Operating System Design

Processes

Neda Nasiriani

Fall 2018

Exam 1 is next week!

- You should read/review the following materials to do well on the exam
  - Reading assignments
  - Activities
  - Quizzes
  - Labs (and Pre-labs of course!)
  - Class notes
- Exam dynamics
  - You can bring 2 two-sided US-letter cheat sheet with **hand written** notes on those! If you have difficulty writing it by hand let me know before the exam
  - It will be 56 minute exam during the class hour, try to be on time

Process Synchronization wrap up Quiz!

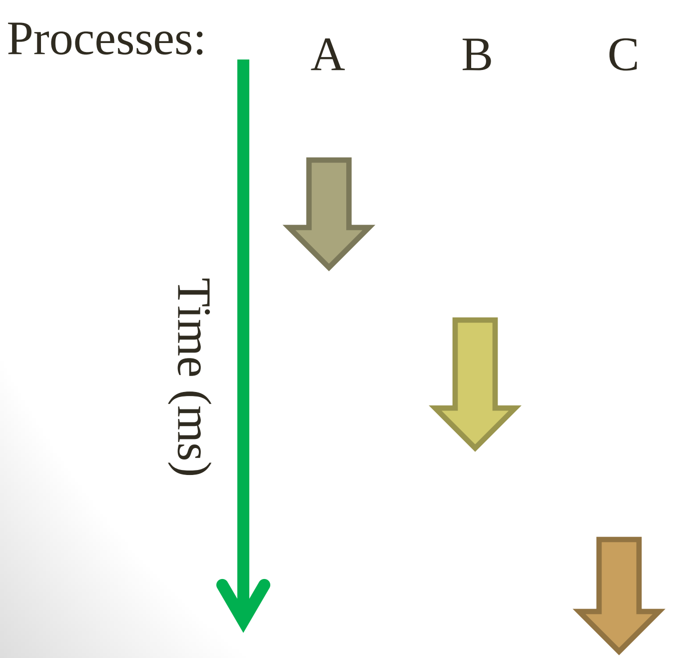
20 minutes.

Quiz Answer!

Processes From CPU Perspective: Scheduling

Specs of the multi-processor Computer System

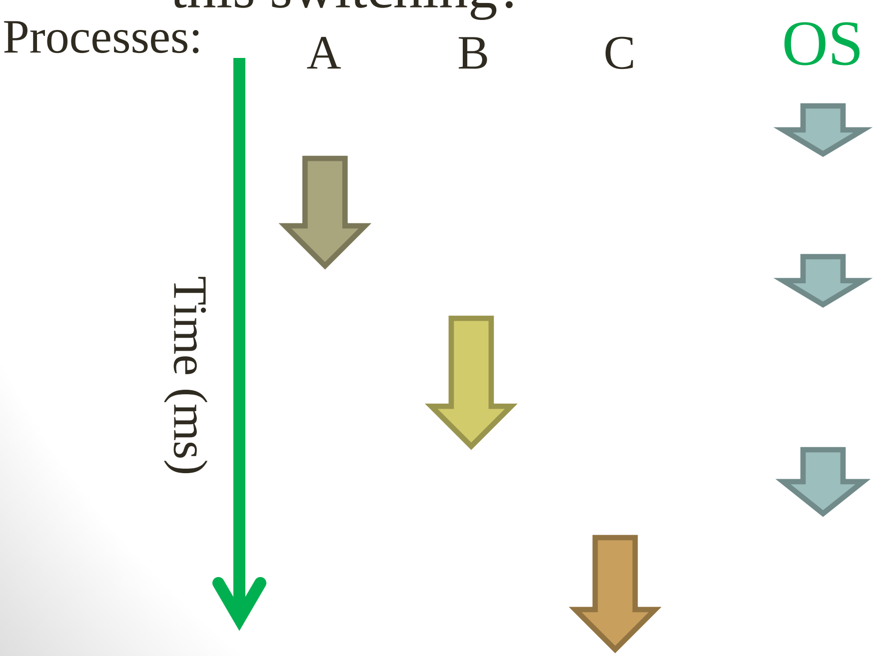
- We want processes to run concurrently, so (i) they can interact with each other, and (ii) maximize CPU utilization
  - Fact: at each time only one process can run on each processor
  - Remedy: So, we should switch processes fast enough so they feel like they are all running simultaneously (illusion)



How can this be implemented in a real computer system?

