Amirkabir University of Technology
(Tehran Polytechnic)

Operating Systems

CPU Scheduling-Part1

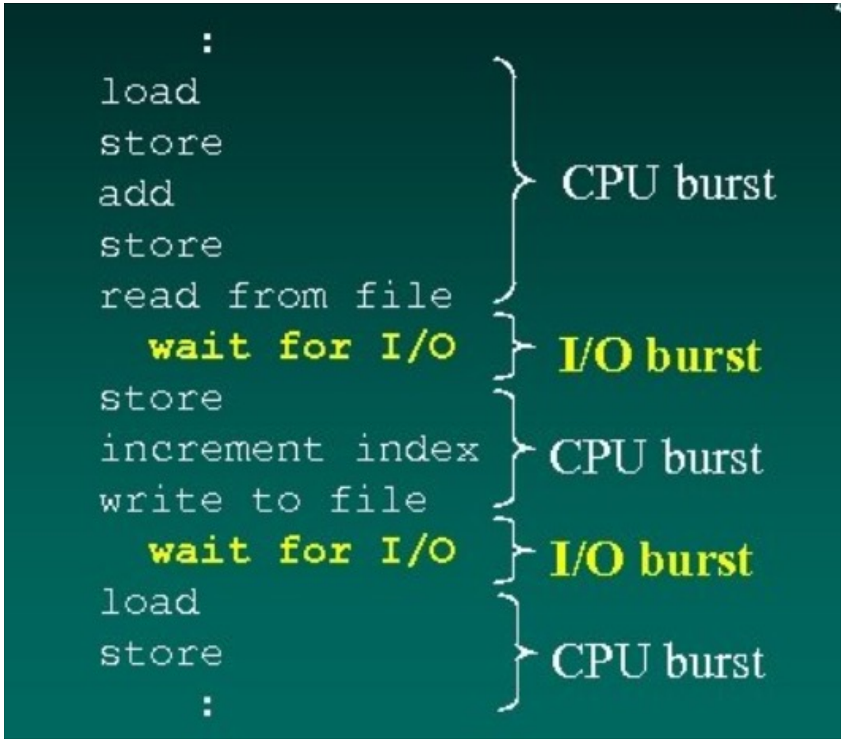
Seyyed Ahmad Javadi

sajavadi@aut.ac.ir

Fall 2021

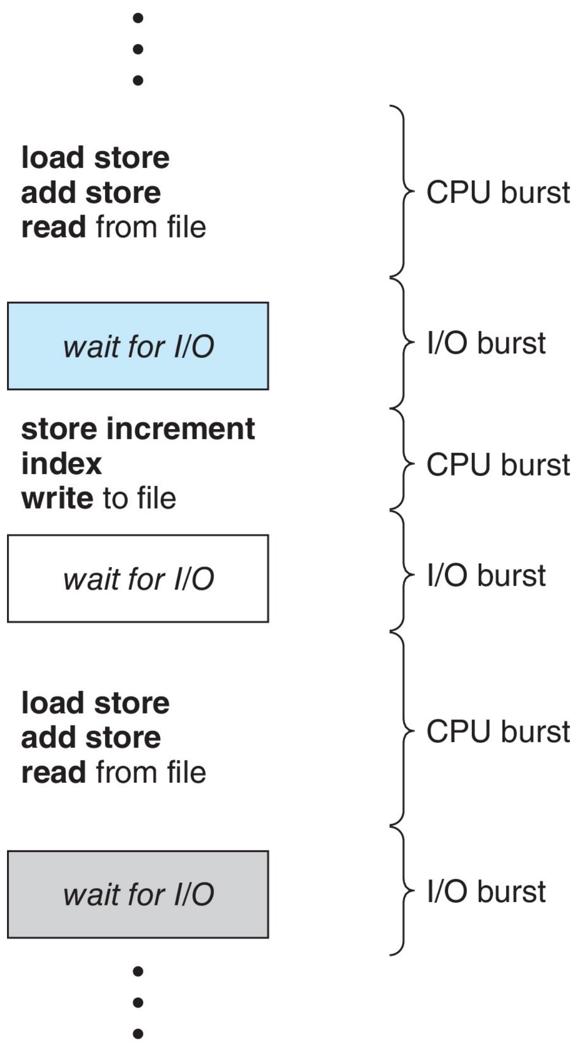
Basic Concepts

- Maximum CPU utilization obtained with multiprogramming
- CPU–I/O Burst Cycle
  - Process execution consists of a **cycle** of CPU execution and I/O wait



