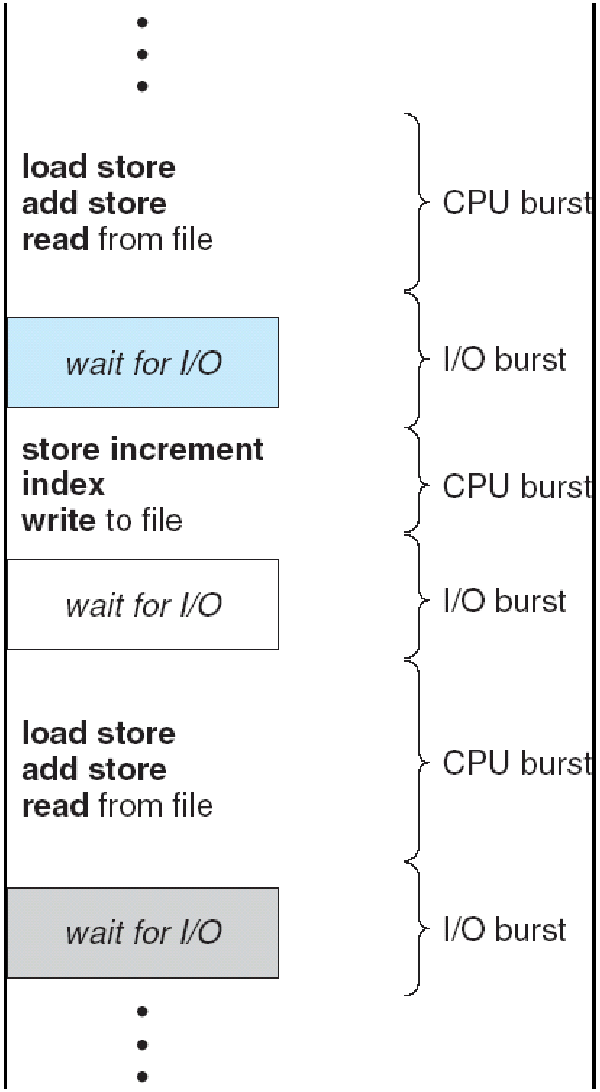
Part 3: Process Scheduling

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms

Basic Concepts

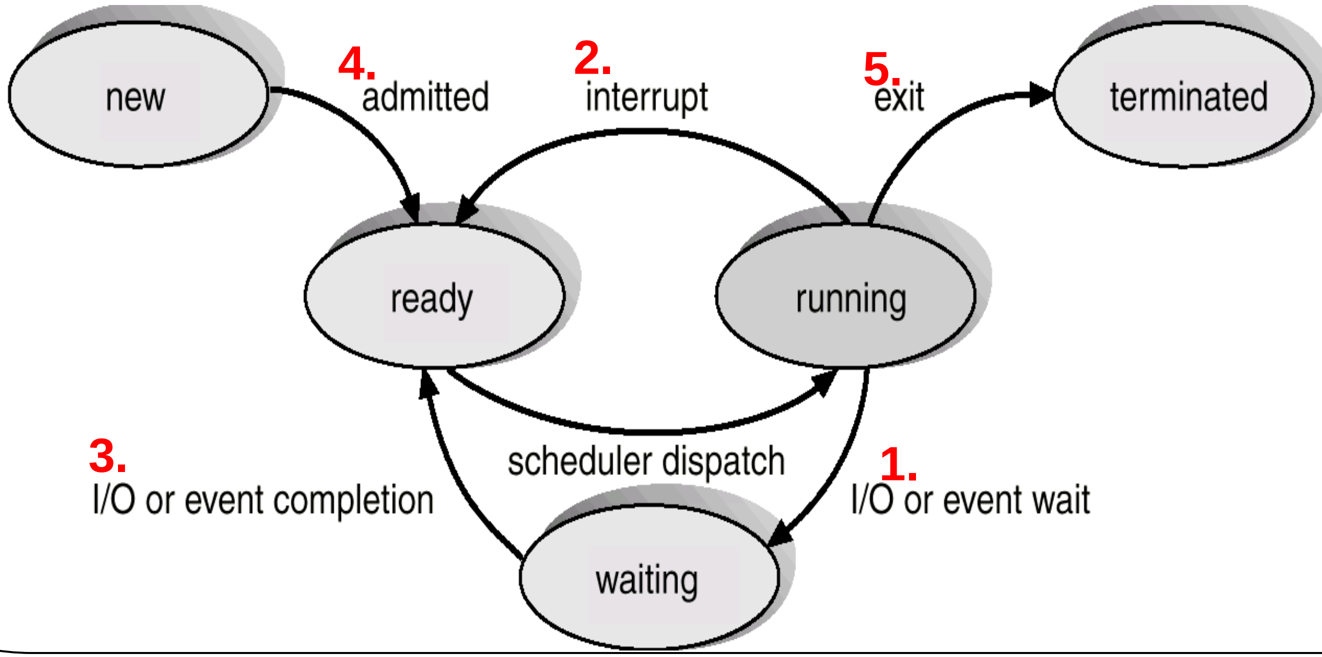
- CPU–I/O Burst Cycle – Process execution consists of a *cycle* of CPU execution and I/O wait.
  - CPU burst: the period for CPU executions.
  - I/O burst: the period for I/O wait.
- The goal of scheduling is to keep the CPU busy (i.e., improve CPU utilization with multiprogramming).
- For simplicity, we focus on the concepts of **short-term scheduler** on a single CPU.

