What's wrong with priorities?

- Fixed priorities:
  - Should I be #4? … #6? … #15?

- Dynamic priorities
  - I have no idea what my priority is because the CPU changes it!

Real-time demands

- We don't always need a LOT of CPU time but we may need it at the right intervals
  - E.g., decode 30 frames per second of video

- We might have tight deadlines
  - E.g., complete task within the next 500 ms

- Conventional process scheduling algorithms focused on fairness, compromise, and providing the best overall experience

Deadlines in real-time systems

- Start time (release time)
  - E.g., response to a sensor: start within 20 ms from sense time

- Stop time (deadline)
  - Scheduler must allot enough CPU time to complete

- Hard deadline
  - There is no value to the computation if it completes after the deadline

- Safety critical system: critical start time and deadline

- Soft deadline
  - The value of a late result diminishes with time

Process types

- Terminating process
  - Runs and exits (e.g., service a sensor event)
  - How much time does it take to run to completion?
  - Deadline = time to finish

- Nonterminating process
  - Interested in time between events
    - E.g., fill a 4 KB audio buffer every 500 ms
    - E.g., decode a video frame every 67 ms
  - Compute time = time to compute periodic event
  - Deadline = time to have periodic results ready

How much can we do?

- Don't expect magic

- E.g.,
  - decoding 1 video frame takes 20 ms
  - we want to decode 2 video frames at 30 frames/sec
  - We'll fail: 2 × 30 × 20 = 1200 ms > 1000 ms (1 sec = 1000 ms)

- If $T = \text{period}$, $D = \text{deadline}$, $C = \text{compute time}$:
  \[ C \leq D \leq T \]
Earliest Deadline Scheduling

- Each process tells OS its time deadline
- Scheduler picks the process in closest to its deadline
  - Usually one process runs to completion if it has an earlier deadline
  - Will be preempted if a process with an even earlier deadline starts

Least Slack Scheduling

- Consider remaining time and deadline
- Look not only at the deadline but how much we can procrastinate
  
  \[
  \text{slack} = (\text{time to deadline}) - (\text{amount of computation})
  \]

  E.g., suppose
  
  - \( C \) (compute time) = 5 ms
  - \( D \) (deadline) = 20 ms from now
  - slack = \( D - C = 15 \) ms

Least Slack vs. Earliest Deadline First

Earliest Deadline First
  - We always work on the earliest deadline process and delay others

Least Slack
  - Get a balanced result in that we keep the differences to deadlines balanced

If there’s not enough time for everything:
  - EDF: may hit only the early deadlines
  - LS: all deadlines may be missed but roughly by the same amount

Rate monotonic analysis

- Method of assigning static priorities to periodic processes
- Works with a static priority scheduler
- Must know all real-time processes running at the same time and their period
- Rate monotonic priority assignment is optimal
  - If the it is possible for all deadlines to be met then they will be met with rate monotonic assignment

Assigning priorities

- Highest frequency (smallest period) process gets the highest priority
- Successively lower frequency processes get lower priorities
- Scheduling is via a simple priority scheduler
- If two processes have the same priority, they can round-robin

Rate monotonic example

- Process A runs every 50 ms for 20 ms
- Process B runs every 50 ms for 10 ms
- Process C runs every 30 ms for 10 ms

Rate monotonic analysis:
  - Schedule C first, then A or B

No rate monotonic priority assignment: C misses a period!
The End