Basic Concepts

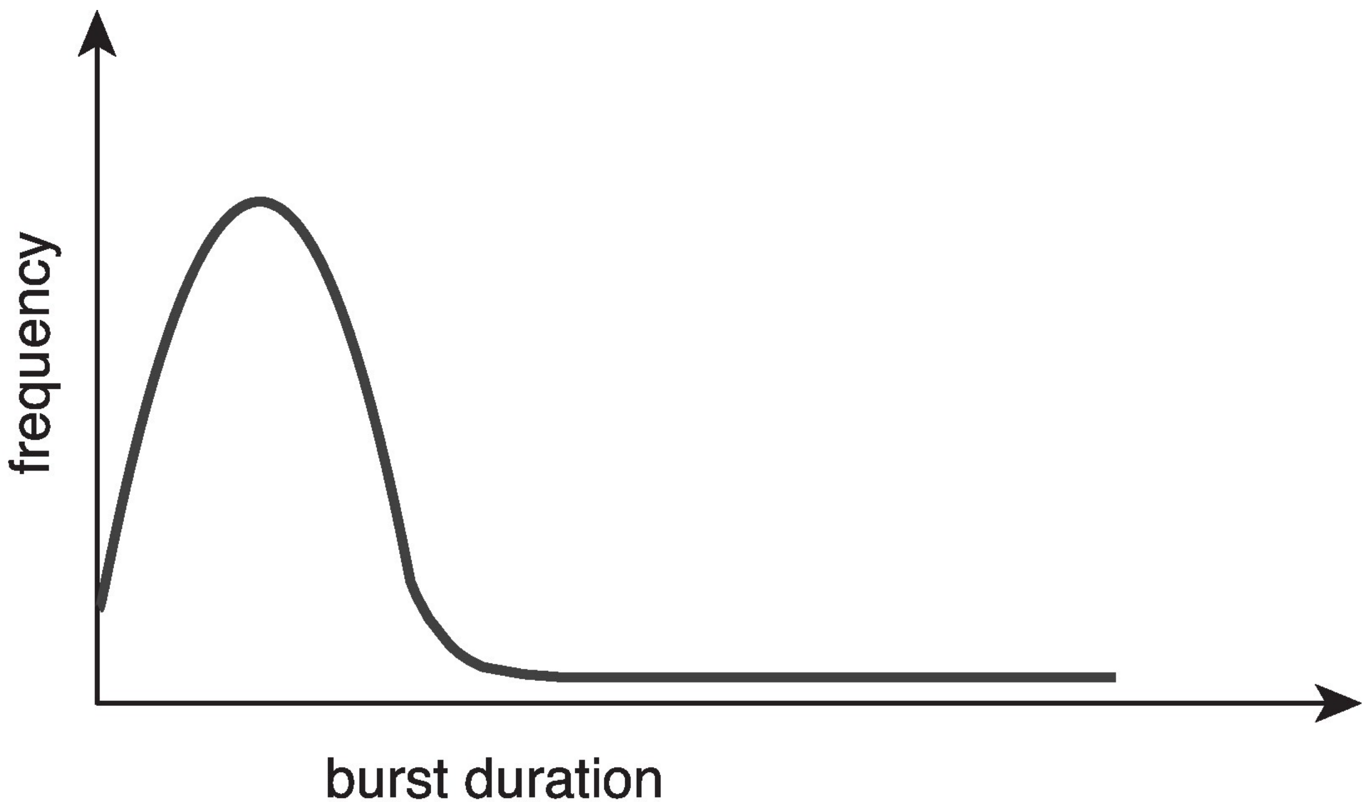
- **CPU burst** followed by **I/O burst**
- CPU burst distribution is of main concern