Alternating Sequence of CPU And I/O Bursts



CPU Scheduler

- Selects one of the processes in the ready queue, and allocates the CPU to it.
- CPU scheduling decisions may take place when a process changes state.
  - Which case(s) does scheduling must occur in order to keep the CPU busy?



CPU Scheduler (cont.)

- For situations 1 and 5, there is no choice. However, there is a choice for situations 2, 3 & 4.
  - **Nonpreemptive** means that once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or requesting I/O/event wait.
    - *Scheduling is nonpreemptive if it happens **only** under situations 1 and 5*
  - **Preemptive** means that the CPU can be taken away from running process at any time by OS.
    - *Scheduling is preemptive if it also happens under situations 2, 3 and 4*

Scheduling Criteria

- CPU utilization – keep the CPU as busy as possible
- Throughput – # of processes that complete their execution per time unit
- Turnaround time – amount of time to execute a particular process, i.e. from the time of submission to the time of completion.

Scheduling Criteria

- Waiting time – amount of time a process has been waiting in the ready queue.
  - Turn-around time consists of three components: CPU burst, I/O burst, and waiting time.
  - If all processes have a single CPU burst (no I/O),

  waiting time= turn-around time – CPU burst time.

- Response time – amount of time it takes from when a request was submitted until the **first** response is produced

Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- **Min waiting time**
- Min response time

Overview of Scheduling Algorithms

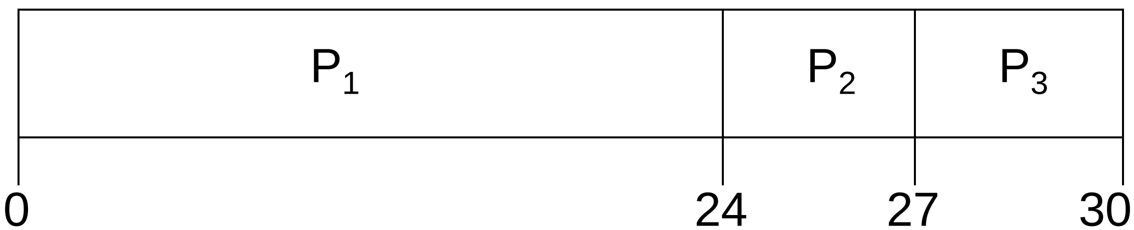
- First-Come, First-Served (FCFS)
- Shortest Job First (SJF)
- Priority Scheduling
- Round Robin (RR)
- Multilevel Queue Scheduling

First-Come, First-Served (FCFS) Scheduling

- Example:

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that processes arrive in the order: P_1 , P_2 , P_3 all at time 0. The Gantt Chart for the schedule is:



