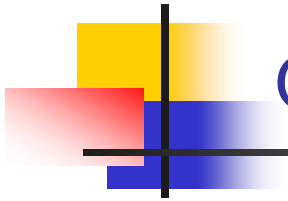
Operating Systems
(Process scheduling)

Chapter 5

These lecture materials are modified from the source lecture notes written by A. Silberschatz, P. Galvin and G. Gagne.

Spring, 2020



Outline

- Examples
- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Multiple-Processor Scheduling
- Thread Scheduling
- Linux Scheduling
- Algorithm Evaluation

Examples

EBS channel

Animation channel

Sports channel

News channel

Movie channel

But, there is only one TV.

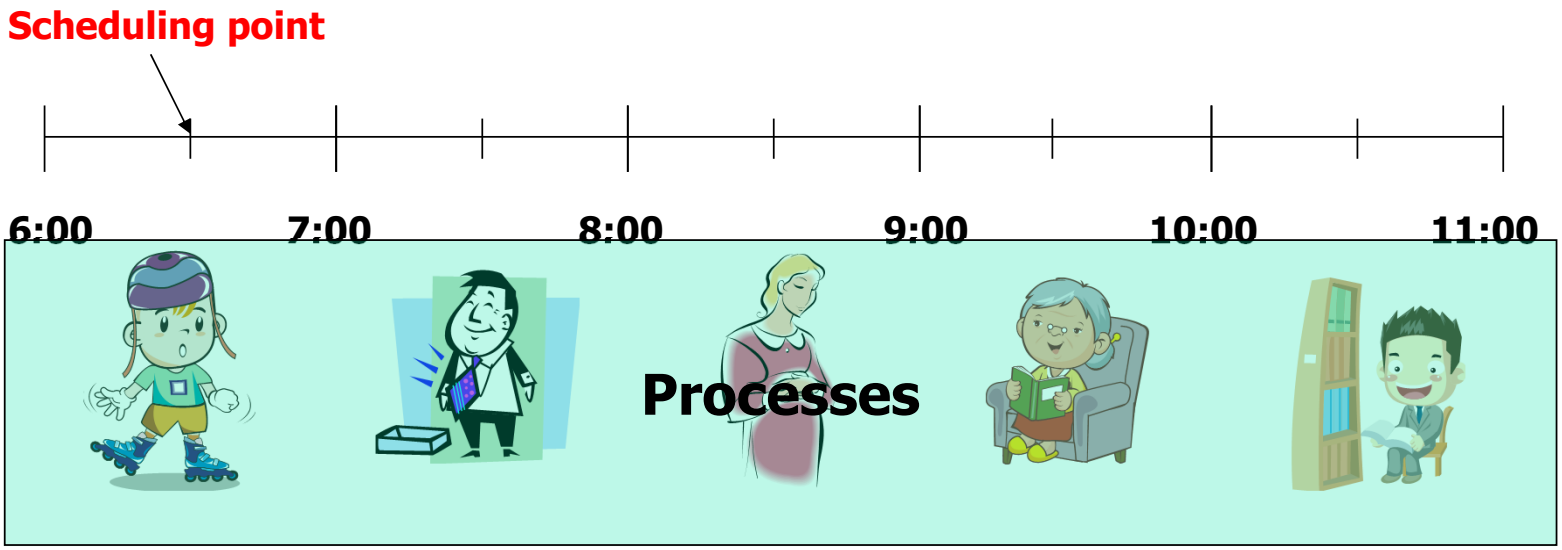
So, they are fighting for TV channel.

How to allocate the TV to each member;

TV Scheduling

How to allocate the CPU to each process;