Specs of the multi-processor Computer System

- We want processes to run concurrently, so (i) they can interact with each other, and (ii) maximize CPU utilization
  - Fact: at each time only one process can run on each processor
  - Remedy: So, we should switch processes fast enough so they feel like they are all running simultaneously (illusion)
- Can the OS kernel as the main process in the system perform this switching?



OS tasks?

- deciding who should run next,
- Handle interrupts if any happened
- …

Specs of the multi-processor Computer System

- We want processes to run concurrently, so (i) they can interact with each other, and (ii) maximize CP... utiliz...
  - Fact: at e... ...sor
  - Rem... ...el like
- Ca...
  t...

Processes:

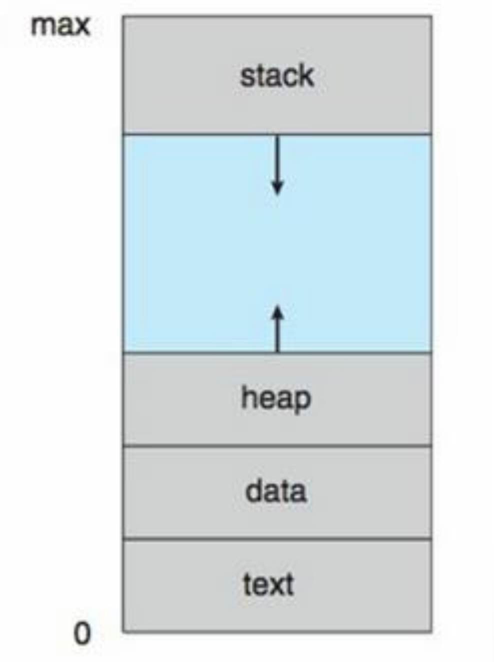
Time (ms)

What does the OS need to know about the Processes to be able to do this Switching?

...s?
- ...ho should run next,
- Handle interrupts if any happened
- …

Processes Components

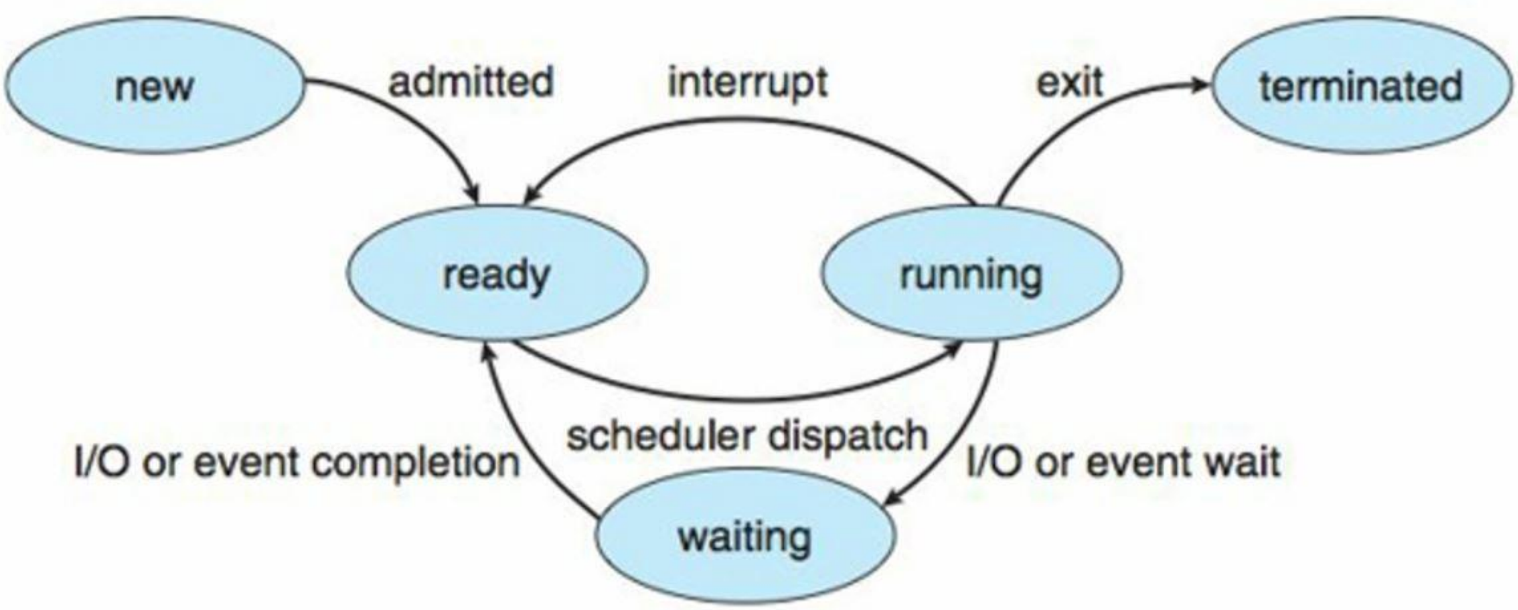
- A process is associated with the following components
  - Text section
  - Data section
  - Heap
  - Stack
  - Program Counter
  - Value of Registers
- The process in memory looks like this