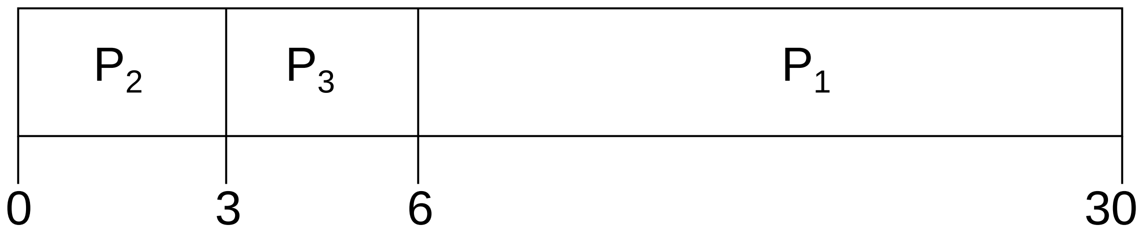
- Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Average waiting time: $(0 + 24 + 27)/3 = 17$

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order

$$P_2 \, , \, P_3 \, , \, P_1 \; .$$

- The Gantt chart for the schedule is:



- Waiting time for $P_1 = 6; P_2 = 0; P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case.

FCFS Scheduling (Cont.)

- Problem of FCFS:
  – will cause c*onvoy effect,* that means short processes suffer because they have to wait for long process to finish.

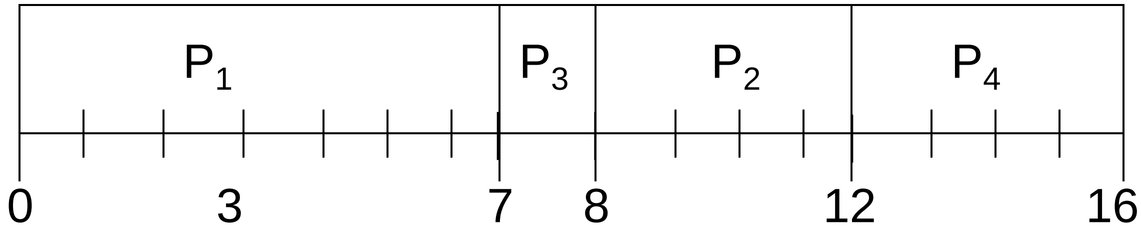
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time.
- Two schemes:
  - nonpreemptive – once CPU given to the process it cannot be preempted until completes its CPU burst.
  - Preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the Shortest-Remaining-Time-First (**SRTF**).
- SJF is optimal – gives minimum average waiting time for a given set of processes.

Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)

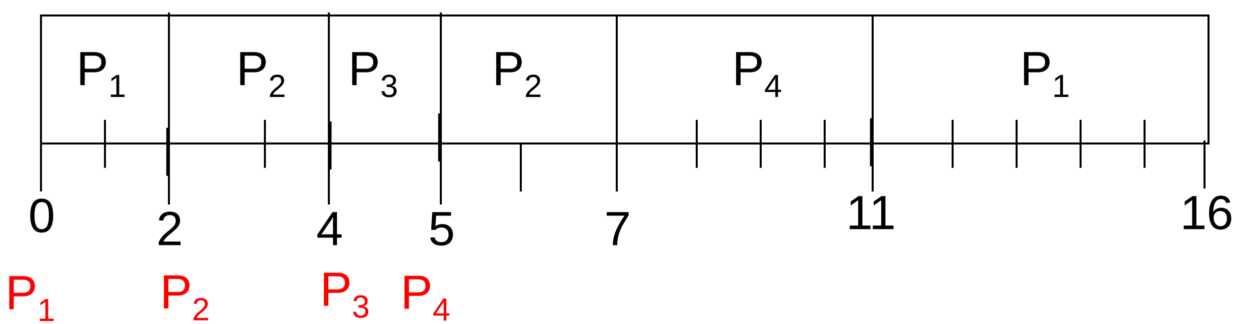


- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

Example of Preemptive SJF (i.e. SRTF)

Process	Arrival Time	Burst Time
P_1	0.0	7 $\rightarrow$ 5 $\rightarrow$ 0
P_2	2.0	4 $\rightarrow$ 2 $\rightarrow$ 0
P_3	4.0	1 $\rightarrow$ 0
P_4	5.0	4 $\rightarrow$ 0

- SJF (preemptive)



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer $\equiv$ highest priority).
  - Preemptive
  - nonpreemptive
- SJF is a priority scheduling where priority is the predicted next CPU burst time.
- Problem $\equiv$ Starvation – low priority processes may never execute.
- Solution $\equiv$ **Aging** – as time progresses increase the priority of the process.

