Announcements

- Reading
  - Project #1 – due 2/15/17 at 5:00 pm
  - Scheduling
    - Chapter 6 (6th ed) or Chapter 5 (8th ed)

Dispatcher

- The inner most part of the OS that runs processes
- Responsible for:
  - saving state into PCB when switching to a new process
  - selecting a process to run (from the ready queue)
  - loading state of another process
- Sometimes called the short term scheduler
  - but does more than schedule
- Switching between processes is called context switching
- One of the most time critical parts of the OS
- Almost never can be written completely in a high level language

Selecting a process to run

- called scheduling
- can simply pick the first item in the queue
  - called round-robin scheduling
  - is round-robin scheduling fair?
- can use more complex schemes
  - we will study these in the future
- use alarm interrupts to switch between processes
  - when time is up, a process is put back on the end of the ready queue
  - frequency of these interrupts is an important parameter
    - typically 10-100ms on systems today
      - Time has been getting longer over past 30 years
  - need to balance overhead of switching vs. responsiveness

CPU Scheduling

- Manage CPU to achieve several objectives:
  - maximize CPU utilization
  - minimize response time
  - maximize throughput
  - minimize turnaround time
- Multiprogrammed OS
  - multiple processes in executable state at same time
  - scheduling picks the one that will run at any give time (on a uniprocessor)
- Processes use the CPU in bursts
  - may be short or long depending on the job

Types of Scheduling

- At least 4 types:
  - long-term - add to pool of processes to be executed
  - medium-term - add to number of processes partially or fully in main memory
  - short-term - which available process will be executed by the processor
  - I/O - which process's pending I/O request will be handled by an available I/O device
- Scheduling changes the ***state*** of a process

Scheduling criteria

- Per processor, or system oriented
  - CPU utilization
    - maximize, to keep as busy as possible
  - throughput
    - maximize, number of processes completed per time unit
- Per process, or user oriented
  - turnaround time
    - minimize, time of submission to time of completion.
  - waiting time
    - minimize, time spent in ready queue - affected solely by scheduling policy
  - response time
    - minimize, time to produce first output
    - most important for interactive OS

Scheduling criteria non-performance related

- Per process
  - predictability
    - job should run in about the same amount of time, regardless of total system load
- Per processor
  - fairness
    - don’t starve any processes, treat them all the same
  - enforce priorities
    - favor higher priority processes
  - balance resources
    - keep all resources busy

In Class Exercise

- Give each group 45 minutes
  - to construct their scheduling algorithm.
  - The algorithm should take a list of runnable processes and pick **one** to run next
  - Any criteria can be used
  - May keep data about processes, but need to describe what it is
- Have each group describe their algorithm
  - Ask the others if it does what they claim it does
  - Offer your own critiques of the algorithm
  - If one of the groups repeats another, still have them describe it
    - Look for any differences in how it achieves its goal

Short-term scheduling algorithms

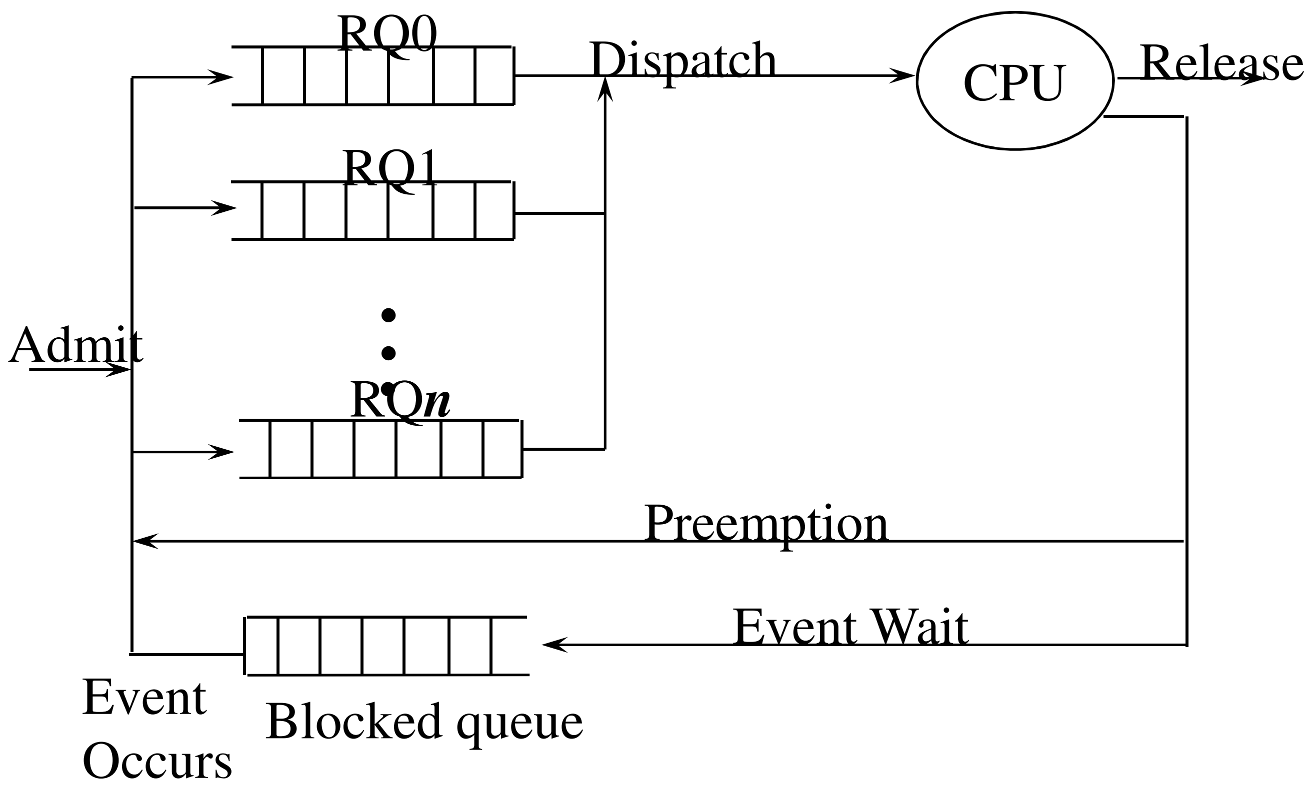
- First-Come, First-Served (FCFS, or FIFO)
  – as process becomes ready, join Ready queue, scheduler always selects process that has been in queue longest
  – better for long processes than short ones
  – favors CPU-bound over I/O-bound processes
  – need priorities, on uniprocessor, to make it effective

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

Priority Based Scheduling

- Priorities
  - assign each process a priority, and scheduler always chooses process of higher priority over one of lower priority
- More than one ready queue, ordered by priorities



Priority Algorithms

- Fixed Queues
  - processes are statically assigned to a queue
  - sample queues: system, foreground, background
- Multilevel Feedback
  - processes are dynamically assigned to queues
  - penalize jobs that have been running longer
  - preemptive, with dynamic priority
  - have ***N*** ready queues (RQ0-RQ***N***),
    - start process in RQ0
    - if quantum expires, moved to i + 1 queue