Amirkabir University of Technology
(Tehran Polytechnic)

Operating Systems

CPU Scheduling-Part1

Seyyed Ahmad Javadi

sajavadi@aut.ac.ir

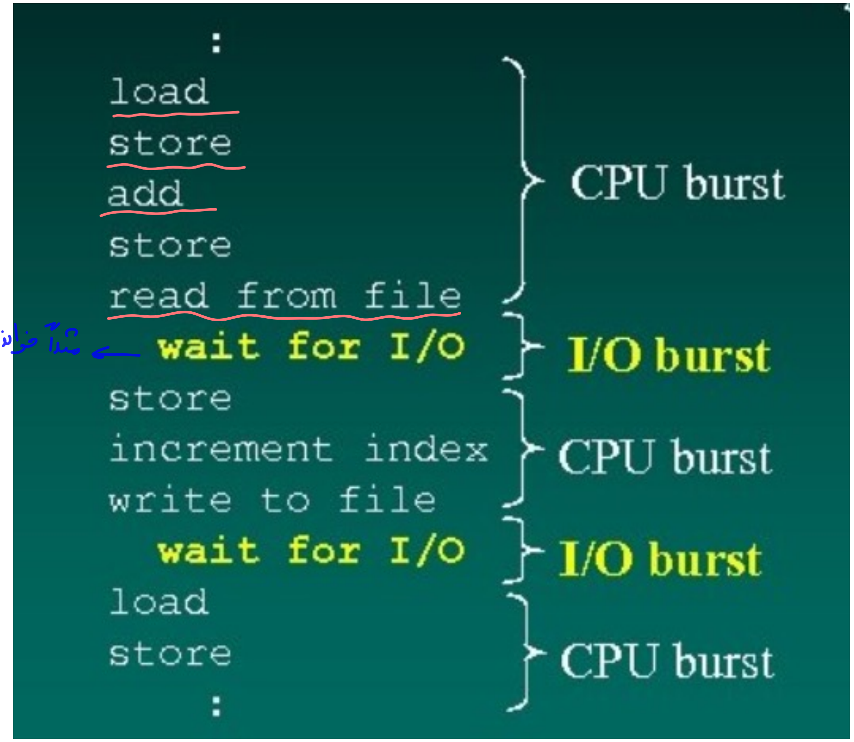
Fall 2021

Basic Concepts

- Maximum CPU utilization obtained with multiprogramming
- CPU–I/O Burst Cycle

  disk, network

  - Process execution consists of a **cycle** of CPU execution and I/O wait



```
    :
load
store
add
store
read from file
  wait for I/O
store
increment index
write to file
  wait for I/O
load
store
    :
```

