terminate.

The execution of the threads should be parallel. Once all threads of a certain kind are done, the main thread should close the pipe end that is no longer in use. Once all threads are done, the main thread should report the moving is finished.

2.1 Code Snippets

You should use the code from **sample_code.txt** as the template for your program. It contains most of the defines. For your convenience, you're provided with a macros that generates a random number of. Example use:

```
1       interval = RANDOM_WITHIN_RANGE(
            MIN_TIME_FOR_HOUSE_DWELLER,MAX_TIME_FOR_HOUSE_DWELLER
            ,seed);
2       sleep(interval);
```

The **seed** variable should be a random number generated for each house dweller, passed as a member of a thread parameters structure. To generate the variable in the main function, you can use **rand()** function, don't forget to call **srand(time(NULL))** beforehand (the template does it for you).

3 Input/Output format

Your assignment has to request the number of house dwellers, movers and truck drivers from the console. Please refer to **sample_execution.txt** and **sample_execution_2.txt**.

4 Submission instructions

Please submit your C file to D2L Assignment box. Make sure your code compiles and runs. Make sure your code follows the specified input/output format. You must use C programming language to solve this assignment. Important to note in this course: if your program produces correct output, it doesn't necessarily mean you have properly managed the resources and penalties are possible. This assignment focuses on threads management and communication, and therefore the TA will be asked to make sure the threads are properly created and terminated.

NOTE: THE INPUT AND OUTPUT OF YOUR PROGRAM IS SPECIFIED SUCH THAT THE MARKER CAN AUTO TEST YOUR SUBMISSIONS. PLEASE FOLLOW THE SPECIFICATIONS! INPUT WILL BE READ VIA stdin (E.G. KEYBOARD). OUTPUT SHOULD BE PRINTED TO stdout (E.G. MONITOR).