

Scheduler Goals By System Type
For batch (non-interactive) systems, possible goals (might conflict):

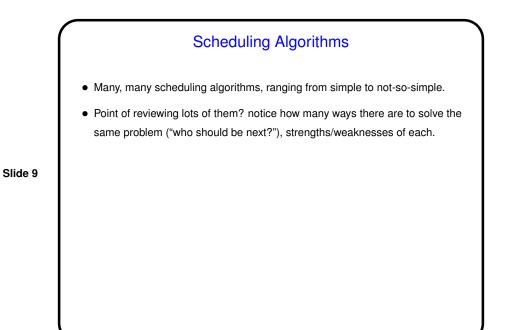
Maximize throughput — jobs per hour.
Minimize turnaround time.
Maximize CPU utilization.

Preemptive scheduling may not be needed.
For interactive systems, possible goals:

Minimize response time.
Make response time proportional (to user's perception of task difficulty).
Preemptive scheduling probably needed.

For real-time systems, possible goals:

Meet time constraints/deadlines.
Behave predictably.



First Come, First Served (FCFS)
Basic ideas:

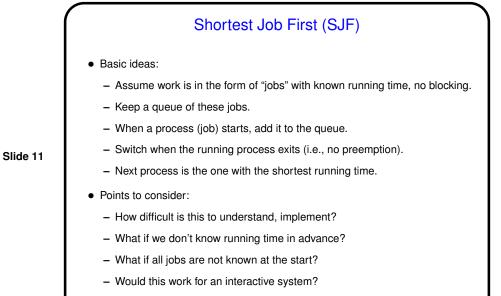
Keep a (FIFO) queue of ready processes.
When a process starts or becomes unblocked, add it to the end of the queue.

Slide 10

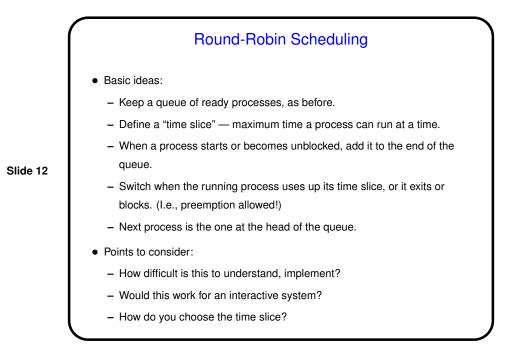
Switch when the running process exits or blocks. (I.e., no preemption.)
Next process is the one at the head of the queue.

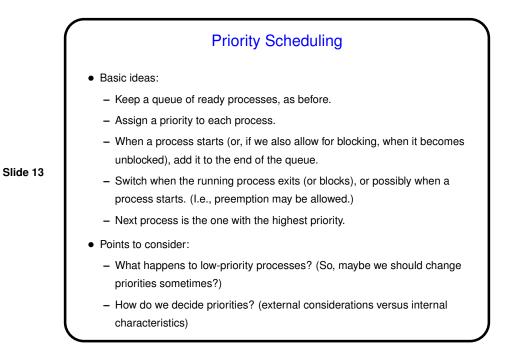
Points to consider:

How difficult is this to understand, implement?
What happens if a process is CPU-bound?
Would this work for an interactive system?

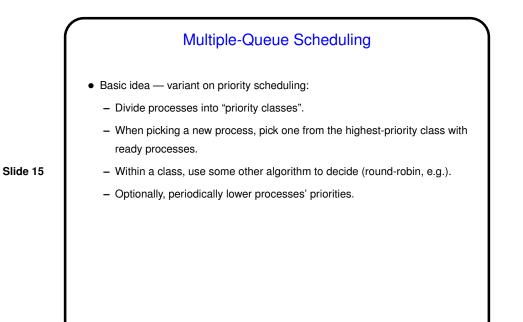


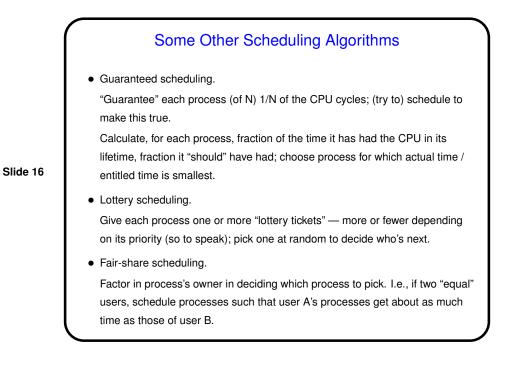
- What's the key advantage of this algorithm?