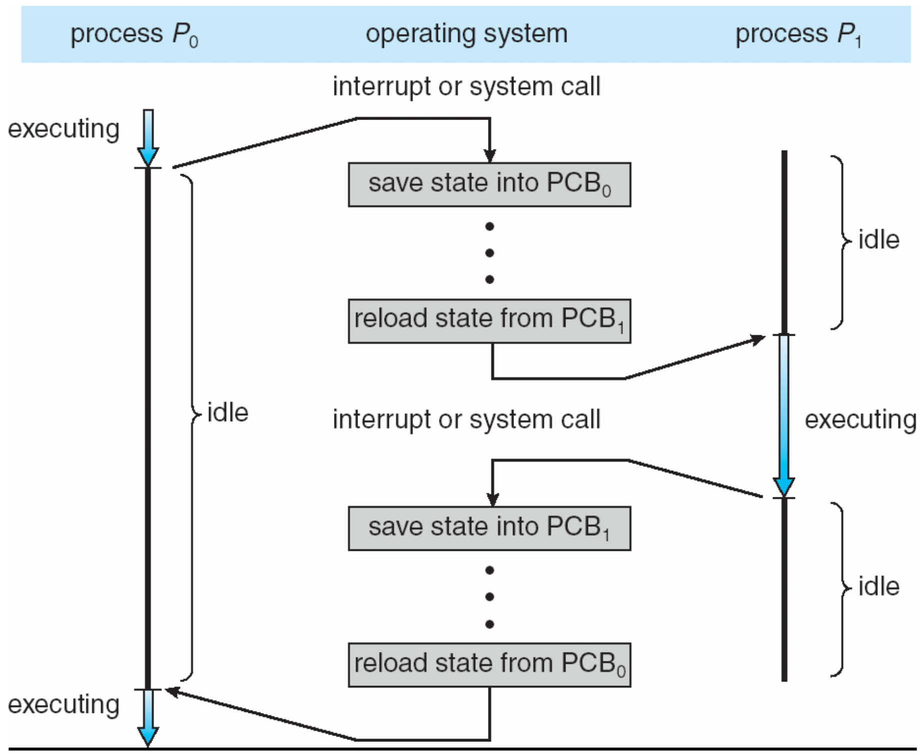
What other information is needed?

- If you want to design a scheduler to divide your time resource between a bunch of different processes, what info would you need in order to schedule effectively and fairly
  - Process state – running, waiting, etc
  - Program counter – location of instruction to next execute
  - CPU registers – contents of all process-centric registers
  - CPU scheduling information- priorities, scheduling queue pointers
  - Memory-management information – memory allocated to the process
  - Accounting information – CPU used, clock time elapsed since start, time limits
  - I/O status information – I/O devices allocated to process, list of open files

