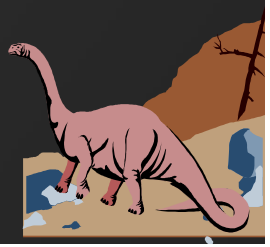
SCHEDULING

The OS as a time keeper

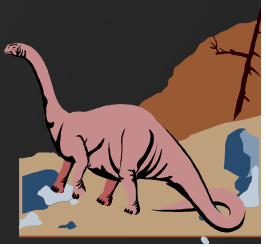
Scheduling

- In multiprogramming systems, the OS decides which one to activate when there is more than one runnable process (in the ready queue).
- The decision is made on the part of the OS called **scheduler** using a *scheduling algorithm.*



Scheduling

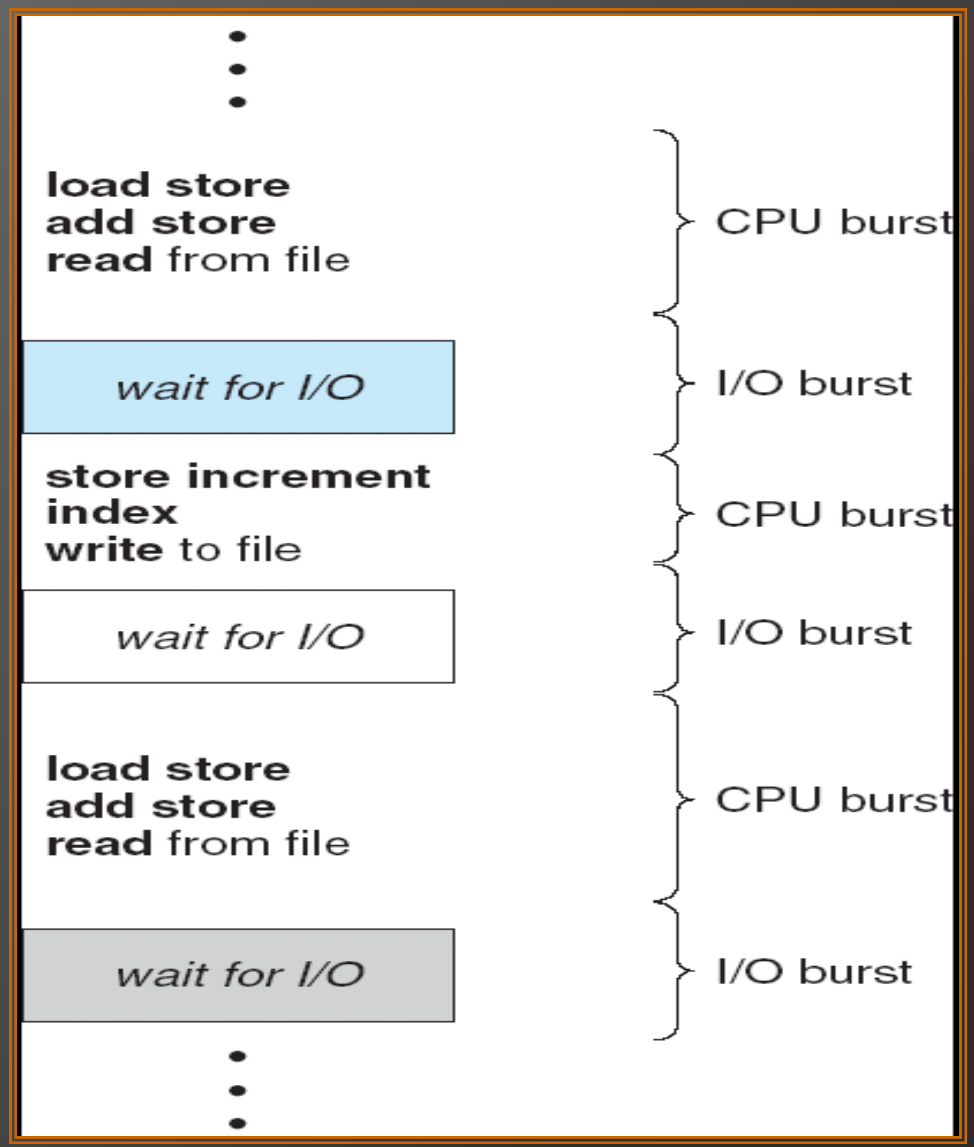
- Refers to the set of policies and mechanisms to control the order of work to be performed by a computer system.