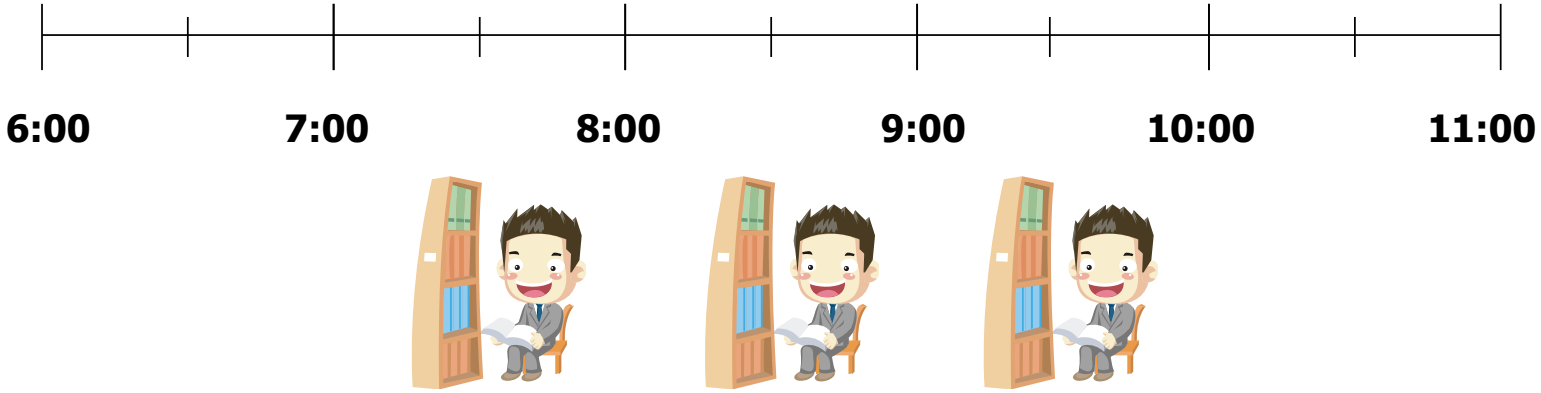
CPU Scheduling



What's the goal?

Make the son get good grades in his studies

Scheduling criteria

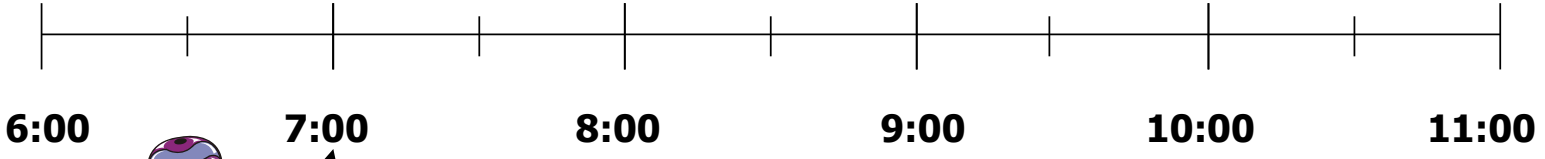


Give him the TV to watch the EBS program

Give the E process the highest priority

Scheduling algorithm

The highest priority

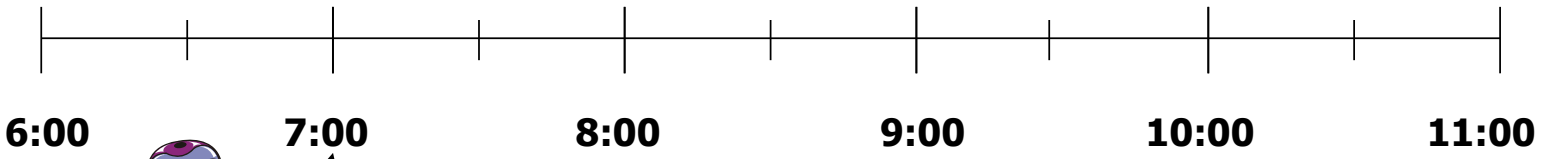


I hope to watch this animation because it ends at 7:30.

Go to your room.

Preemptive scheduling

The highest priority



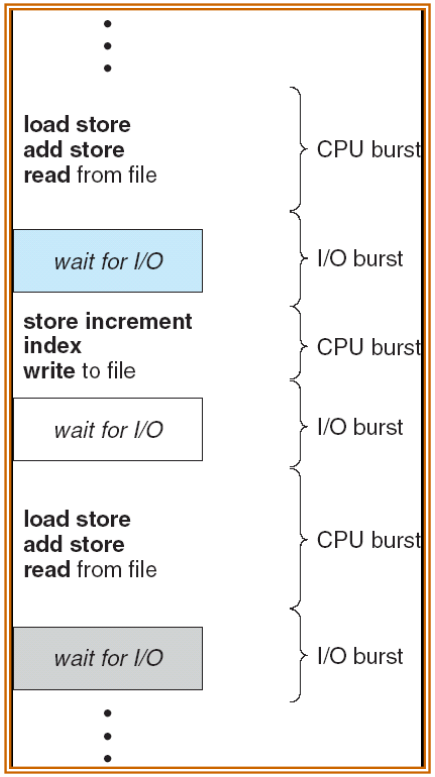
I hope to watch this animation because it ends at 7:30.

