Operating Systems
CS220

Lecture 9

CPU Scheduling

17th May 2021

By: Dr. Rana Asif Rehman

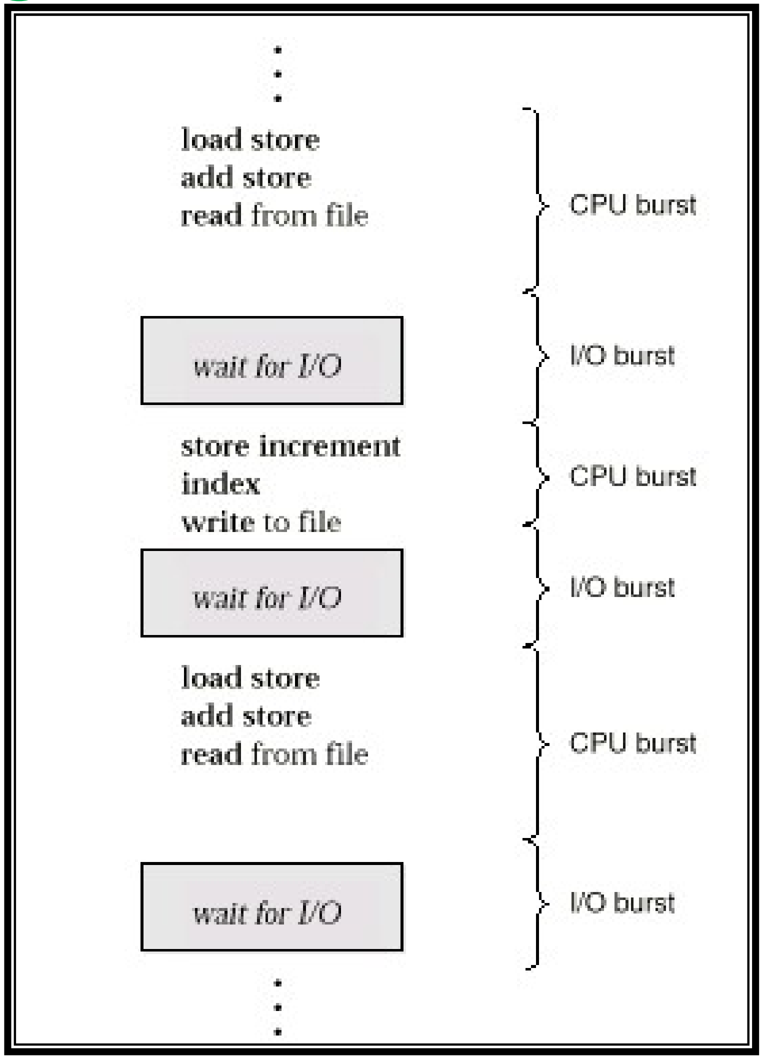
What's in today's lecture

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Multiple-Processor Scheduling
- Real-Time Scheduling
- Algorithm Evaluation

Basic Concepts

- In multiprogramming systems, where there is more than one process runnable (ready), the OS must decide which one to run next
- We already talked about the dispatcher, whose job is to allow the next process to run
- It's intimately associated with a scheduler whose job is to decide (using a scheduling algorithm) which process to have the dispatcher dispatch
- We will focus on the scheduling, understand the problems associated with it, and look at some scheduling algorithms

Alternating Sequence of CPU And I/O Bursts



Nature of Process

- Not all processes have an even mix of CPU and I/O usage
- A number crunching program may do a lot of computation and minimal I/O
  - This is an example of a CPU-BOUND process
- A data processing job may do little computation and a lot of I/O
  - This is an example of an I/O-BOUND process

Types of Schedulers

- We have to decide which ready process to execute next – this is **Short-Term Scheduling or CPU Scheduler** and occurs frequently
- Short-term scheduling only deals with jobs that are currently resident in memory
- Jobs are constantly arriving, and **Long-Term Scheduling or Job Scheduler** must decide which to let into memory and in what order
- **Medium-Term Scheduling** involves suspending or resuming processes by swapping (rolling) them out of or into memory (from disk)