Histogram of CPU-burst Times

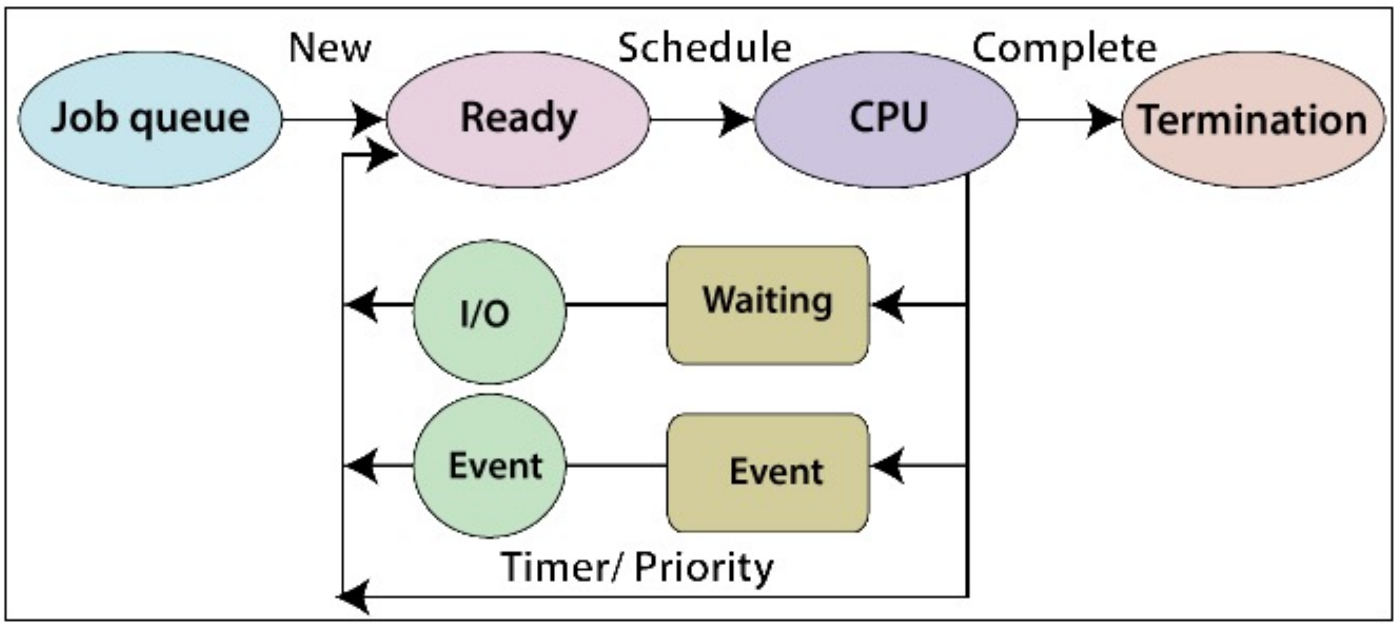
Large number of short bursts

Small number of longer bursts



CPU Scheduler

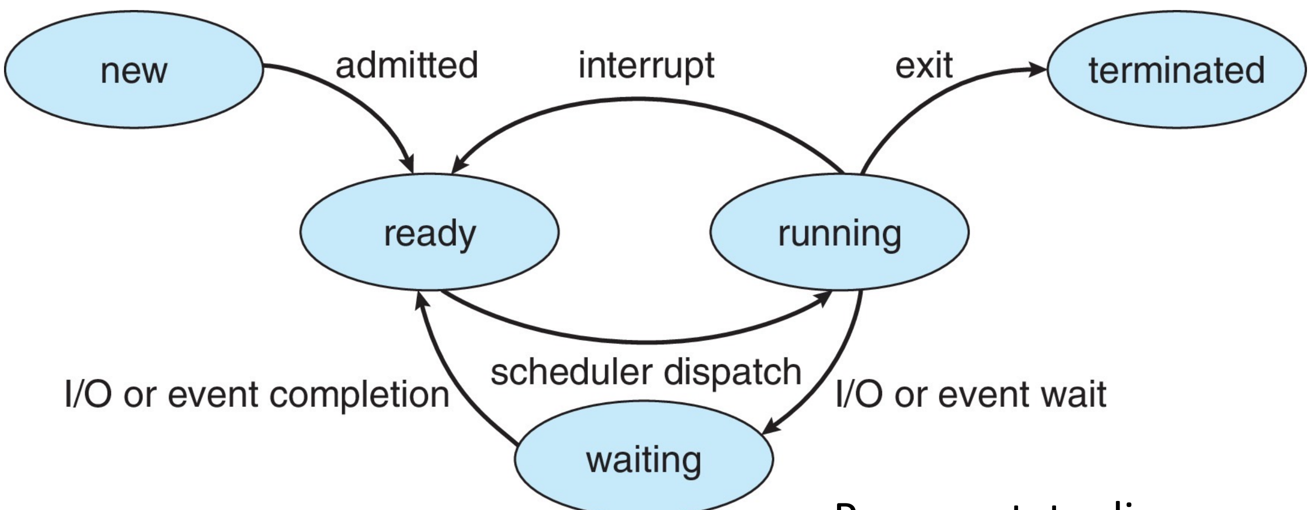
- The CPU scheduler selects from among the processes in ready queue and allocates a CPU core to one of them.
  - Queue may be ordered in various ways.



https://www.tutorialandexample.com/process-schedulers-and-process-queue/

CPU Scheduler (cont.)