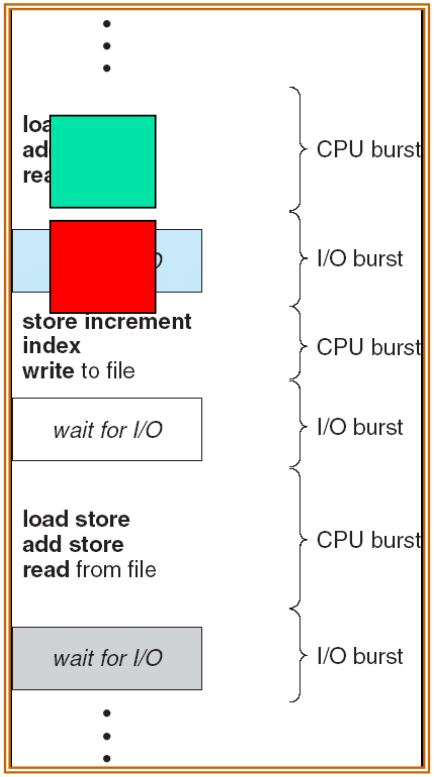
OK, I will wait until 7:30.

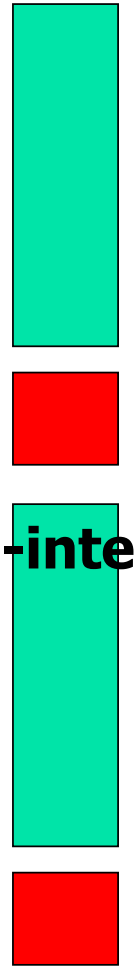
Nonpreemptive scheduling

Basic concepts

- Property of processes
  - Alternating Sequence of CPU And I/O Bursts

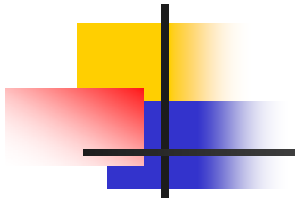




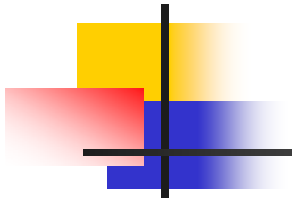


CPU-intensive

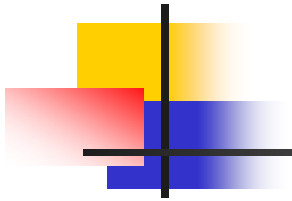
IO-intensive



- General rule of scheduling
  - Give higher priorities to I/O bound processes than CPU-bound processes



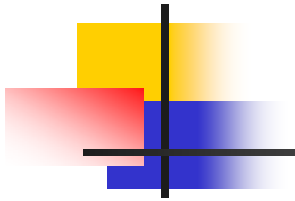
- CPU scheduler
  - Selects from the processes in memory that are ready to execute, and allocates the CPU to one of them
  - CPU scheduling decisions may take place when a process:
    - 1. Switches from running to waiting state
    - 2. Switches from running to ready state
    - 3. Switches from waiting to ready
    - 4. Terminates (from running to termination)



- When does scheduler make decisions?

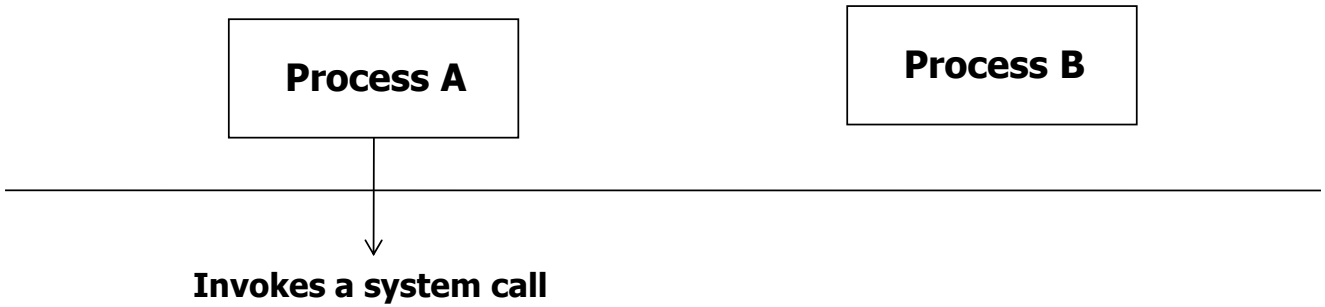
- Nonpreemptive or cooperative
  - process runs until it voluntarily relinquishes CPU:
    - process blocks on an event (e.g., I/O or synchronization)
    - process terminates