مثلاً خواندن از keyboard

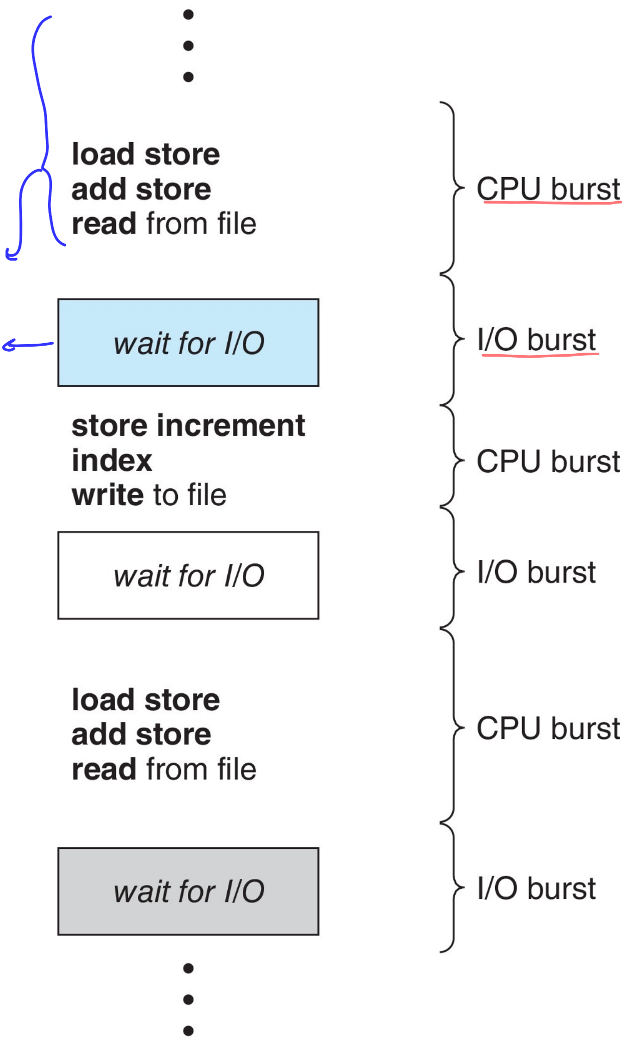
Basic Concepts

زمان‌بندی را می‌خواهیم به صورتی انجام دهیم که بیشترین بهره‌وری را داشته باشیم!