- CPU scheduling decisions may take place 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**



Process state diagram

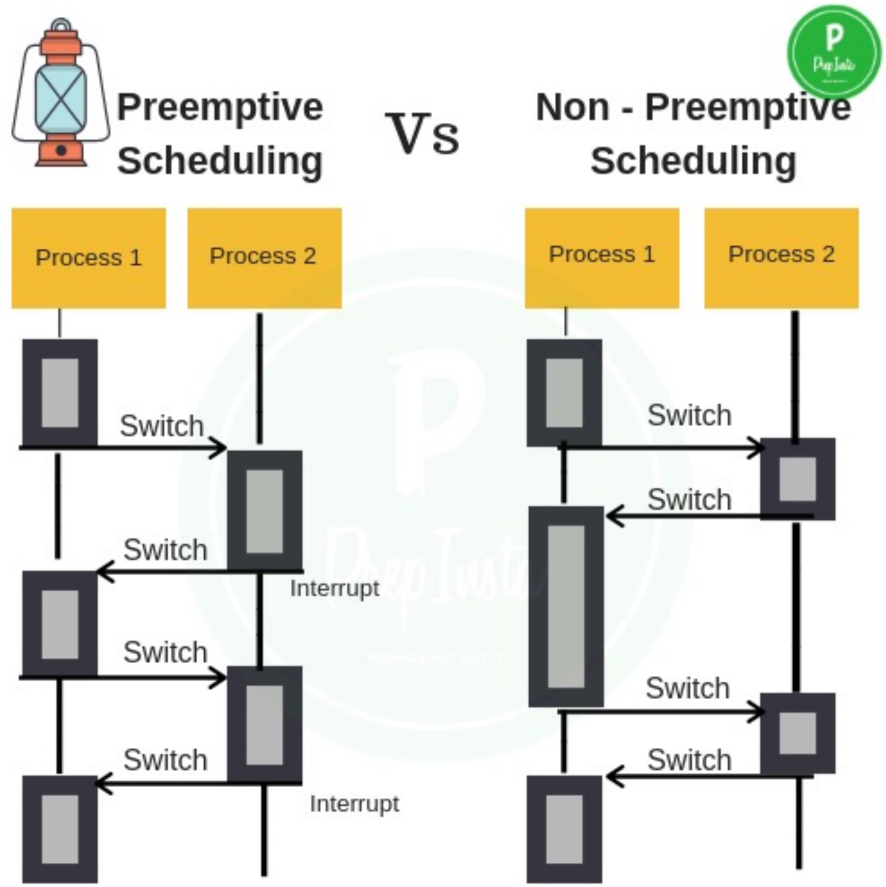
CPU Scheduler (cont.)

- Four possible scheduling situations
  1. Switches from running to waiting state
  2. Switches from running to ready state
  3. Switches from waiting to ready
  4. Terminates

- For situations 1 and 4, there **is no choice in terms of scheduling**.
  - A new process must be selected for execution.
  - If at least one process exists in the ready queue
- For situations 2 and 3, however, there is a choice.

Preemptive and Nonpreemptive Scheduling

- **Non-preemptive (or cooperative)**
  - Circumstances 1 and 4
- **Preemptive**
  - Circumstances 2 and 3