- Preemptive
  - All of the above, plus:
    - Event completes: process moves from blocked to ready
    - Timer interrupts
  - Considerations
    - Shared data access



- Preemption in the kernel
  - System call or
  - Interrupt

- System call

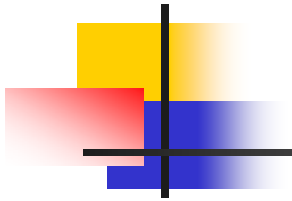


A system call is executed on behalf of process A and Process A is preempted by process B

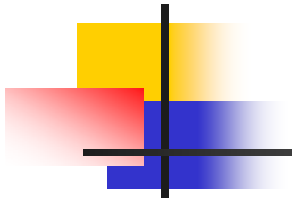
Most old versions of UNIX wait a system call to complete before context switch.

: Non-preemptive kernel

But the non-preemptive kernel is a poor one for supporting real-time computing.



- HW interrupt handler
  - The sections of code affected by interrupts must be guarded from simultaneous use.
- For example,
  - From Linux 2.6
    - The kernel can be preempted when it is executing an system call
    - But, the kernel cannot be preempted when it is executing an interrupt routine.



- Dispatcher
  - Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
    - 1. switching context
    - 2. switching to user mode
    - 3. 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

Expensive

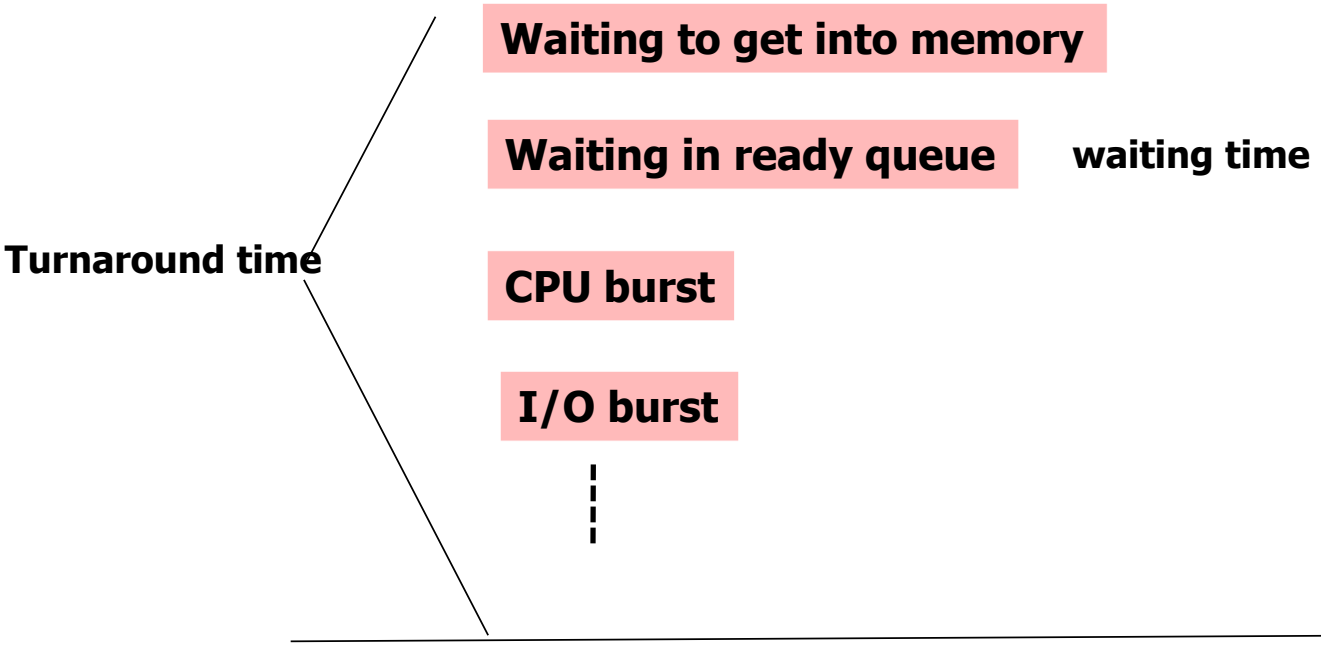
Scheduling Criteria

- Scheduling criteria
  - CPU utilization
    - keep the CPU as busy as possible
  - Throughput
    - # of processes that complete their execution per time unit
  - Turnaround time
    - Completion of process – submission of 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,