Scheduling

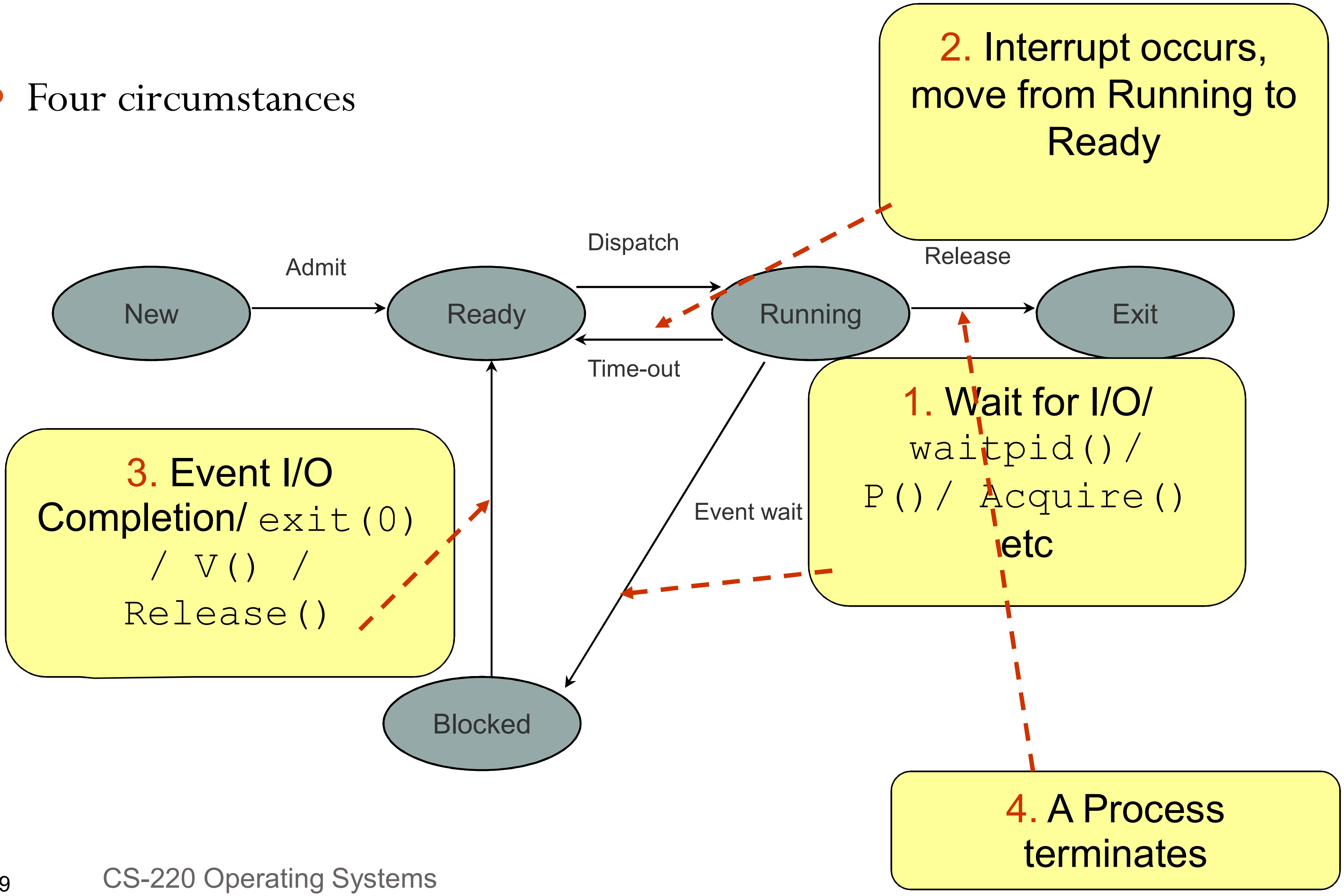
- Short term scheduler (CPU Scheduler)
  - Whenever the CPU becomes idle, a process must be selected for execution
  - The Process is selected from the Ready queue
- Ready queue is not necessarily a FIFO queue
- It can be
  - Priority based
  - A Tree
  - Unordered linked list etc

Short term Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
- 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
- Scheduling under 1 and 4 is ***nonpreemptive***
- All other scheduling is ***preemptive***