Round Robin (RR)

- Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are n processes in the ready queue and the time quantum is q, then no process waits more than $(n\text{-}1)q$ time units.
- Performance
  - q large $\Rightarrow$ FCFS
  - q small $\Rightarrow$ q must be large with respect to context switch, otherwise overhead is too high.

Example: RR with Time Quantum = 20

Process	Burst Time
P_1	53 → 33
P_2	17 → 0
P_3	68 → 48
P_4	24 → 4

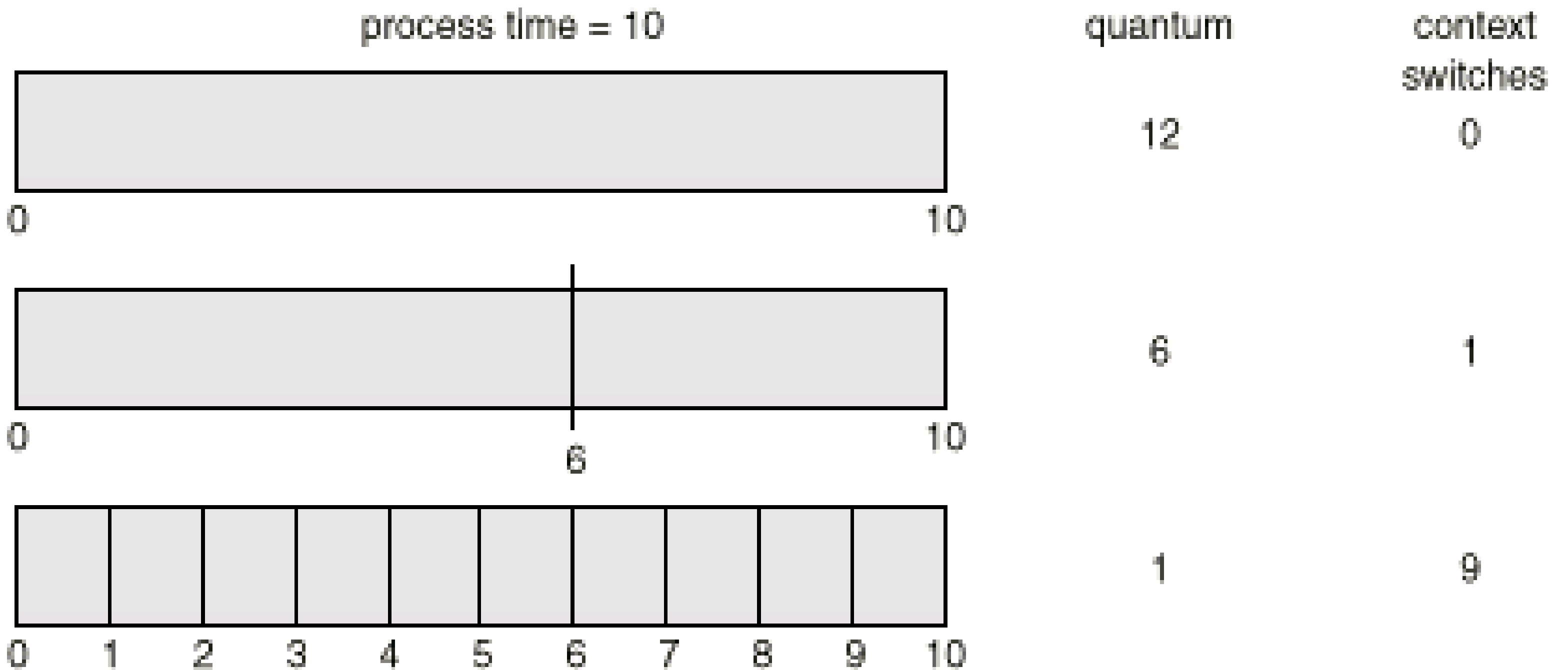
- The Gantt chart is (all processes come at time zero, in the order of P1, P2, P3, P4):

