CSE 522 – REAL TIME EMBEDDED SYSTEMS

ASSIGNMENT 1

Submitted by,

Carmel Mary Jose (1213149364)

Sharath Renjit Naik (1213340750)

Assignment Objectives

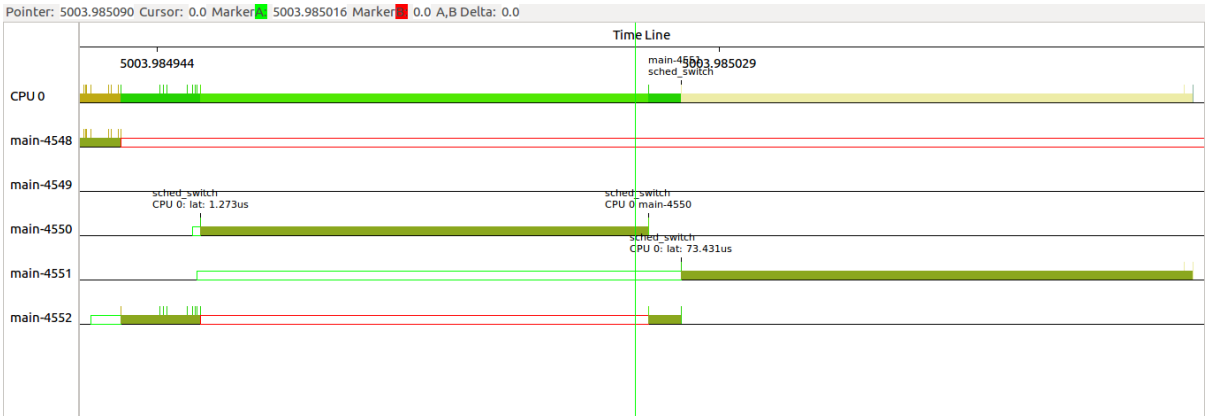
1. To program real-time tasks on Linux environment, including periodic and aperiodic tasks, event handling, priority inheritance, etc.
2. To use Linux trace tools to view and analyze real-time scheduling.

To be included in the report

Verification of scheduling of the tasks a specific task set by showing that

1. Tasks are scheduled under the SCHED_FIFO with the assigned priorities, and
2. Priority inversion under normal and PI-enabled mutexes.

EXECUTION OF TASKS UNDER SCHED_FIFO



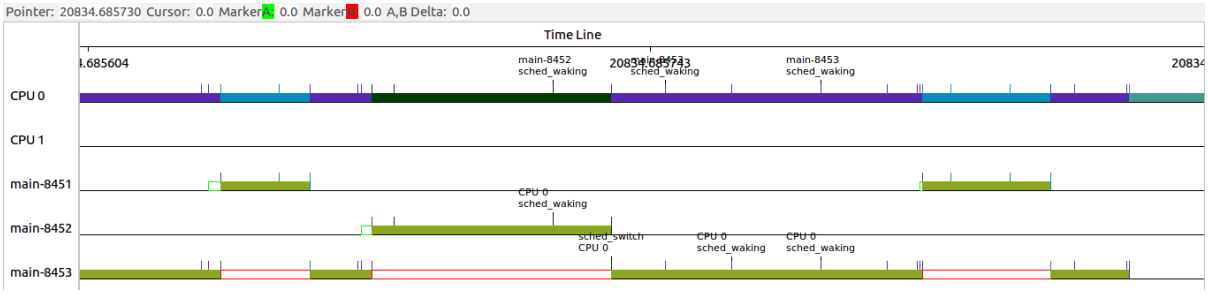
In the figure above the following are the priorities of the threads shown

PID	NATURE	SCHED_FIFO PRIORITY
4548	main()	N/A
4549	Aperiodic| Mouse	99
4550	Periodic | Period = 500	20
4551	Periodic | Period = 200	10
4552	Aperiodic | Event 1	10

After the mouse click trigger we can see that 4552 has woken up and started executing. During its execution, 4550 wakes up for its period, since it has higher priority than 4552, 4552 gets blocked and 4550 proceeds to execution. Now, 4551 also enters the ready queue. Since it has a lower priority than 4550, it must wait. Upon completion of 4550, 4551 proceeds to execute due to the FIFO nature of scheduling and lastly 4551 executes.

PRIORITY INVERSION

1. Normal mutexes



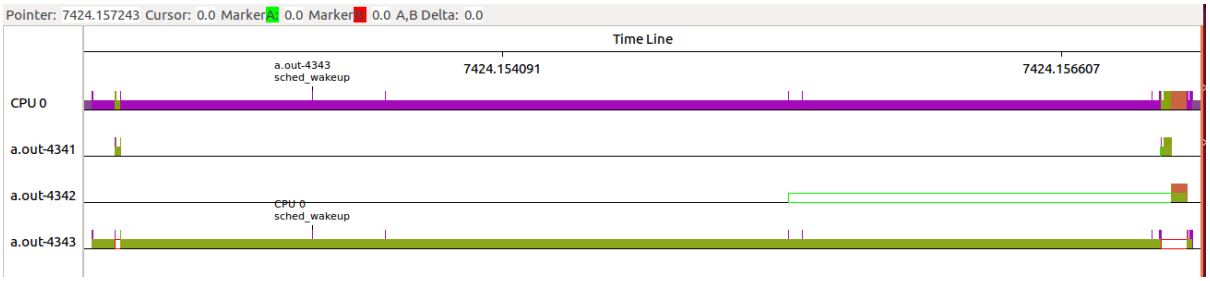
For this section, we use threads with the following characteristics:

PID	NATURE	SCHED_FIFO PRIORITY
8451	Periodic | Period = 10	90
8452	Periodic | Period = 11	70
8453	Periodic | Period = 9	50

As per the figure we see that 8453 is currently executing and gets pre-empted by 8451, which has higher priority. However, 8453 has locked resources for its execution which is required by 8451 so it gets blocked until those resources are freed. Meanwhile, 8452 wakes up for its execution and since it doesn't require any of the resources locked by 8453, it proceeds to execute its commands and sleeps when concluded at which time, 8453 continues to execute. When 8453 has released resources required by 8451, it get's pre-empted and 8451 proceeds to execute and lastly, when 8451 finished execution, 8453 executes its remaining steps.

The essence of this is that 8451 which has the highest priority is blocked by 8453 which has the lowest priority and during execution, 8452 which has priority between the others can execute when the highest priority task, 8451 remains to be blocked. This is a case of priority inversion, where the highest priority task is blocked due to resource locking while lower priority tasks proceed with the execution.

2. PI-enabled mutexes



For this section, we use threads with the following characteristics:

PID	NATURE	SCHED_FIFO PRIORITY
4341	Periodic | Period = 10	90
4342	Periodic | Period = 11	70

4343	Periodic | Period = 9	50

In the program with PI-enabled mutexes, 4343 is executing when it is pre-empted by 4341, of higher priority, however, due to locking of resources by 4343, 4341 is blocked and 4343 continues with its execution. In due course, 4342 is ready for execution. However, it's unable to execute because 4343 has inherited the priority of 4341 when the latter was blocked by the former. So now, when 4343 releases resources, 4341 executes followed by 4342 and finally 4343 executes its final steps.

Here, in contrast to the previous case, even though the highest priority task was blocked by the lower priority task, the intermediate priority task is unable to execute due to the PI-enabled mutexes which causes priority inheritance. In this case, this inheritance causes the lowest priority task to obtain the highest priority whereby it can stop the execution of the intermediate priority task.