Lab 5 - DVA454 HT2024

Johan Sandred - jsd21003
Erik Folkesson - efn24005

1. What is the main difference between preemptive and cooperative scheduling?

Preemptive scheduling means that the scheduler interrupts tasks in the middle of their execution to allow other tasks of higher priority to execute instead. Cooperative scheduling does not perform this kind of interruption, but instead waits for the task to willingly relinquish its execution right by yielding or entering a blocked state, for example by sleeping or waiting for a semaphore.

2. What is the difference between vTaskDelay() and vTaskDelayUntil()?

vTaskDelay() will delay the execution of the task by the provided duration while vTaskDelayUntil() will instead delay it until the provided time point has been reached.

3. Design of Assignment 2 (priority inversion).

Our design has three tasks with low, mid, and high priorities. Each task has a release time: the time from when the task is created to when it starts its execution. The order of the release times for the different tasks goes from low priority task < high priority task < mid priority task.

The low priority task starts by taking a mutex and then enters a loop meant to simulate work. While the low priority task is performing its work the high priority task reaches its release time. The high priority task then starts its execution, preempting the low priority task. The high priority task then tries to take the same mutex that the low priority task is holding, becoming blocked in the process. Since the high priority task is blocked, this results in a context switch back to the low priority task so it can continue with its execution.

However, before the low priority task finishes its work, the mid priority task reaches its release time and will preempt the low priority task. This means that the mid priority task gets to run even though there is a high priority task that wants to run, but the high priority task cannot run since the low priority task is holding onto the mutex that the high priority task is waiting on. This creates a situation where the mid priority task effectively can be seen as having a higher priority than the high priority task, creating a priority inversion.

Once the mid priority task finishes its work, the low priority task can be scheduled and finish performing its work and finally relinquish its mutex. As soon as the low priority task is no longer holding onto the mutex, the high priority task will preempt the low priority task. The high priority task can then finally take the mutex and finish its work. At that point, as long as the none of the other tasks have started their next period, the low priority task is allowed to continue executing, eventually finishing its own period.