

ITEE



What's New:

Some questions this time are better tackled collectively: split the work of e.g. drawing a Gantt chart for each algorithm, then compare notes, otherwise you will run out of time.

Learning objectives for this week:

- Understand the major concepts we've covered, including:
 - CPU scheduling algorithms
 - techniques for evaluating and comparing scheduling algorithms
 - design trade-offs
- Apply these concepts to reasoning about scheduling algorithms
- Apply these concepts to choosing an appropriate algorithm in a given situation
- Understand the performance implications, including the various criteria for optimizing a scheduler

As before, you should aim to answer all of these questions by the end of the course. Being able to answer them by the end of week 5 is a useful goal. Be prepared: if you know what you can't do yourself, you will make most efficient use of the tutorial.

1. Concepts

- a. Why is it useful to differentiate CPU-bound and I/O-bound processes?
- b. Explain why starvation is a problem.
- c. Explain the role that context switching overheads have in designing an optimal scheduling algorithm.

2. Specific Algorithms

- a. Which is easier to implement: shortest job first (SJF) or first come first served (FCFS)? Explain your answer.
- b. Why is waiting time a useful criterion to evaluate alternative scheduling algorithms?
- c. Explain why preemptive SJF is better than nonpreemptive SJF. Use an example to illustrate your point.
- d. Explain why *exponential averaging* is a useful way to use history to estimate the length of the next CPU burst. If we used exponential averaging to calculate the overall mark for a course with several assignments and a final exam, what value of α would you prefer? Why?

3. Evaluation of Algorithms

- a. For the following scenario, draw Gantt charts for each of SJF, preemptive SJF and round robin (time quantum 10):

process	arrival time	burst time
P ₁	0	53
P ₂	10	17
P ₃	20	68
P ₄	30	24

- b. Based on answers of (a), which algorithm gives the shortest average wait time? Given the complexity of estimating how long the next CPU burst is, would you choose SJF over round robin, *given the data here*? What might we be missing which could change the answer?
- c. In a proposed scheduling scheme, all processes start with priority 0, which means they can be processed immediately if ready and at the head of queue 0. If they fail to complete a time quantum (10 time units), their priority is reduced to 1 (in this case, bigger numbers are worse) and they drop to queue 1. Queue 1 processes are never scheduled until there are no ready queue 0 processes. A queue 1 process has a time quantum of 20, and is promoted to queue 0 if it does not complete its time quantum (e.g., because it has to request I/O).
 - i. Which of the previously considered schemes in this question does this approach *most closely* resemble?
 - ii. Considering all schemes covered in lectures, what would you call this proposed new scheme?
 - iii. What major flaw does it have, and how could you fix the flaw? Describe your improved version.
 - iv. Redo part (a) with the variations in this question (original and your improved version).
- d. Consider values of $\alpha = 0, 1, 0.25$ and 0.5 . How does the value of τ vary with each value of α , with $\tau_0 = 32$, and successive values of $t = 16, 12, 8, 4, 100, 100$? (If you have time, draw a graph in the style of Figure 6.3, p 160 for each value of τ . Note that the horizontal axis of the graph represents time quanta.)

4. Discussion Questions

- a. Can a real-time scheduler can be implemented in conjunction with a general-purpose operating system? Discuss.
 - b. Multilevel feedback queues are a mechanism to balance throughput with responsiveness. Discuss how they can achieve both goals, looking at any real operating system of your choice.
5. Do the questions in the week 4 classroom exercise (available in [PowerPoint](#) and [PDF](#), as well as in the lecture notes, if you bought the whole set) if you didn't do them already.