Process Lifecycle



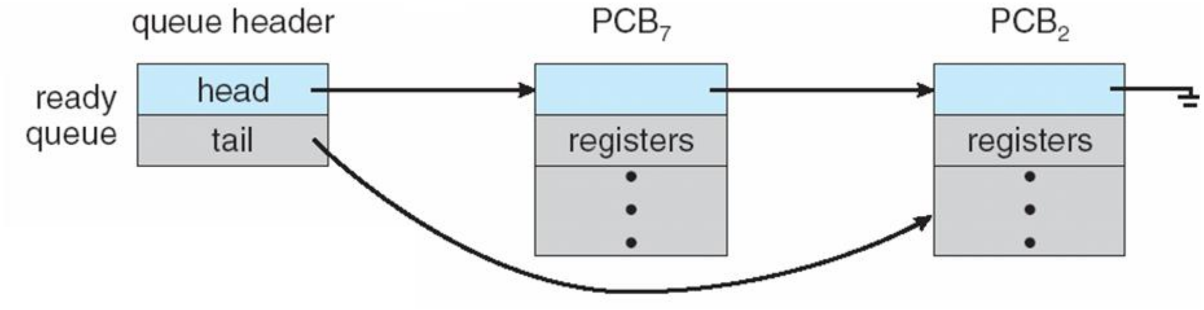
CPU Switch between Processes

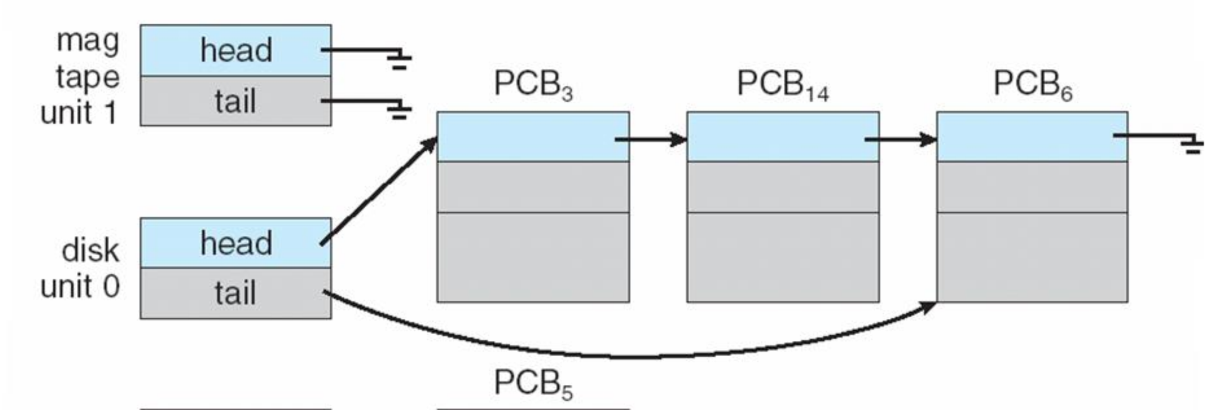


- Context Switch: When CPU switches to another process, the system must **save the state** of the old process and load the **saved state** for the new process via a **context switch**
- This time is pure overhead!

Scheduler

- A list of all processes PCBs is available to OS scheduler

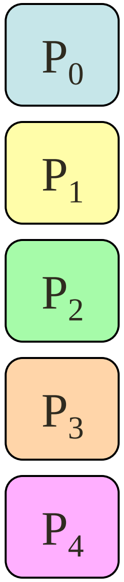


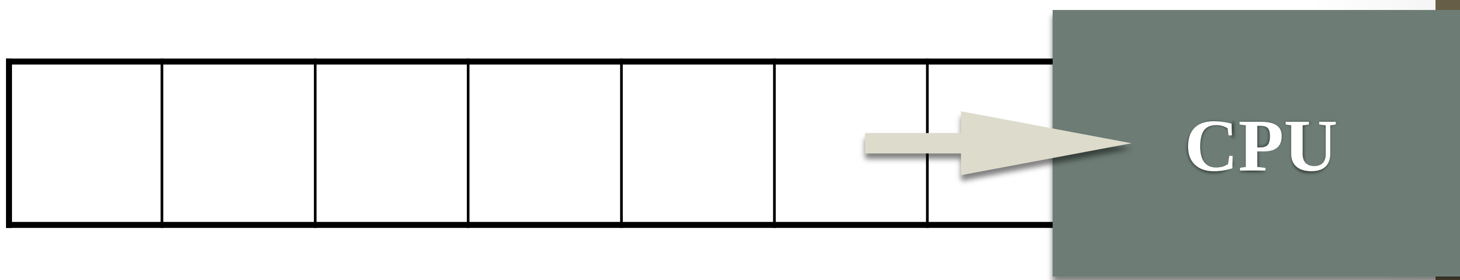
- Ready queue: a list of all processes which are ready and waiting to execute
- Device queue: a list of all processes waiting for an I/O operation on a device, e.g., Disk queue, terminal queue
- 