When to select a new process to Run

- Four circumstances

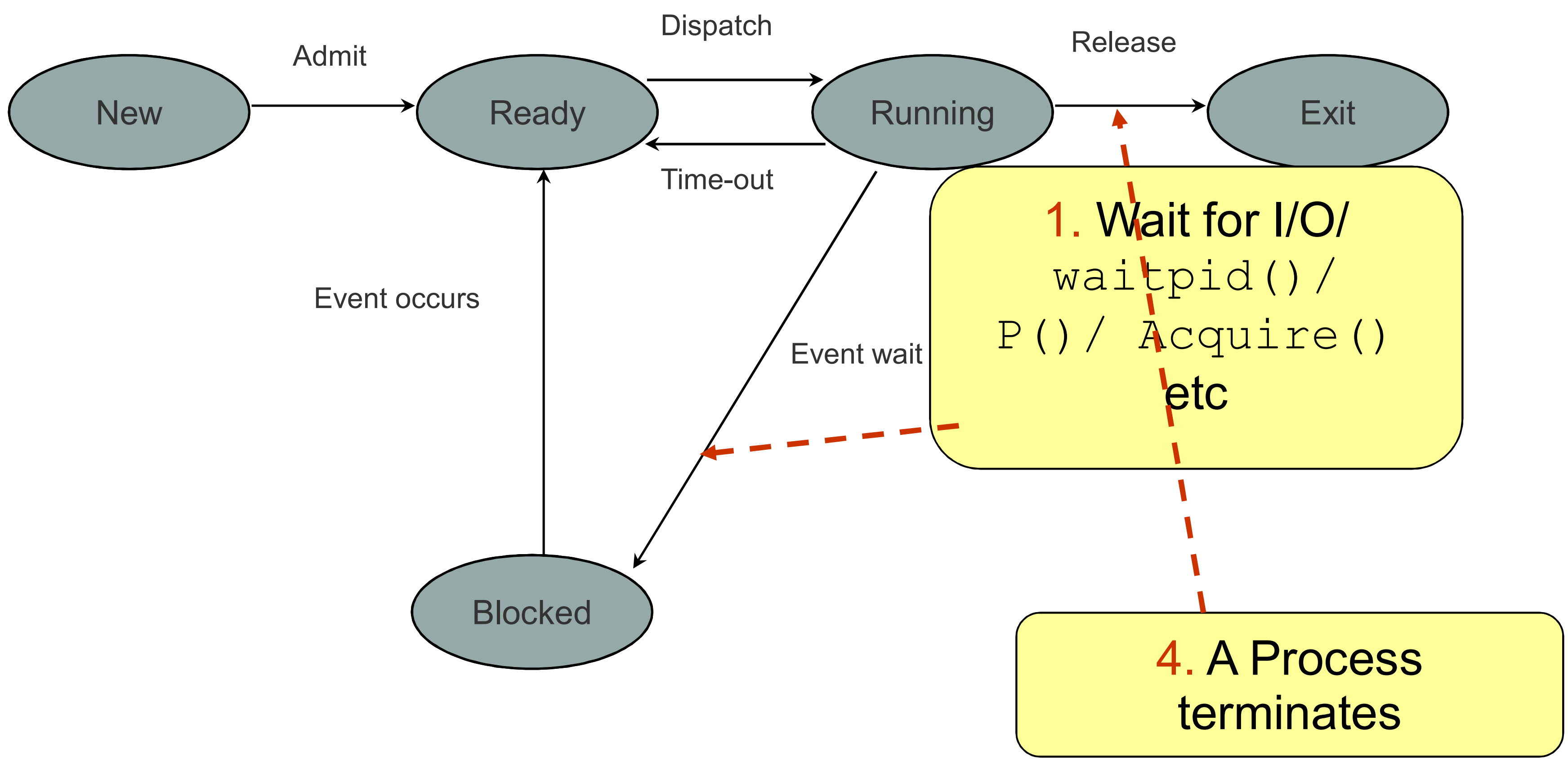


Non Preemptive Scheduling

- Once the CPU has been allocated to a process
- The process keeps it until
  - It Terminates
  - Or has to wait for:
    - I/O
    - Mutex
    - Child process
    - Semaphore
    - Conditional Variables etc
- There is no way, to get the CPU back, FORCEFULLY

Non Preemptive Scheduling

- Only the case 1 and 4
- Must select a new process, if any, from the Ready Queue

