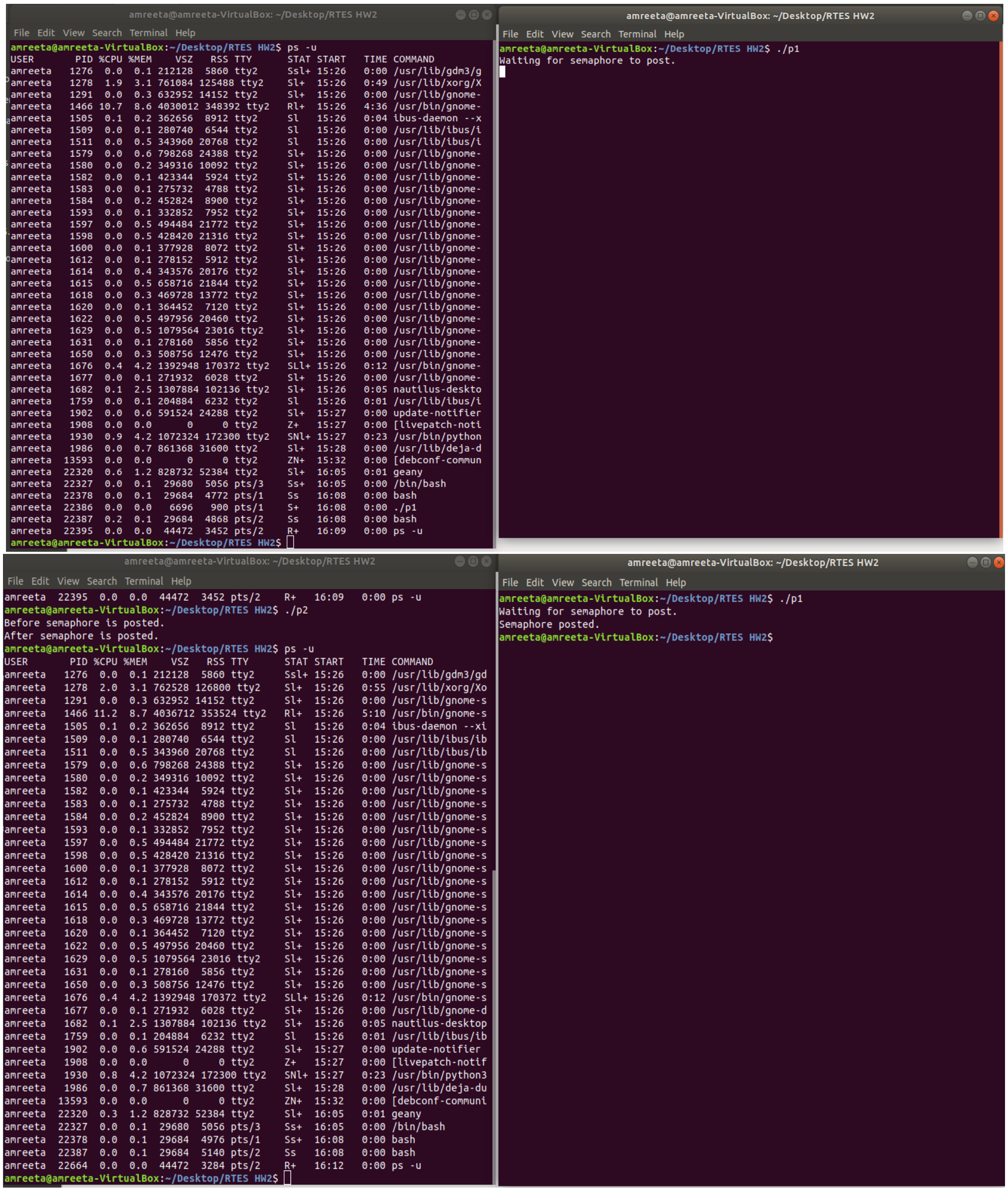
Homework set 2
02/28/2019

Question 1 code is run on virtual machine.

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.



On execution, process 1 opens a semaphore named "/Amreeta" and waits for it. ps command is used to display the running state of the process. Process 2 posts the semaphore named "/Amreeta". As the semaphore is posted, the Process 1 can come out of sem_wait and ps command shows that process 1 is not running anymore.

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?

RM policy is based on assignment of priority to tasks based on the frequency of occurrence of task. The task with the highest frequency gets the most priority, according to this policy. Hence, it is a fixed priority policy. EDF, on the other hand, is a dynamic priority policy which involves the dynamic mapping between the absolute deadlines and priorities as the tasks are given priority on the basis of the earliest approaching deadline. Even though EDF gives increased processor utilization, there are several drawbacks of EDF which include:

- Implementation Complexity: The implementation of EDF is complex and inefficient due to dynamic priority management. RTOS and commercial kernels available in the market are designed to implement fixed priority scheduling, so in order to implement EDF, there is a need for updating the kernel's fixed priority based on dynamic deadline calculation, which increases the computation involved.

- Runtime Overhead: The runtime overhead for EDF is considered to be more than that of RM as the implementation complexity for EDF is of order n - O(n), where n is the number of tasks to be scheduled, due to dynamic priority assignment. Since RM is a fixed priority policy, it does not involve runtime priority calculation and hence it has complexity of order O(1).

- Schedulability Analysis:

  RM Policy
  RM LUB Sufficient Condition

$$U = \sum_{i=1}^{m} (Ci/Ti) \le m(2^{\frac{1}{m}} - 1), U \lim m \to \infty = \ln(2) \approx 0.69$$

  EDF Policy
  Sufficient Feasibility Test

$$\forall tasks \in 1...m, T_{hyperperiod} = 0...(T_1 T_2 ... T_{m-1} T_m)$$
$$(T_1 T_2 ... T_{m-1} T_m) = \frac{(T_1 T_2 ... T_{m-1} T_m)}{T_1} C_1 + ... + \frac{(T_1 T_2 ... T_{m-1} T_m)}{T_m} C_m$$
$$\therefore \sum_{i=1}^{m} (Ci/Ti) \le 1$$

- Response Jitter and Latency: Jitter and Latency can make EDF schedule unpredictable. Despite these factors, if tasks are schedulable using both the policies, due to fixed priority assignment in RM policy, the task schedule is predictable, while in the case of EDF, jitter and latency may vary the dynamic deadlines making the task schedule unpredictable even though the overall task set is schedulable over the LCM period.

Overload Situation

In case of RM Policy, if the task overrun occurs at the i^{th} task, then all the tasks with priority lower than i may miss the deadline. In case of EDF, if the task overrun occurs at the i^{th} task, then all the tasks may miss the deadline because the dynamic deadline of the i^{th} task will become negative and since priority is assigned to the task with the lowest deadline, the other tasks will not be able to preempt. So, in case of EDF, if overrun occurs for any task, it creates a domino effect and all the tasks in turn may miss the deadlines.

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?

$$\text{CPI} = \frac{CPU\ Frequency}{Frames\ \ per\ second\ X\ Instruction\ count\ per\ frame} = \frac{1000000000}{100000\ X\ 2120} = 4.716$$

$$\text{Overlap Time} = \frac{Intermediate\ IO\ Time\ per\ frame}{Time\ to\ execute\ one\ frame} = \frac{4.5\ X\ 10^{-6}}{10^{-5}} = 0.45\ \text{seconds}$$

Overlap Percentage = 45%

REFERENCES

- “Rate Monotonic vs. EDF: Judgement Day” - Giorgio C. Buttazzo.
- “Real Time Embedded Components and Systems with Linux and RTOS”- Sam Siewert, John Pratt.