

## Short-term scheduling algorithms

- **First-Come, First-Served (FCFS, or FIFO)**
  - as process becomes ready, join Ready queue, scheduler always selects process that has been in queue longest
  - better for long processes than short ones
  - favors CPU-bound over I/O-bound processes
  - need priorities, on uniprocessor, to make it effective

## Algorithms (cont.)

- **Round-Robin (RR)**
  - use preemption, based on clock - time slicing
    - generate interrupt at periodic intervals
  - when interrupt occurs, place running process in Ready queue, select next process to run using FCFS
  - what's the length of a time slice
    - short means short processes move through quickly, but high overhead to deal with clock interrupts and scheduling
    - guideline is time slice should be slightly greater than time of "typical job" CPU burst
  - problem dealing with CPU and I/O bound processes

## Algorithms (cont.)

- Shortest Process Next (SPN)

- non-preemptive
- select process with shortest expected processing time
- improves response time, but increases its variability, reducing predictability - provably decreases average waiting time
- problem is estimating required processing time
- risk of starving longer processes, as long as there are shorter processes around
- not good for time sharing - non-preemptive

## Algorithms (cont.)

- Shortest Remaining Time (SRT)

- preemptive version of SPN
- scheduler chooses process with shortest expected remaining process time
- still need estimate of processing time, and can starve longer processes
  - no bias in favor of longer processes, as in FCFS
  - no extra interrupts as in RR, so reduced overhead
- must record elapsed service times
- should give better turnaround time than SPN