**ECEN 5623**

**Real Time Embedded System**

**Homework set 2**

**Ayush Dhoot**

**2.5 Implement a Linux process that is executed at the default priority for a user-level application and waits on a binary semaphore to be given by another application. Run this process and verify its state using the ps command to list its process descriptor. Now, run a separate process to give the semaphore causing the first process to continue execution and exit. Verify completion.**

**A screenshot of a computer screen

Description automatically generated**

Figure 1: Running process ‘wait’ which waits for ‘unlock’ to complete execution

**A screenshot of a computer screen

Description automatically generated**

Figure 2: Running process ‘unlock’ which will unlock ‘wait’ and both will execute

**A screenshot of a computer screen

Description automatically generated**

Figure 3: Both processes have executed (verified using ps -a command)

**3.5 If EDF can be shown to meet deadlines and potentially has 100% CPU resource utilization, then why is it not typically the hard real-time policy of choice? That is, what are drawbacks to using EDF compared to RM/DM? In an overload situation, how will EDF fail?**

EDF scheduling can have 100% CPU utilization as well as meet deadlines but they are least preferred choice for scheduling of hard real-time systems. There are many shortcomings of EDF which makes it unsuitable for use as policy of hard real-time systems. One of the basic and most prominent one is that EDF being a dynamic priority preemptive scheduling policy needs to recalculate the priorities after each deadline passes. This takes up more computational time for calculating the new priorities based on which task has its deadline earlier. In real-time systems, even a slight miscalculation or delay in the completion of a task may lead to catastrophic events, EDF is not considered for scheduling of the system. Rate Monotonic on the other hand, does not have 100% utility (unless the tasks are harmonic) but they have static priority calculated before the scheduling starts and do not require extra computation time for calculating the priorities after each deadline. Another issue with the EDF scheduling is that there are high overheads occurring in order to have resource sharing in a real-time system. The deadlines for the tasks are meant to be crossed due to this problem of resource sharing.

In case of overload in a system, use of EDF policy for scheduling is not able to properly schedule the system. When few of the tasks miss their deadlines due to some reason, it causes a condition called as overload. As EDF is a dynamic priority-based policy, the priorities are computed after each deadline has passed. So, in a case where there is a task missing its deadline, the calculation of priorities can get complicated and, in many cases, lead to fatal problems. Even if there is precise prediction done for the system, there will be a possibility of task missing its deadline and that will cause the rest of the tasks to miss their deadlines. This cascading effect occurring in EDF scheduling due to an overload in the system is another reason for not preferring EDF for real-time systems. In overload situations, there is very less possibility of identifying which task causes the problem but even if it is identified, the prediction of the next tasks or the time where the task will overshoot its deadline is very difficult.

In EDF policy, even a least priority task exceeding the deadline assigned to it might cause the most critical task to miss its deadline in an overload situation causing the system to fail.

**4.2 If a system must complete frame processing so that 100,000 frames are completed per second and the instruction count per frame processed is 2,120 instructions on a 1 GHz processor core, what is the CPI required for this system? What is the overlap between instructions and IO time if the** **intermediate IO time is 4.5 microseconds?**

Data:

Frames per second(FPS) = 100,000

Instruction count per frame = 2120

Clock period = 1GHz

intermediate IO time (IOT)= 4.5 μs

Required:

CPI

Overlap between instruction and IO time

Solution:

Part 1:

We assume that the execution time is 1 second.

Part 2:

**References:**

<https://nptel.ac.in/courses/Webcourse-contents/IIT%20Kharagpur/Embedded%20systems/Pdf/Lesson-30.pdf>