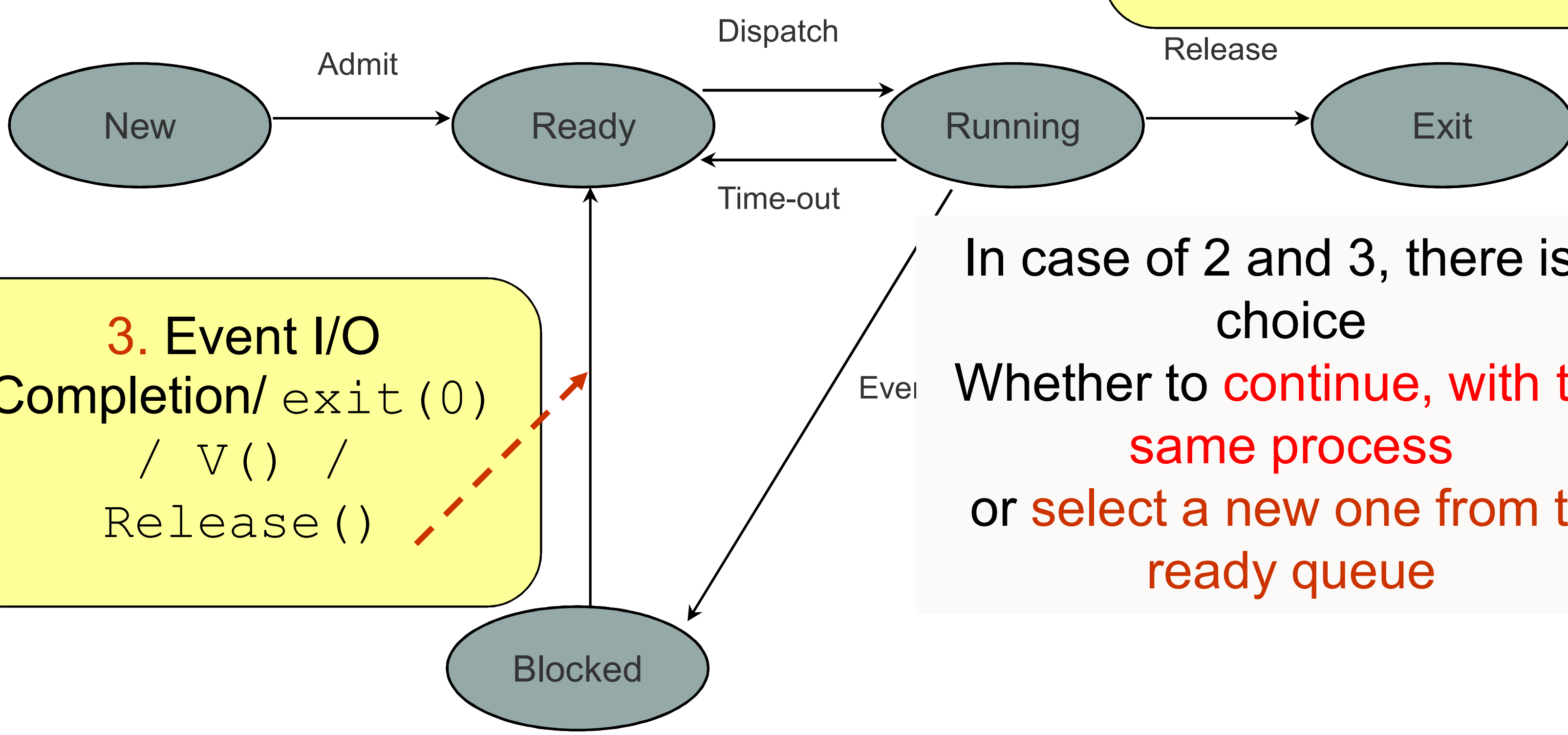


Preemptive Scheduling

- Possible to get the CPU back from a process, even if the process has not completed its execution.

Preemptive Scheduling

2. Interrupt occurs, move from Running to Ready



3. Event I/O Completion/ `exit(0)` / `V()` / `Release()`

In case of 2 and 3, there is a choice
Whether to continue, with the same process
or select a new one from the ready queue

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term 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
  - Keep the CPU as busy as is possible
  - May range from 0% to 100%
- Throughput
  - Number of processes completed per unit time
  - E.g. long processes
    - 1 process / hr
  - Short processes
    - 10 processes / hr

Scheduling Criteria

- Turnaround Time
  - How long it take to execute a Process

```
Turnaround = Completion_Time
- Submission_Time
```

$$\texttt{Turnaround} = \texttt{Wait\_Time}_{\texttt{GetIntoMemory}} + \texttt{Wait\_Time}_{\texttt{ReadyQueue}} + \texttt{Wait\_Time}_{\texttt{BlockQueue}} + \texttt{CPU\_Execution\_Time}$$