Scheduler

- **Short-term scheduler** (or **CPU scheduler**) – selects which process should be executed next and allocates CPU
  - Sometimes the only scheduler in a system
  - Short-term scheduler is invoked frequently (milliseconds) $\Rightarrow$ (must be fast)
- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
  - Long-term scheduler is invoked infrequently (seconds, minutes) $\Rightarrow$ (may be slow)
  - The long-term scheduler controls the **degree of multiprogramming**
- Processes can be described as either:
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts
- Long-term scheduler strives for good ***process mix***

Basic Concepts





Questions:

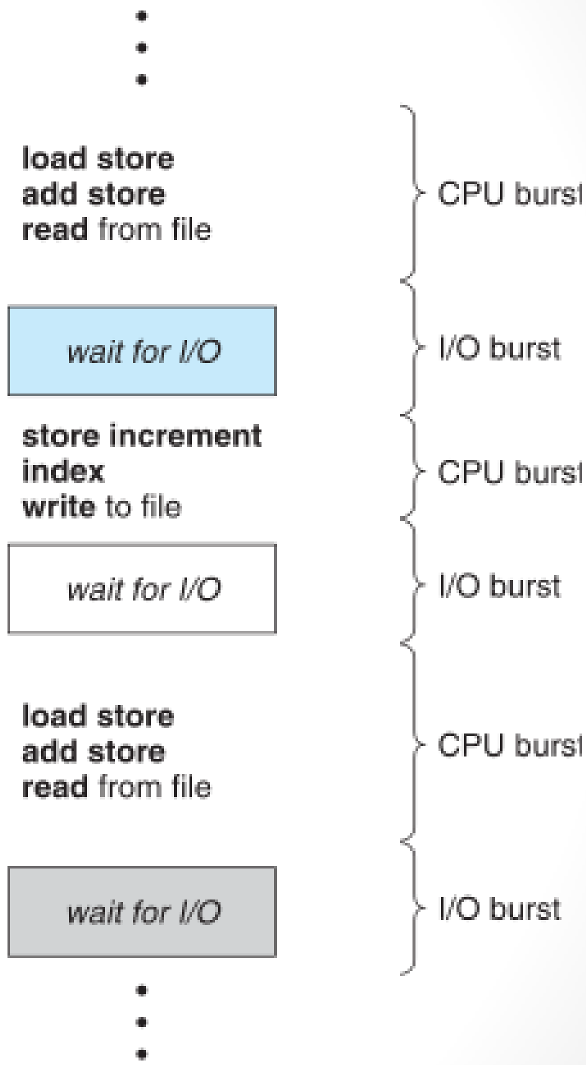
- When does a process start competing for the CPU?
- How is the queue of ready processes organized?
- How much time does the system allow a process to use the CPU?
- Does the system allow for priorities and preemption?
- What does it mean to maximize the system's performance?