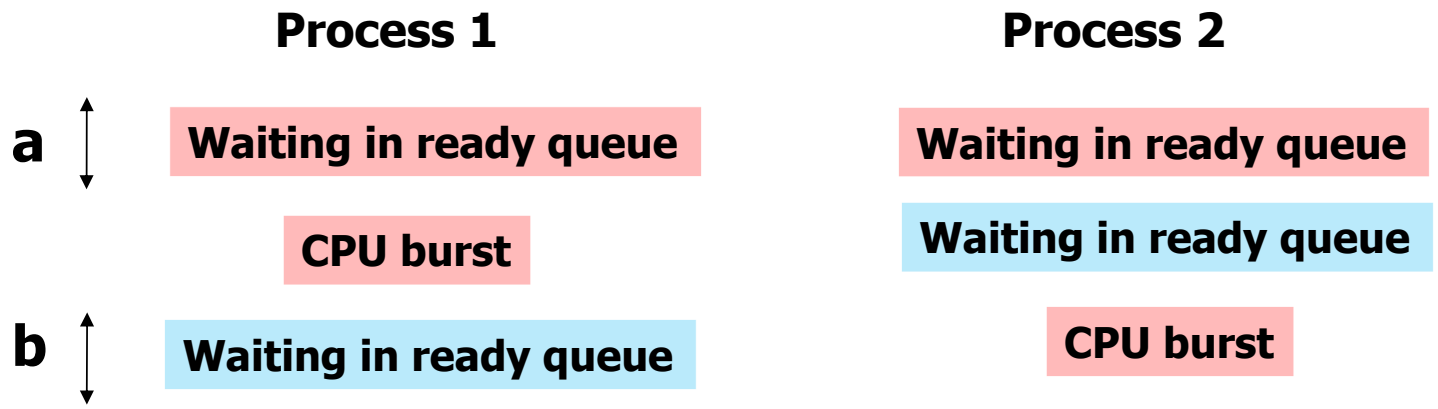
Applied to the entire system

From the point of view of a particular process

- A difference between turnaround time and waiting time

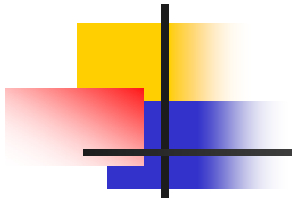


A difference between response time and waiting time



The duration of "a" is the response time of process 1

The sum of durations (a+b) is the waiting time of process 1



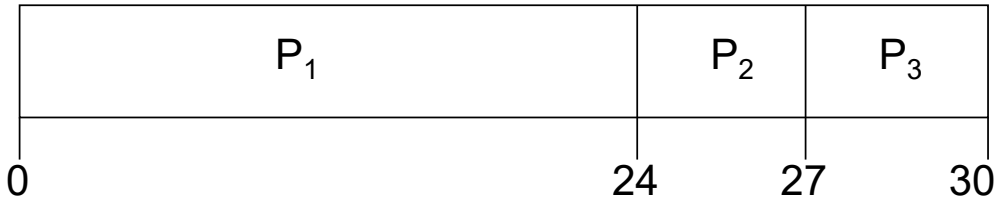
- Optimization criteria
  - Maximize CPU utilization
  - Maximize throughput
  - Minimize turnaround time
  - Minimize waiting time
  - Minimize response time
  - Average ?
  - Variance ?