Shortest Remaining Time Next Basic idea — variant on SJF: Assume that for each process (job), we know how much longer it will take. Keep a queue of ready processes, as before; add to it as before. Switch when the running process exits *or* a new process starts. (I.e., preemption allowed — requires recomputing time left for preempted process.) Next process is the one with the shortest time left. Points to consider: How does this compare with SJF?



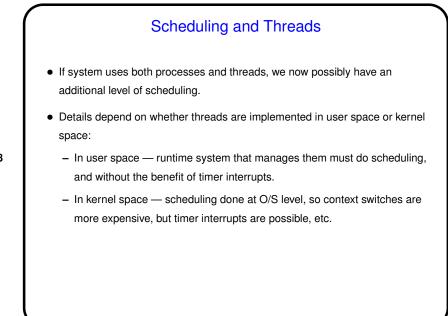


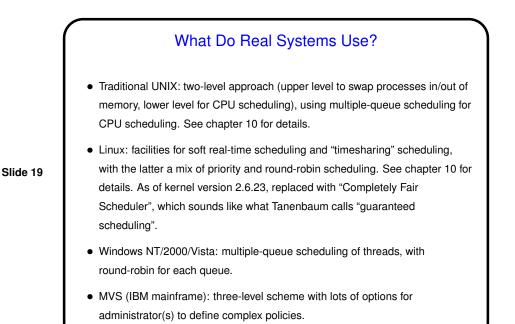
Scheduling in Real-Time Systems

 "Real-time system" — system in which events must ("hard real time") or should ("soft real time") be handled by some deadline. Often events to be handled are periodic, and we know how often they arrive and how long they take to process.

Slide 17

- Role of scheduler in such systems could be critical.
- An interesting question sometimes getting everything scheduled on time is impossible (example?). If we know periodicity and time-to-handle of all types of events, can we decide this? (Yes — general formula in textbook; can be interesting to work through details.)
- Complex topic; see chapter 7 for more info.

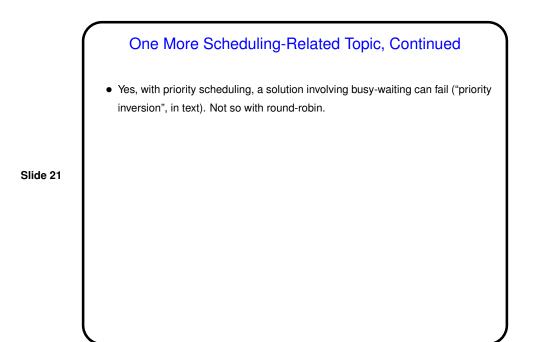


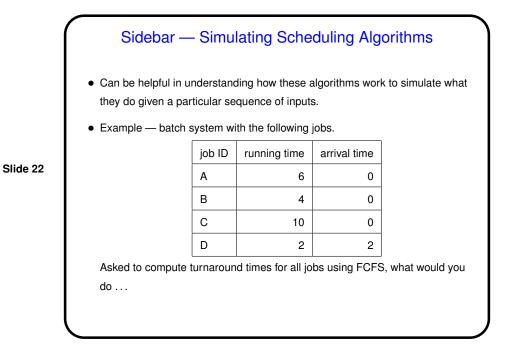


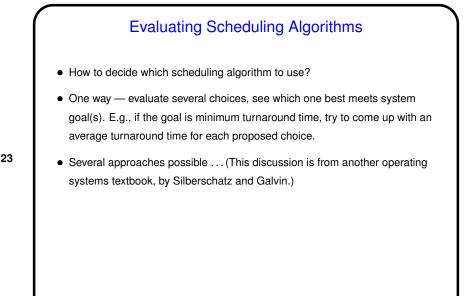
One More Scheduling-Related Topic

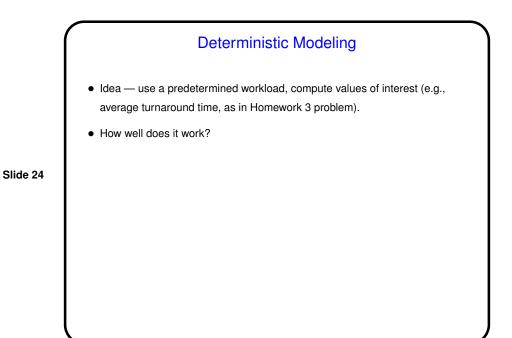
• A question I used to use as homework:

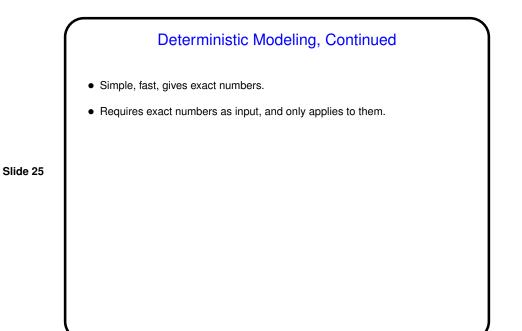
Recall that some proposed solutions to the mutual-exclusion problem (e.g., Peterson's algorithm) involve busy waiting. Do such solutions work if priority scheduling is being used and one of the processes involved has higher priority than the other(s)? Why or why not? How about if round-robin scheduling is being used? Why or why not? Notice that a process can be interrupted while in its critical region; if that happens, it is considered to still be in its critical region, and other processes wanting to be in their critical regions are supposed to busy-wait.