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

مثلاً 1ms ⇐ مدت زمان رو اندازه می‌گیریم ← در این بازه هیچ I/O انجام شده است

مثلاً 1s ⇐ I/O اولین ←

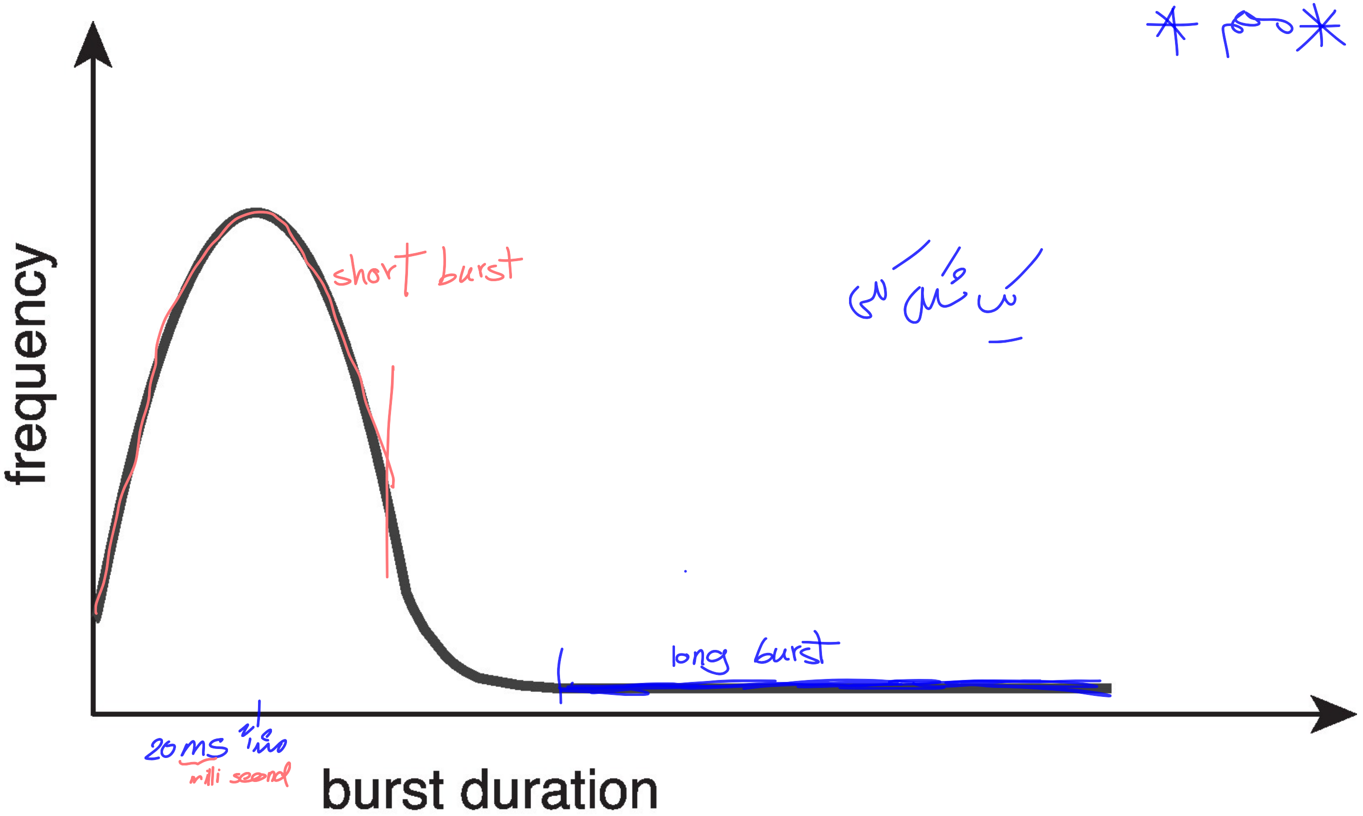


Histogram of CPU-burst Times

Large number of short bursts

Small number of longer bursts

* مهم *



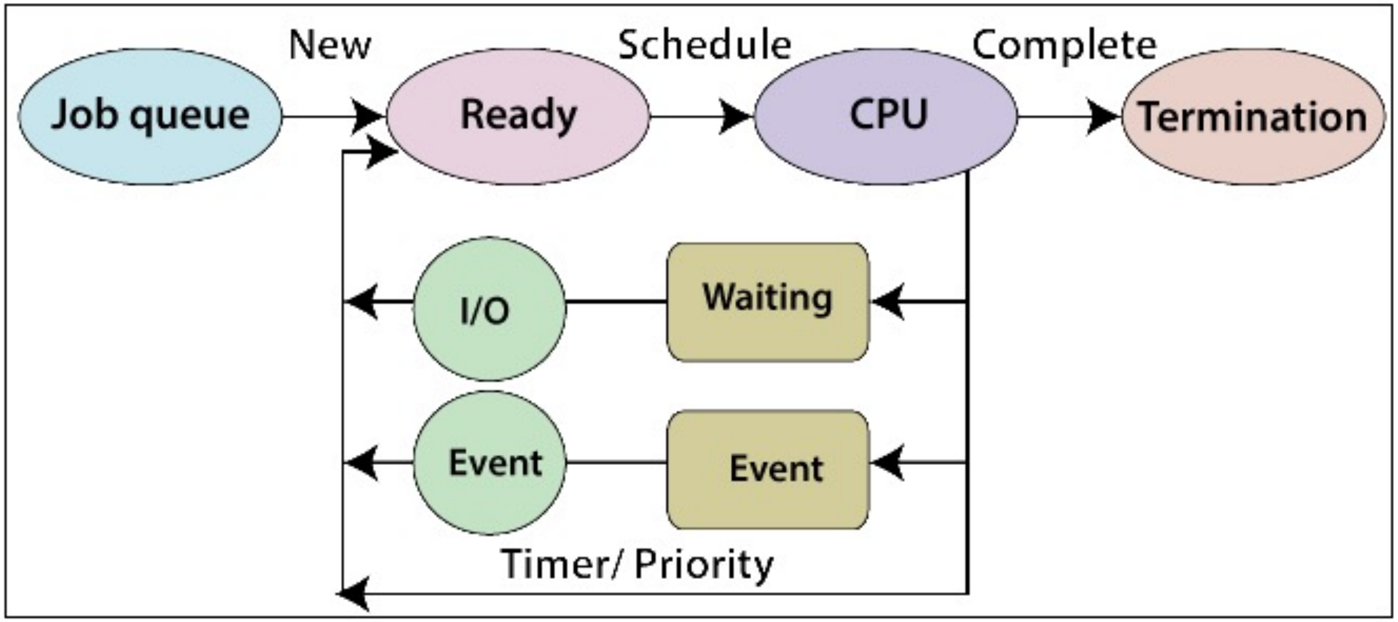
CPU Scheduler

یک ماژول مهم (یک قطعه کد مهم)

- The CPU scheduler selects from among the processes in ready queue and allocates a CPU core to one of them.

الگوریتم های زمان بندی متفاوت

  - 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

     مثلاً I/O اتفاق می‌افته

     اون proc که در running بود یک core دستش بود. حالا که رفته توی waiting، این core بیکاره! برای همین cpu scheduler یک proc جدید رو میاره توی running، و اون core رو اختصاص میده بهش!