P_1	P_2	P_3	P_4	P_1	P_3	P_4	P_1	P_3	P_3

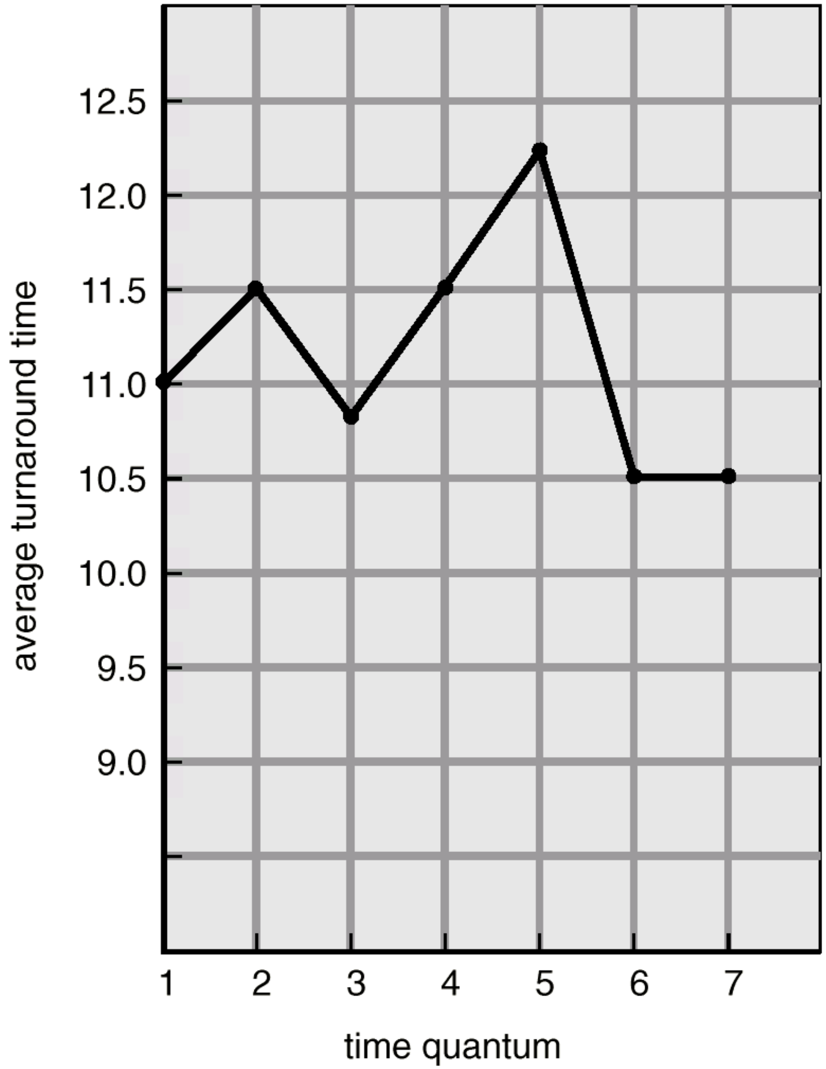
0 20 37 57 77 97 117 121 134 154 162

- Typically, higher average turnaround than SJF, but better *response time*.