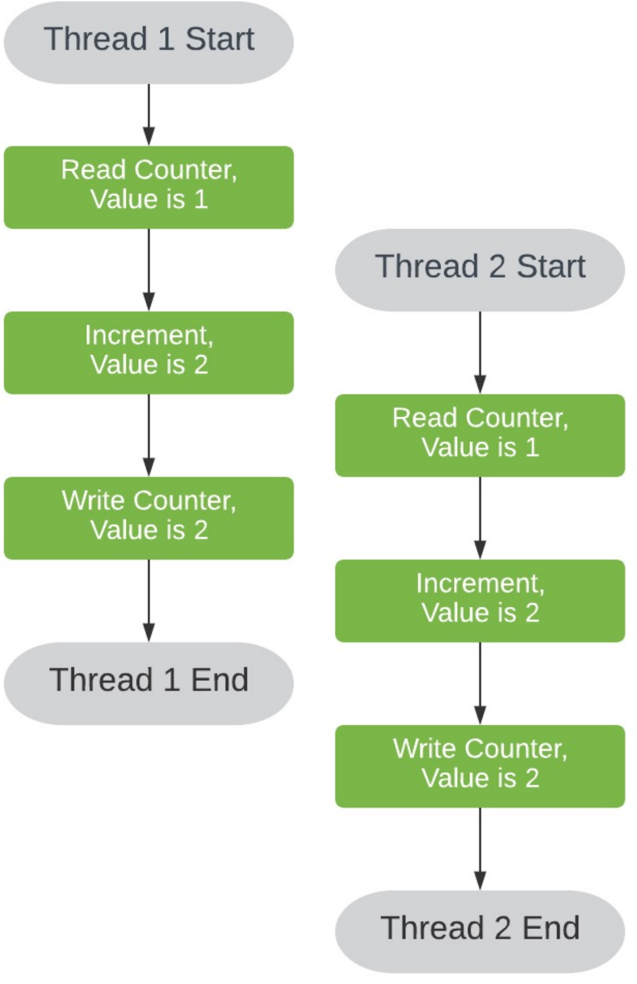
Preemptive and Non-preemptive Scheduling (cont.)

- **Non-preemptive scheduling**
  - Once the CPU has been allocated to a process, the process keeps the CPU until it releases it either by terminating or by switching to the waiting state.

- Virtually **all modern operating systems use preemptive scheduling algorithms**.
  - Including Windows, MacOS, Linux, and UNIX

Preemptive Scheduling and Race Conditions

- **Preemptive scheduling** can result in **race conditions** when ***data are shared*** among several processes.



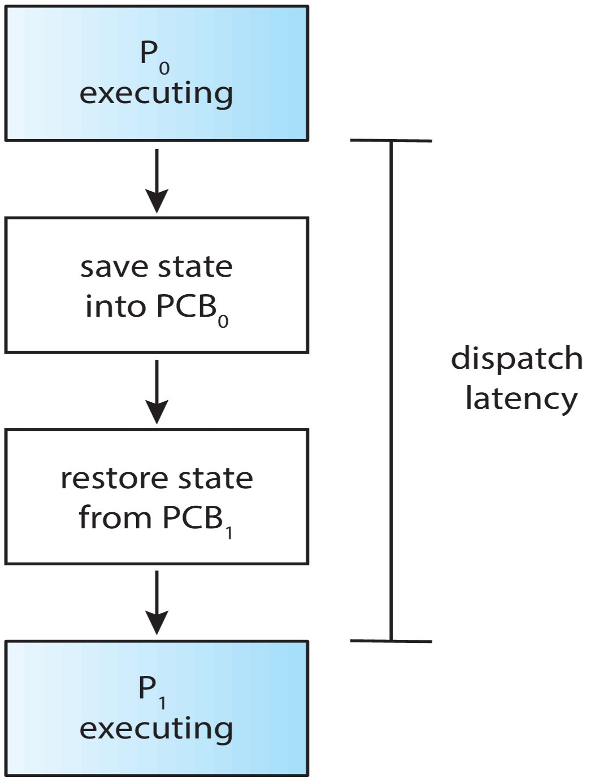
Preemptive Scheduling and Race Conditions (cont.)

- Consider the case of two processes that share data.
  - While one process is **updating the data**, it is preempted so that the second process can run.
  - The second process then tries to read the data, which are in an **inconsistent state**.

- This issue will be explored in detail in Chapter 6.

Dispatcher

- Gives control of the CPU to the process selected by the CPU scheduler
- This involves:
  - Switching context
  - Switching to user mode
  - Jumping to the proper location in the user program to restart that program.
- **Dispatch latency**
  - Time it takes for the dispatcher to stop one process and start another running.



