OSES LAB #3 – Scheduling, Priority Inversion, Priority Ceiling, Deadlock

Please read the assignment description carefully

Purpose of the lab

These exercises assess your knowledge of scheduling, priority ceiling, and deadlock.
All of them shall run on SimulIDE / Arduino Uno.

Deliverables and deadlines

You shall provide a report (pdf file only) where you describe the oil file and the code you implemented for each exercise we proposed. The output produced by your solutions shall also be included (insert the screenshot in the report).
The report shall be uploaded to the portale della didattica using the Elaborati section.
The report shall be provided by Nov 22nd , 2024, 18:00.

Exercise #1

Define three periodic tasks with implicit deadline (deadline equal to period) and the following parameters:

Task	*Period*	*Execution time*
A	1000ms	200ms
B	1500ms	700ms
C	2800ms	300ms

All tasks must be ready for execution at t=0. Use the function `do_things(ms)` to simulate execution time. It traps the current task in a busy loop for `ms` milliseconds.

```
void do_things(int ms)
{
    unsigned long mul = ms * 504UL;
    unsigned long i;

    for(i=0; i<mul; i++) millis();
}
```

Side question: Using `delay(ms)` for this would lead to an incorrect behavior. Why?

Measure the *worst-case response time* of task C under an optimal priority assignment and print it out. Run the code for an adequate amount of time to ensure the system encounters a critical instant (the critical instant for a task is the task activation time that leads to the task maximum response time). Compare your measurement with theoretical expectations.

- Hint: The function `unsigned long millis()` can be used to measure response time. It returns the number of milliseconds elapsed since boot.

Exercise #2

Update the code of exercise #1 so that A and C share a resource in the following way:

- On each activation, A requests the resource, keeps it for 200ms, and releases it.
- On each activation, C requests the resource after executing for 100ms, keeps it for 200ms, and releases it.

Keep the period, execution time, and priority of A and C the same as before. Instrument the code so that its output contains enough information to demonstrate how and when *priority inversion and immediate priority ceiling* come into effect.

Exercise #3

Mr. C. Lever believes he can use a message to implement critical regions in tasks A and C, instead of using a resource like in exercise #2, in this way:

- At t=0 an initialization task runs once and sends one message to a message object.
- When task A or C want to enter their critical section, they repeatedly check the message object until they successfully receive the message.
- They send back the message at the end of their critical section.
- Message type and content are irrelevant.

Implement Mr. Lever's idea and discuss whether it works or not. Use the same task parameters as in exercise #2.