     مثلاً time interrupt داده میشه (فرصت اجرای اون برنامه تموم شده)
  2. Switches from **running to ready state**

     دوباره cpu scheduler یک proc دیگه رو میاره توی running
  3. Switches from **waiting to ready**

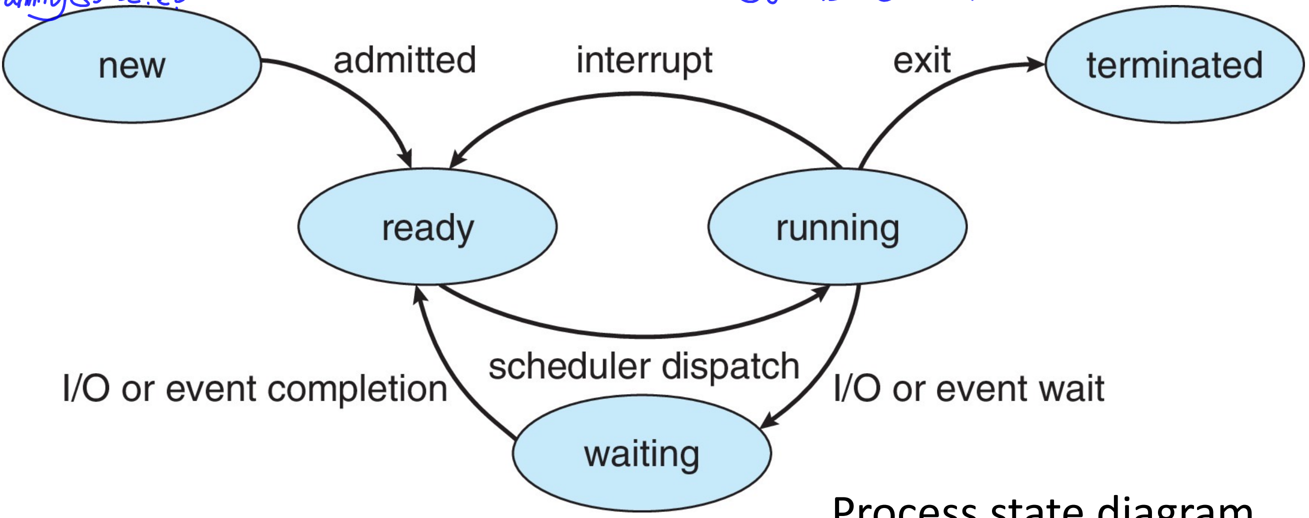
     هر تغییری در پردازه‌های آماده‌ی اجرا، cpu scheduler را مجبور می‌کنه یه دور دیگه اولویت‌ها رو چک کنه و در صورتی که اولویت یک proc بالاتر از proc در حال اجراست، آن را بیاورد داخل running

     ready queue

     p1 در running

     p2 از waiting اومده توی running و اولویتش از p1 بیشتره!
  4. **Terminates**

     یکی proc تموم میشه. یک proc دیگه باید بیاد توی running



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

→ یک core خالی شده! باید
دوباره ازش استفاده کنی تا بهره‌وری
maximum بشه!

- For situations 2 and 3, however, there is a choice.

← اینکه چه زمانی cpu رو
ازش بگیره! مثلاً توی حالت 2
چه زمانی cpu رو از اون پردازه بگیره
و بفرسته توی حالت ready و به دست
یک پردازه‌ی دیگه!

→ اگه زود ازش بگیریم، ممکنه
کار مفیدی هنوز انجام نداده باشه!
(context switch → pure overhead!) ⇐
اگه هم دیر بگیریم، مشکلات دیگه‌ای هست!

← همچنین تو حالت 3
بحث اولویت‌ها هم است!

← مثلاً یه بحث مهم همین زمان
switch کردن با یک task
با اولویت بالاتر است. یک سری taskها
اگه به موقع switch نکنیم بهش، Exterminate!