Scheduling Criteria

CPU UTILIZATION

THROUGHPUT

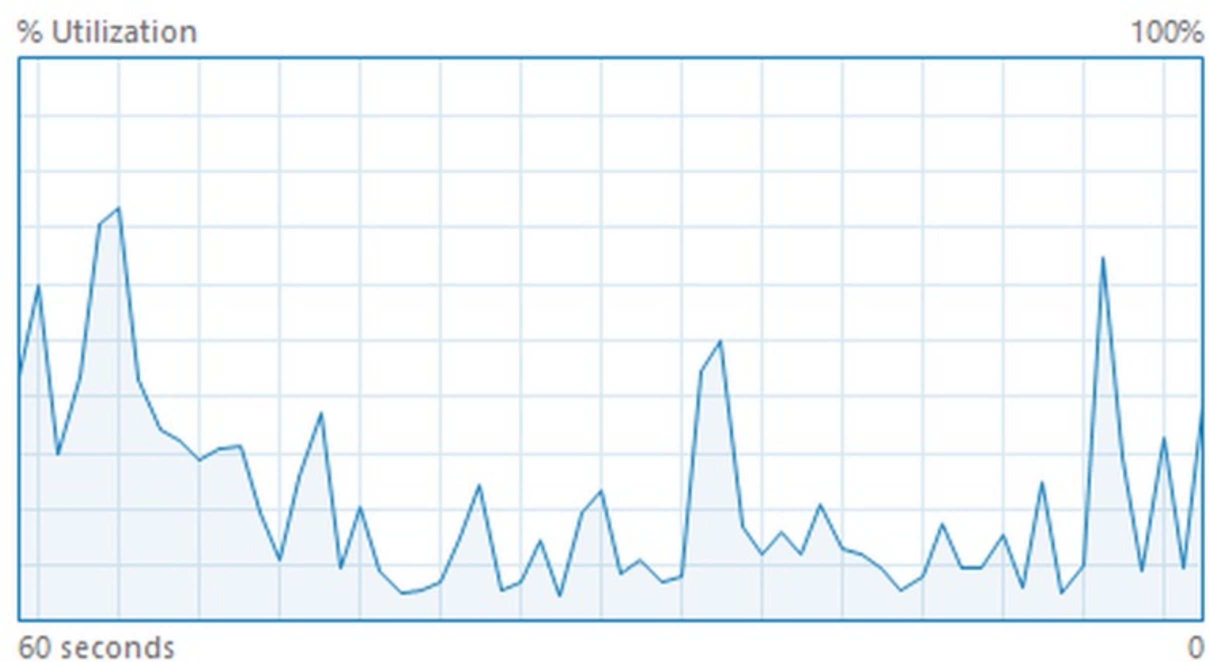
TURNAROUND TIME

WAITING TIME

RESPONSE TIME

CPU utilization

- Keep the CPU as busy as possible.



Throughput

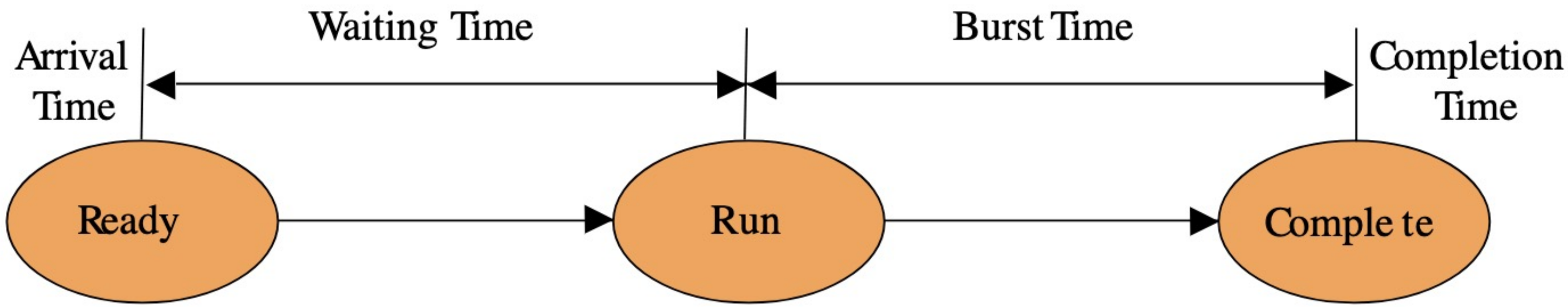
- Number of processes that complete their execution per time unit.

Turnaround time

- Amount of time to execute a particular process.
- Sum of the periods spent waiting, in the ready queue, executing on the CPU, and doing I/O.

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.

Scheduling Algorithm Optimization Criteria

Criteria	Min or Max?
CPU utilization	
Throughput	
Turnaround time	
Waiting time	
Response time	

Scheduling Algorithm Optimization Criteria

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

First-Come, First-Served

SCHEDULING ALGORITHM

First- Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
- 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 and Convoy effect

- **Short process behind long process.**
  - Consider one CPU-bound and many I/O-bound processes.

- **What is the important side-effect?**

FCFS Scheduling and Convoy Effect (Cont.)

- **Short process behind long process.**
  - Consider one CPU-bound and many I/O-bound processes.

- What is the side-effect?
  - Results in **lower CPU and device utilization** than might be possible if the shorter **processes were allowed to go first.**