Assignment 3

Georgiy Krylov

October 2, 2024

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 p.m. (1 minute before Saturday) on Friday, the 11th of October 2024.**

2 Task

In this assignment you're working on moving company simulation (multi-buffer producer-consumer problem). The threads should belong to the same process, e.g. we are not using `fork()` in this assignment. 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. This action takes a random number of seconds and should be facilitated by using the **sleep** function and the pre-defined macro for generating random numbers (see later in Subsection 2.1). The items have a certain weight (Random weight, see later in Subsection 2.1). The house dweller puts the box 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 (an integer that you are to assign when creating threads), the package weight and how much it took to prepare the package. The thread can only produce so many items (defined in the sample code). After the thread has 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 and set them down next to the truck, the mover should report that the job is finished and terminate.
- A thread that simulates the truck driver. A truck driver loads up a package from the ground over a random interval of time. A truck can carry as much weight as needed, but can fit only a pre-defined number of crates (i.e., the volume is limited, the mass is not). After loading a crate, the thread should report the 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. It should be done by first reporting that the truck is fully loaded. This message should also contain the total weight and the time it takes to take a round-trip. The round trip is a random number, corresponding to the number that the thread must be put to sleep(). 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, as well as the total weight and the time it takes for a one way trip. This report should use the same statement as for the round trip, differing only in the message. Once the truck is done, it should report its ID, 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 you will need to use. For your convenience, you're provided with a macro that generates a random interval. Example use:

```
interval = RANDOM_WITHIN_RANGE(
    MIN_TIME_FOR_HOUSE_DWELLER,MAX_TIME_FOR_HOUSE_DWELLER
    ,seed);
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** to see the output.

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! THE INPUT WILL BE READ VIA stdin (E. G., KEYBOARD). THE OUTPUT SHOULD BE PRINTED TO stdout (E. G., MONITOR).