Preemptive and Nonpreemptive Scheduling

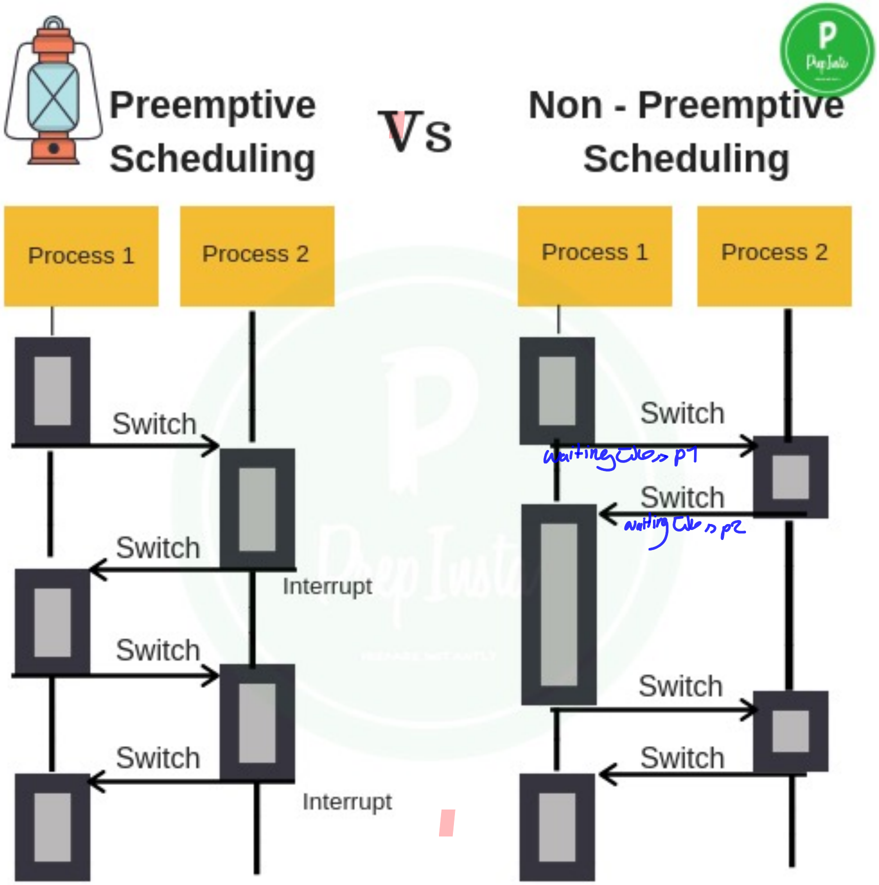
نمی‌تونیم یک پردازه را تا زمانی‌که terminate یا waiting نشده، cpu را ازش بگیریم!

غصب‌شدنی

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

غصب‌شدنی! یعنی cpu را از پردازه بگیریم!

- **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.

sheduler براساس یک الگوریتم خاص هرموقع که خواست می‌تونه cpu رو بگیره از یک proc

- Virtually **all modern operating systems use preemptive scheduling algorithms**. →

چون در حالت Non-preemptive اصلا
responsive نیست! ممکنه یک برنامه خیلی طول بکشه
و درنتیجه از cpu استفاده‌ی بهینه نشه!

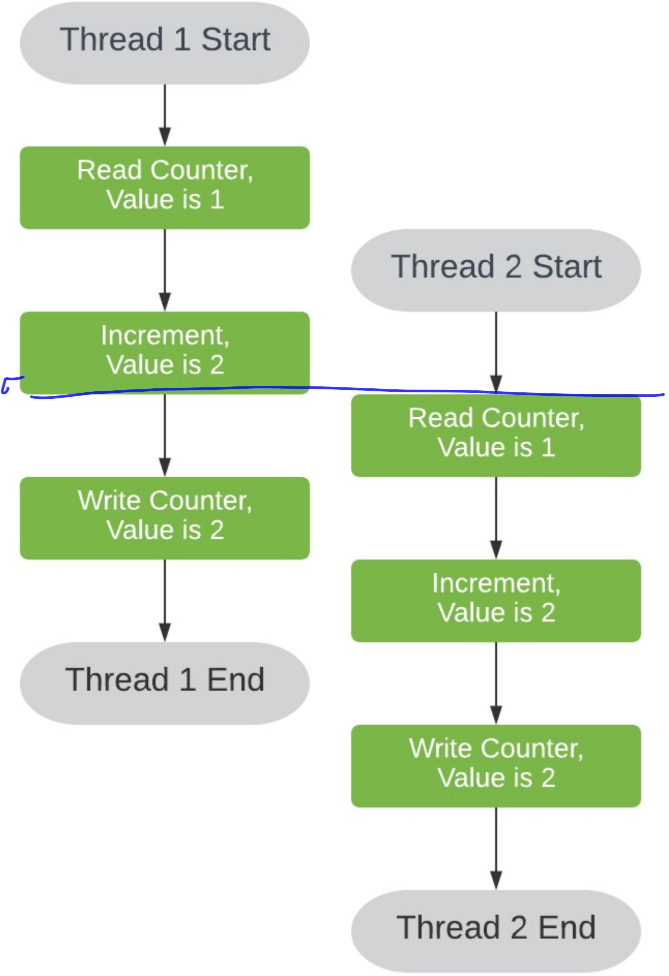
  - Including Windows, MacOS, Linux, and UNIX

Preemptive Scheduling and Race Conditions

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

ایجاد

احتمال race condition در Non-preemptive کمتر است!



