Mattia Piras: 4524072

README

This is the first assignment of Real Time Operating System whose aim was the implementation of the Priority Ceiling Protocol in C++.

Note: run the program few times to see that it is schedulable

RUNNING THE PROGRAM

In order to run the program, the following commands are needed:

g++ -pthread assignment1.cpp -o <name of the executable>

as superuser: **sudo su**

./name of the executable

DESCRIPTION

It was fun coding it even the presence of few drawbacks. One of them, was how to calculate WCET for each task. This has been overcome by computing the blocking time of each critical section in each task and by considering which critical section could block which task. I had a to consider, as the algorithm says, the longest one for each task Bi (Obviously the task with the highest period, so the lowest priority has Bi = 0 because none of the other task, with a higher priority, shouldn't block it).

I remembered to consider both direct and indirect blocking.

As usual (not for all, only for this course) tasks are scheduled with Rate Monotonic.

In the main function after initialization for Rate Monotonic scheduling I set up the environment for using priority ceiling protocol being acquainted to give the right priority to each mutex; choice determined by considering which mutex protect which task's critical section.

Then I had to cope with the various critical sections and the decision about when to start and to finish them. I chose to start critical sections before some for loops in order to increase their durations and then lock and unlock mutexes as critical sections has started and have finished. The three global variables are modified as request.

Then in order to have a better estimate of this WCET I recomputed it every 10 executions of each task function.

FUTURE DEVELOPMENT

- Mechanisms of counting missed deadlines can be implemented