Scheduling Criteria

- Scheduling Algorithm does not effect the waiting time in Block Queue
- It only effect the Waiting Time in the Ready Queue
- Waiting Time
  - Sum of the periods spent waiting in the Ready Queue

W.T = Finish Time – Burst Time – Arrival Time

Scheduling Criteria

- Turnaround Time is not a good criteria for Interactive Systems
- A process may
  - Produce "Some" output
  - Computes new results, while previous results are output to the user
- Response Time
- **`Response_Time = First_Response_Start_Time`**
  - **`Submission_Time`**

Additional Scheduling Criteria

- There are also other elements to consider:
  - ***Priority/Importance*** of work – hopefully more important work can be done first
  - **Fairness** – hopefully eventually everybody is served
    - Implement policies to increase priority as we wait longer… (this is known as "priority aging")
  - **Deadlines** – some processes may have hard or soft deadlines that must be met
  - Consistency and/or predictability may be a factor as well, especially in interactive systems

Optimization Criteria

- We would like to Maximize
  - CPU Utilization
  - Throughput
- And Minimize
  - Turnaround Time
  - Waiting Time
  - Response Time

Scheduling Algorithms

- First come, First serve (FCFS)
- Shortest Job First (SJF)
- Priority Scheduling
- Round-Robin Scheduling
- Multi-level Queue Scheduling
- Multi-level Feed back queue Scheduling

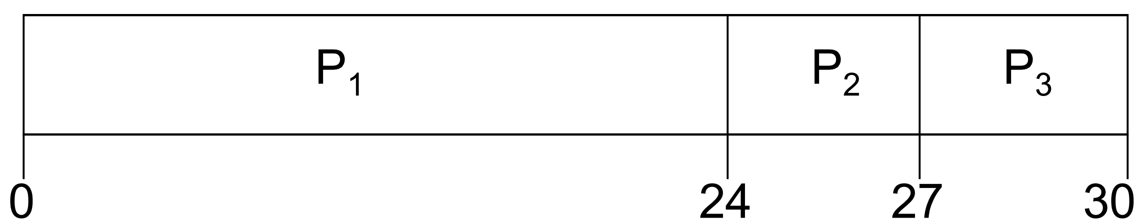
1. First-Come, First-Served (FCFS) Scheduling

- Simplest scheduling algorithm:
  - Run jobs in order that they arrive
- Uni-programming:
  - Run until done
- Multi-programming:
  - Run until done or Blocks on I/O
- Nonpreemptive
  - A Process keeps CPU until done or I/O
- Advantage:
  - Simplicity

FCFS Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

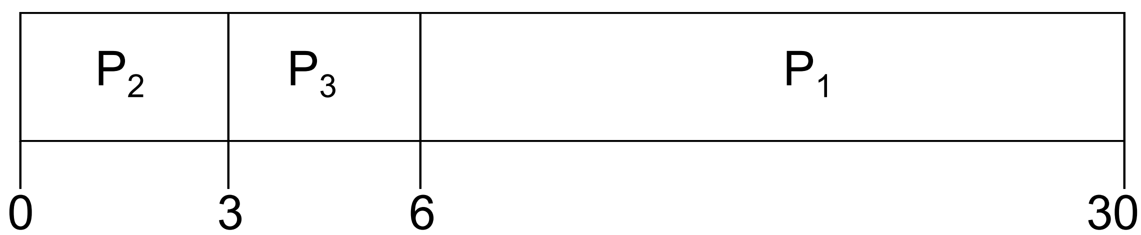
- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule is:



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

FCFS Scheduling (Cont.)

