Scheduling Algorithms

- Static time driven:
    - We define a time frame
    - At the end of a frame, we decide what to do next
        - Non-preemptive: do next task if idle
        - preemptive : do next task in the queue
    - Tasks are pre-ordered in queue
        - Periodic tasks

      E.g. t1, t2, t1, t3, t2, t1, t3, t3
        - When the task completes, it goes to the end of the queue
            - Circular look-up table

    - Can also assign tasks to frames and repeat the frames
        - A.k.a time division multiplexing

      E.g. IMG91
        - Pre-computed & a table for ptr

    - How do we determine task order?
        - We have a few vars to guide us:
            - Execution time
            - Deadlines
            - priority
            - Period
            - Task dependencies
            - Frame size
                - Need to have enough time for tasks to finish
                - Need slack time
                    - We want to limit this as well
        - We Use dynamic scheduling algorithm to aid our design

- Dynamic Scheduling
    1. Round-robin
        - Standard time-slicing
        - Real-time? NO.
        - Features: sharing, responsiveness, throughput
        - How about we have really short-slices?
            - Every task gets a chance to run.
            - No deadlines & no determinism ≡(equivalent) no real-time
        - GPT
            - round-robin scheduling is not a real-time scheduling algorithm by default because it does not consider task deadlines

        - To be considered a real-time scheduling algorithm, the algorithm must ensure that tasks are scheduled in a way that guarantees their deadlines will be met.

2. Priority based
    - Every task is given a priority
    - Implement a priority queue
        - Preemption: if front queue has a higher priority, preempt
    - We ensure more important tasks get more CPU time
    - How do we assign priority?