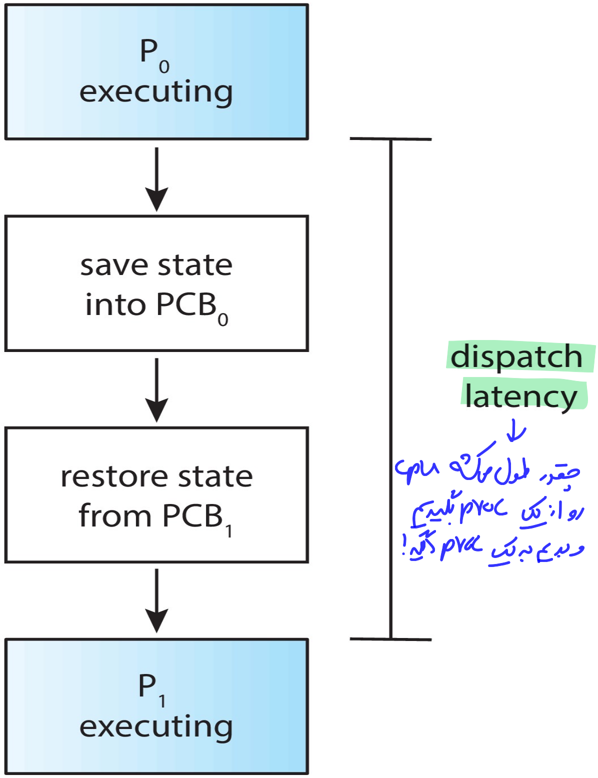
تایمش تموم میشه

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 (from ready queue)
- 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.



چقدر طول میکشه ما cpu رو از یک proc بگیریم و بدیم به یک proc دیگه!

Scheduling Criteria

CPU UTILIZATION

THROUGHPUT

TURNAROUND TIME

WAITING TIME

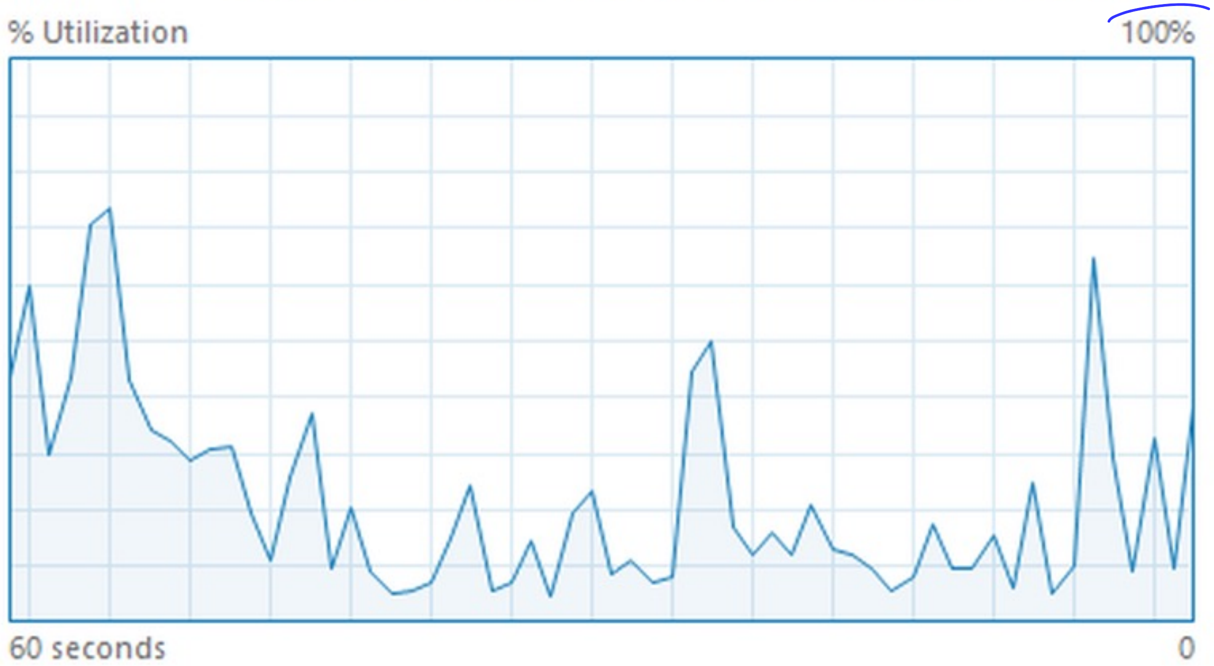
RESPONSE TIME

CPU utilization

- Keep the CPU as busy as possible.