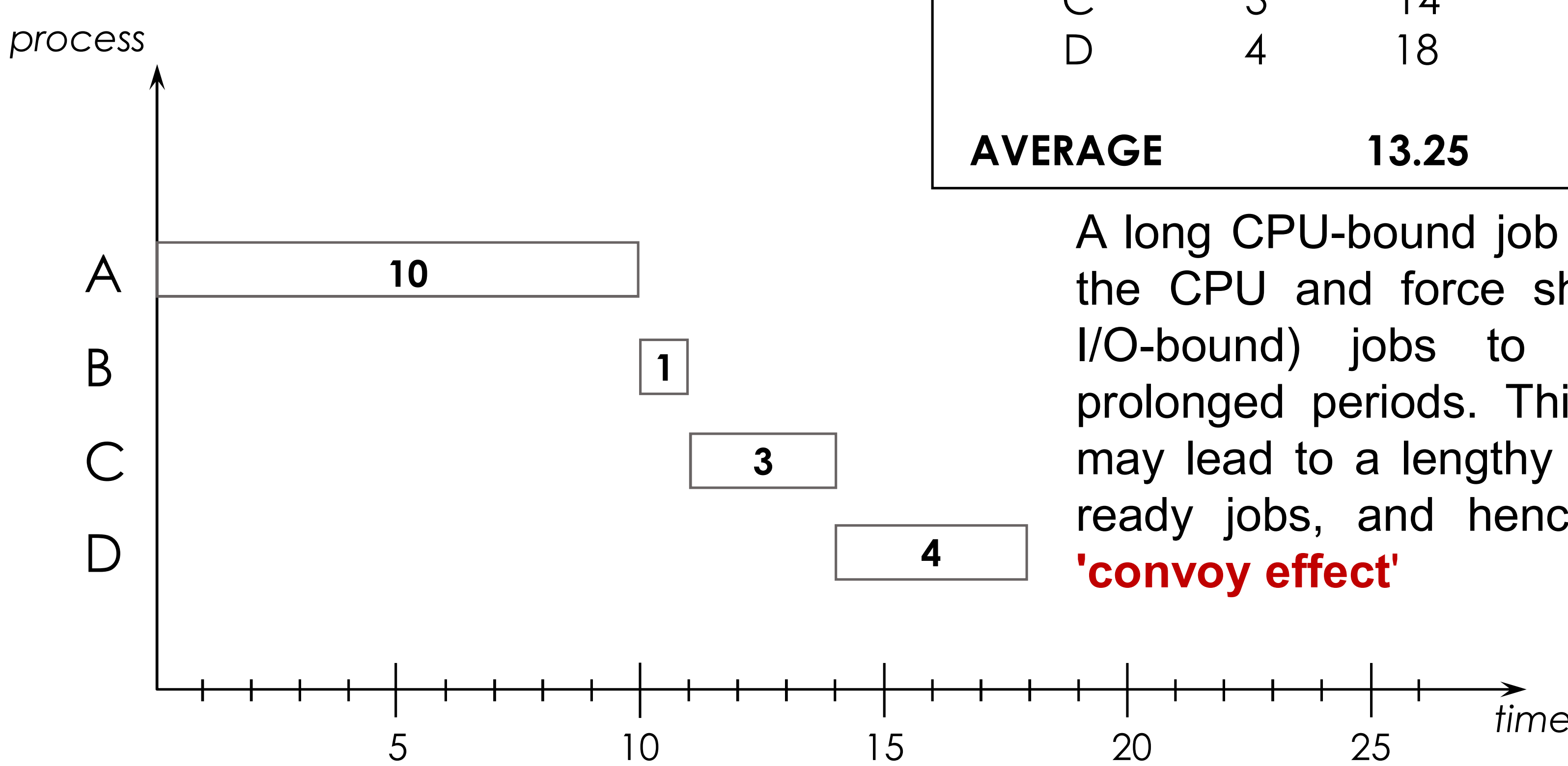
- Suppose that the processes arrive in the order P_2 , P_3 , P_1
- The Gantt chart for the schedule is:



- 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

Convoy Effect

process	service time t_s	turnaround time t_t	waiting time t_w
A	10	10	0
B	1	11	10
C	3	14	11
D	4	18	14
AVERAGE		**13.25**	**8.75**



A long CPU-bound job may hog the CPU and force shorter (or I/O-bound) jobs to wait for prolonged periods. This in turn may lead to a lengthy queue of ready jobs, and hence to the **'convoy effect'**

FCFS Scheduling (Cont.)

- FCFS is:
  - Non-preemptive
  - Ready queue is a FIFO queue
  - Jobs arriving are placed at the end of ready queue
  - First job in ready queue runs to completion of CPU burst
- **Advantages:** simple, low overhead
- **Disadvantages:** long waiting time, inappropriate for interactive systems, large fluctuations in average turnaround time are possible

References

- Operating System Concepts (Silberschatz, 9^{th} edition) Chapter 5