How a Smaller Time Quantum Increases Context Switches



Turnaround Time Varies With The Time Quantum

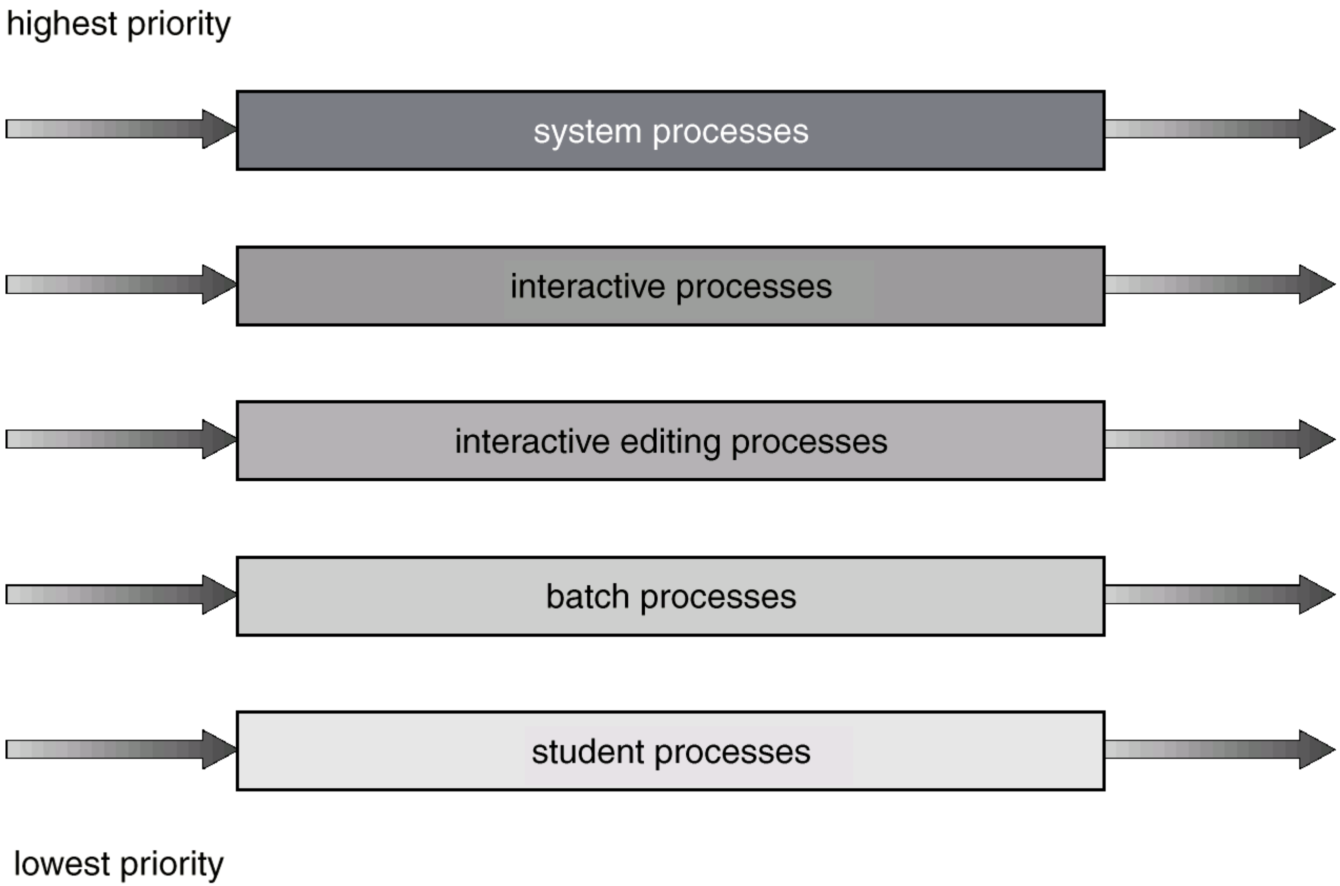


process	time
P_1	6
P_2	3
P_3	1
P_4	7

Multilevel Queue

- Ready queue is partitioned into separate queues:
  foreground (e.g. interactive)
  background (e.g. batch)
- Each queue has its own scheduling algorithm,
  foreground – e.g. RR
  background – e.g. FCFS

Multilevel Queue Scheduling



Multilevel Queue (Cont.)

- Scheduling must be done between the queues.
  - Fixed priority scheduling; i.e., serve all from foreground then from background. Possibility of starvation for the background processes.
  - Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; examples are:
    - 80% to foreground queue
    - 20% to background queue

Advanced Readings

- The following animations are very good tools for learning CPU scheduling.
  - http://www.utdallas.edu/~ilyen/animation/cpu/program/prog.html
  - http://cs.uttyler.edu/Faculty/Rainwater/COSC3355/Animations/