ایده‌آل اینه که ۱۰۰٪ باشه ← ۱۰۰٪ یعنی بیشترین استفاده از cpu

ولی همین مشکلاتی هم داره !



Throughput

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

Turnaround time

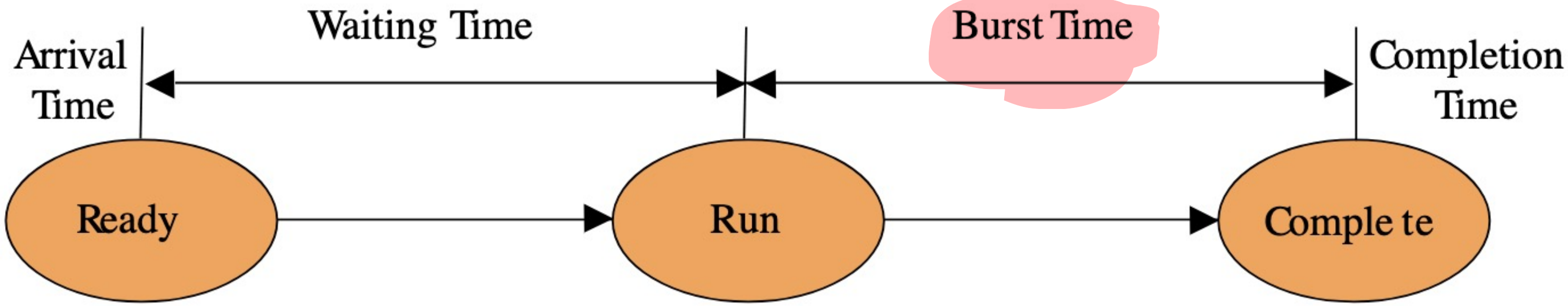
- Amount of time to execute a particular process.
- Sum of the periods spent waiting, in the ready queue, executing on the CPU, and doing I/O.

از زمانی که یک proc پاشو میذاره تو system
تا زمانی که خارج میشه

Waiting time

- Amount of time a process has been waiting in the **ready queue**.

. جمع تمام زمان‌هایی که یک proc در ready queue سپری می‌کند!



Response time

- Amount of time it takes from when a request was submitted until the first response is produced.

از لحظه‌ای که درخواست cpu رو مطرح می‌کنه
تا زمانی که اجرا شه

اولین درخواست!

تفاوت با waiting time:

جمع تمام زمان هایی که منتظر
مونده توی ready quae

Scheduling Algorithm Optimization Criteria

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

Scheduling Algorithm Optimization Criteria

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

این 5 تا باهم توی یک مسیر نیستند!
ولی باید سعی کنیم یک تعادلی بین این 5 تا
برقرار کنیم.

مثلا: اگر cpu رو بدیم به یک برنامه‌ی cpu intensive
از نظر cpu utilization ، ۱۰۰٪ داره استفاده می‌کنه
اما برای بقیه response time و waiting time شون خیلی زیاد میشه!

First-Come, First-Served

SCHEDULING ALGORITHM

First- Come, First-Served (FCFS) Scheduling

first come in ready queue

این روش non-preemitive است !

Process	Burst Time
P_1	24 time unit
P_2	3
P_3	3

اولویت : p1 > p2 > p3

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

اینکه هر پردازه از کی تا کی cpu دستشه

- 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.** → چون Non-preemptive هست، نمی‌تونم متوقفش کنم!
  - Consider one CPU-bound and many I/O-bound processes.


یعنی اینا به خاطر
تراکتوره گیر افتادن!

- **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.**

پس اگر یک برنامه برای یک مدت ، ۱۰۰٪ cpu دستش باشه لزوماً خوب نیست!

shortest job first ← راه حل