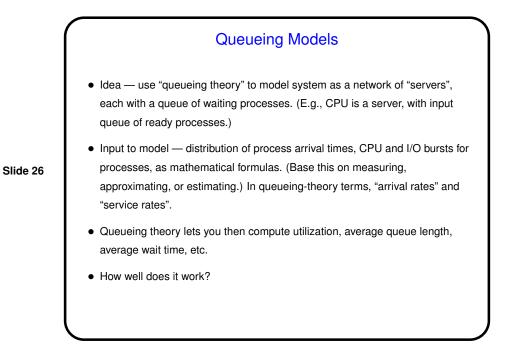


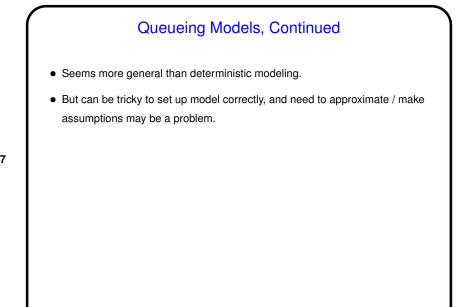


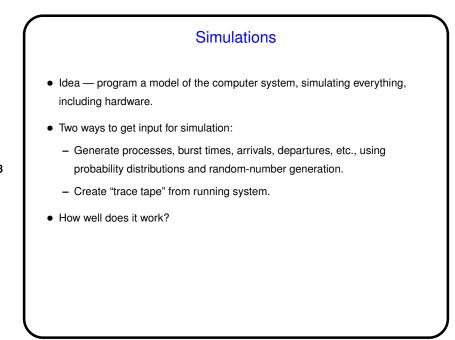


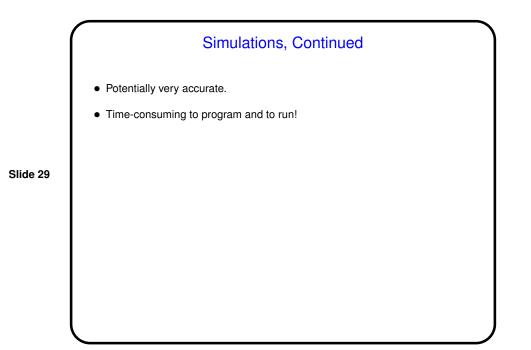


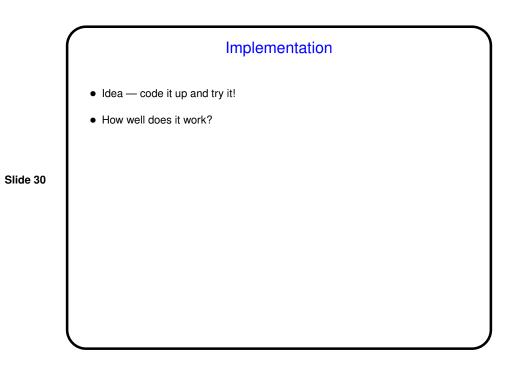


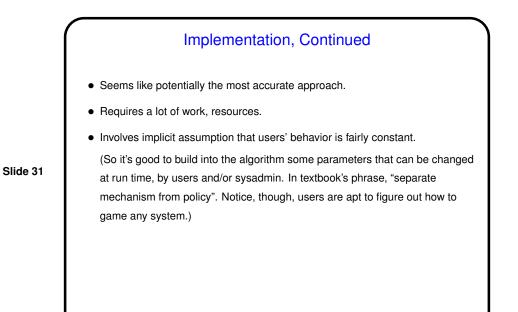












Recap — Scheduling Algorithms • Main idea — decide which process to run next (when running process exits, becomes blocked, or is interrupted). • Many possibilities, ranging from simple to complex. Real systems seem to use hybrid strategies. Slide 32 • How to choose one? - Be clear on goals. - Maybe evaluate some possibilities to see which one(s) meet goals analytic or experimental evaluation. - Build in some tuning knobs - "separate policy from mechanism".

