# TTK4145
# Exercise 9

Anders Fagerli

## 1 Task 1

1. Why do we assign priorities to tasks?

   We assign priorities to tasks as some tasks may be more important than others, so the scheduler executes the prioritized task more often. E.g if two tasks are monitoring the state of a battery, it may be more critical to monitor the temperature than the voltage, and we can then assign more resources to the important task.

2. What features must a scheduler have for it to be usable for real-time systems?

   The scheduler must be predictable and finish tasks within a valid deadline. Tasks may be time-critical, so the scheduler must make sure all tasks are completed within a reasonable time, even if they have varying priorities.

## 2 Task 2

Draw a Gantt chart

1. without priority inheritance

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | | | | | E | | | | | | | Q | V | E | |
| b | | | E | V | | V | E | E | E | | | | | | |
| c | E | Q | | | | | | | | Q | Q | | | | E |

2. with priority inheritance

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | | | | | E | | | Q | | V | E | | | | |
| b | | | E | V | | | | | V | | | E | E | E | |
| c | E | Q | | | | Q | Q | | | | | | | | E |

# 3 Task 3

1. What is priority inversion? What is unbounded priority inversion?

   A higher priority task is waiting for a lower priority task, e.g to release a shared resource, indirectly inverting the priorities of the two tasks. If unbounded, the higher priority task may end up waiting indefinitely.

2. Does priority inheritance avoid deadlocks?

   No, two separate tasks may each lock two different shared resources and end up in a deadlock if the other shared resource is needed by one of the tasks.

# 4 Task 4

1. There are a number of assumptions/conditions that must be true for the utilization and response time tests to be usable (The "simple task model"). What are these assumptions? Comment on how realistic they are.

   - The application consists of a fixed set of tasks (Realistic, we define the tasks ourselves)
   - All tasks are periodic, with known periods (Period of all tasks may be unreasonable to know)
   - All tasks are independent of each other (Unrealistic in larger, complex systems)
   - All overheads, context-switching times etc. are ignored (Unsure)
   - All tasks have a deadline equal to their period (Reasonable to assume that a task's execution time is within its period)
   - All tasks have a fixed worst-case execution time (Execution time can be set arbitrarily large, so should be reasonable)

2. Perform the utilization test for the task set. Is the task set schedulable?

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} = 0.88$$

$$n(2^{\frac{1}{n}} - 1) = 0.78$$

   $U \not\leq n(2^{\frac{1}{n}} - 1)$, so we do not know if the task is schedulable as the utilization test is only sufficient, not necessary.

3. Perform response-time analysis for the task set. Is the task set schedulable? If you got different results than in 2), explain why.

$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j \tag{1}$$

   Using Equation 1, we get $R_c = 5$, $R_b = 15$ and $R_a = 50$, all within their respective periods. The task is therefore schedulable, as the response-time analysis is both sufficient and necessary.