Basic Concepts

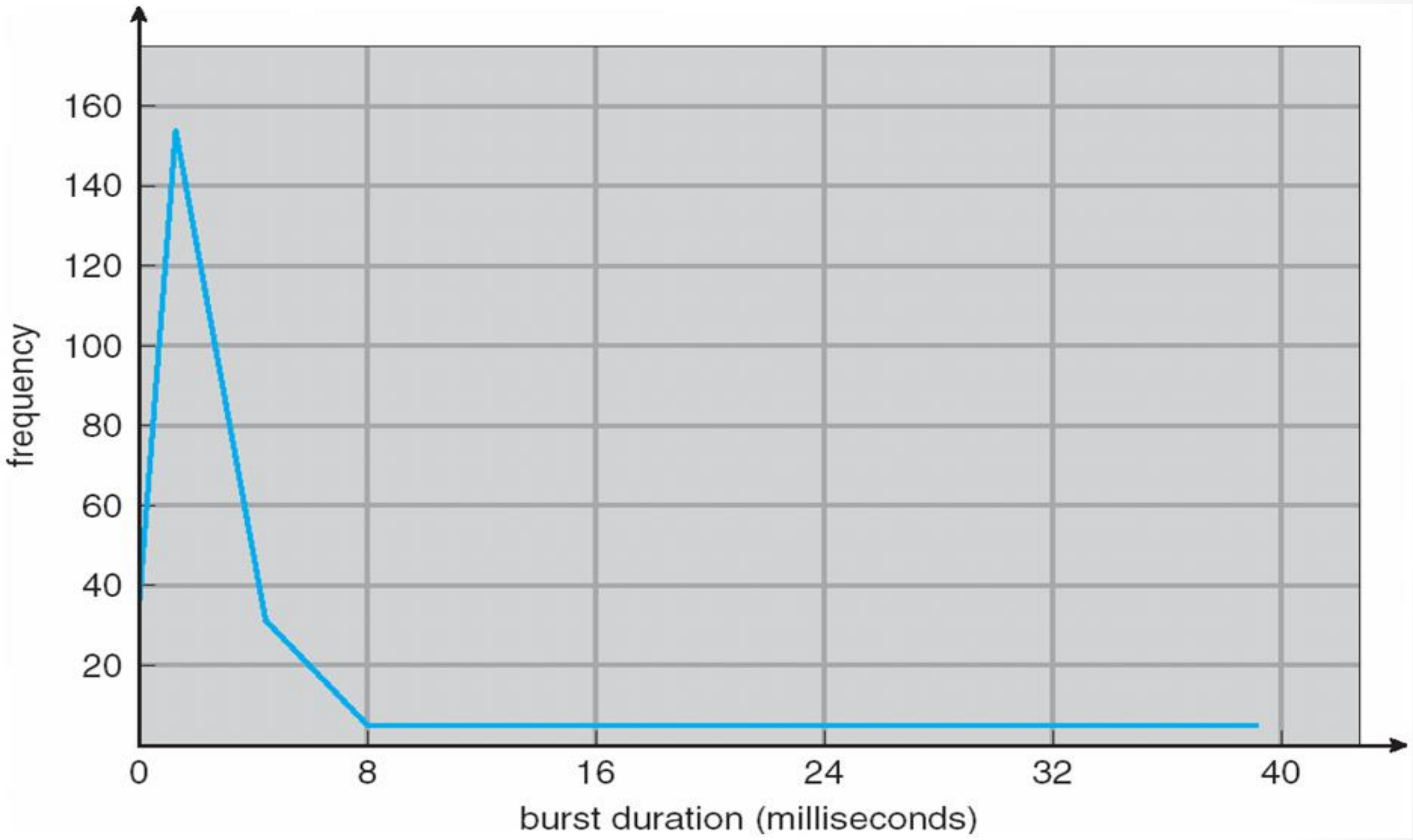
- You want to maximize CPU utilization through the use of multiprogramming.
- Each process repeatedly goes through cycles that alternate CPU execution (a CPU burst) and I/O wait (an I/O wait).
- Empirical evidence indicates that CPU-burst lengths have a distribution such that there is a large number of short bursts and a small number of long bursts.

Alternating Sequence of CPU and I/O Bursts

- Goal: maximize CPU utilization with multiprogramming
- Process execution consists of cycles of CPU execution and I/O wait
- A CPU burst is followed by an I/O burst
- The probability distribution of CPU bursts is an important concern



Histogram of CPU-burst Times



CPU Scheduler

- AKA *short-term scheduler*.
- Selects from among the processes in memory, which are ready queue and has the dispatcher give the CPU to one of them.

- The schedule needs to execute when a process:
  1. Switches from running to waiting state,
  2. Switches from running to ready state,
  3. Switches from waiting to ready,
  4. Terminates.

What is a good scheduler?

High CPU util

High Throughput

Low waiting time

…

Scheduling Criteria

These are performance metrics such as:

- CPU utilization – high is good; the system works best when the CPU is kept as busy as possible.
- Throughput – the number of processes that complete their execution per time unit.
- Turnaround time – amount of time to execute a particular process.
- Waiting time – amount of time a process has been waiting in the ready queue.
- Response time – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment).

It makes sense to look at averages of these metrics.

Optimizing Performance

- **Maximize** CPU utilization.
- **Maximize** throughput.
- **Minimize** turnaround time.
- **Minimize** waiting time.
- **Minimize** response time.