Scheduling Algorithms

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

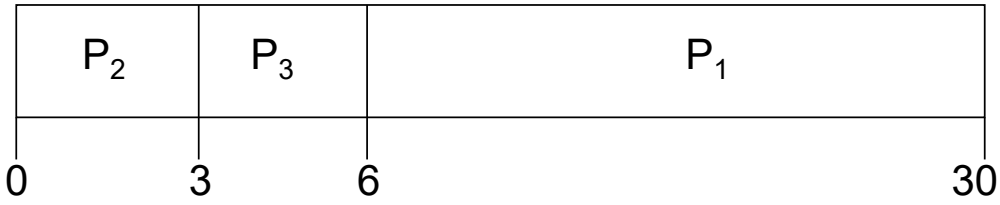
Process	Burst Time
P_1	24
P_2	3
P_3	3

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

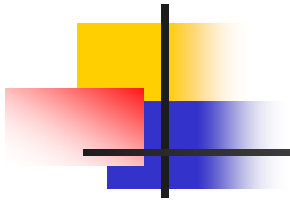


  - Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
  - Average waiting time: (0 + 24 + 27)/3 = 17

- First-Come, First-Served (FCFS) Scheduling (cont'd)
  - Process arrival order
    - P_2 , P_3 , P_1
  - The Gantt chart for the schedule is:

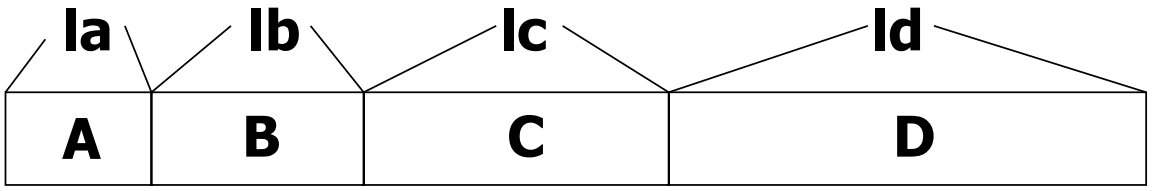


  - Waiting time for $P_1 = 6; P_2 = 0; P_3 = 3$
  - Average waiting time: (6 + 0 + 3)/3 = 3
  - Much better than previous case
  - *Convoy effect*
    - short process behind long process
    - e.g. 1 cpu-bound process, many I/O bound processes
    - This increases average waiting time



- 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 it 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

- Why is it optimal?



Average waiting time

A: 0 B: la C: la+lb D: la+lb+lc (3la+2lb+lc)/4

Average turnaround time

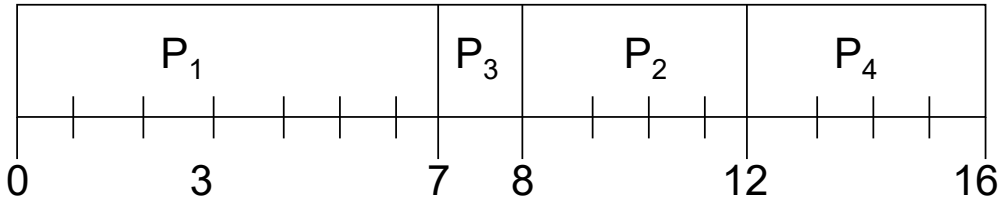
A: la B: la+lb C: la+lb+lc D: la+lb+lc+ld

(4la+3lb+2lc+ld)/4

- SJF (non-preemptive)

	Arrival time	CPU burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

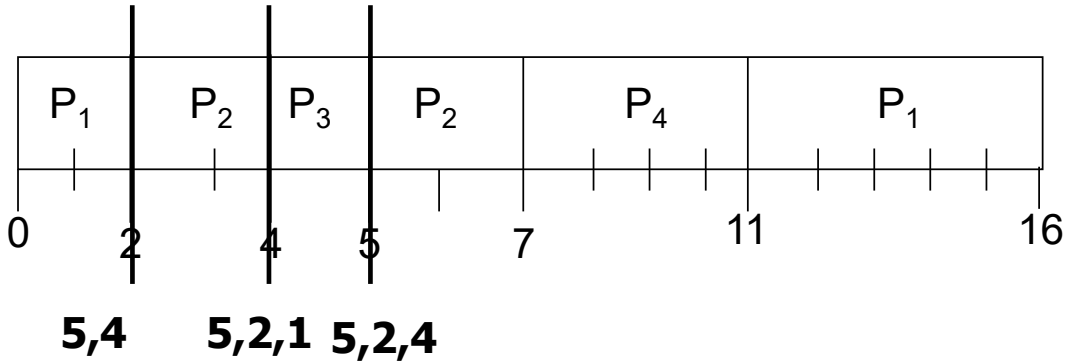
- SJF (non-preemptive)



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

- SJF (preemptive)

	Arrival time	CPU burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



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