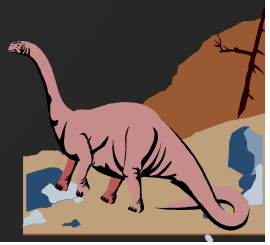


Scheduling Concepts

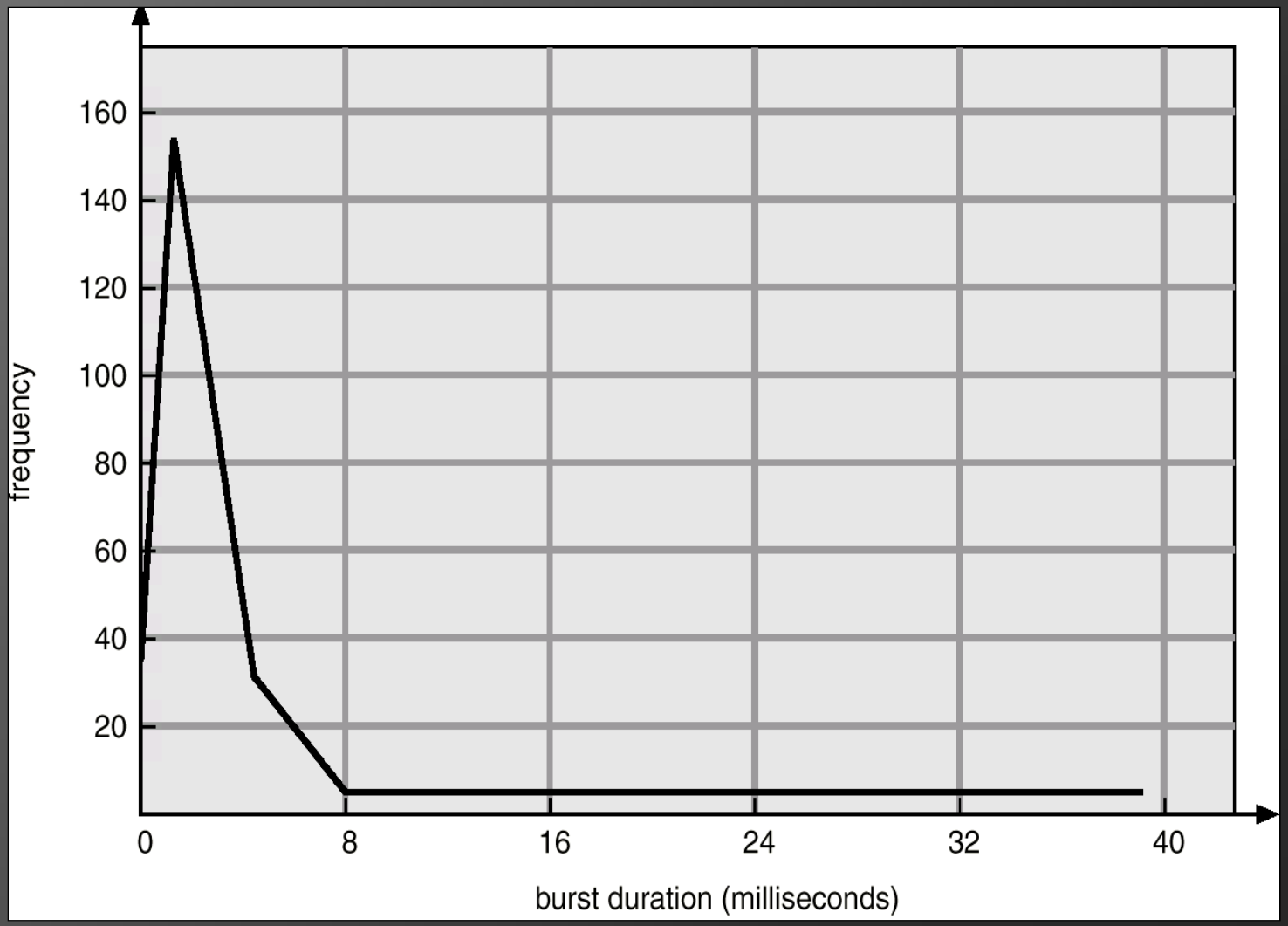
- OS must keep the CPU busy as much as possible by executing a (user) process until it must wait for an event, and then switch to another process.
- Maximum CPU Utilization obtained with multiprogramming.
- Processes alternate between consuming CPU cycles (CPU-burst) and performing I/O (I/O-Burst)

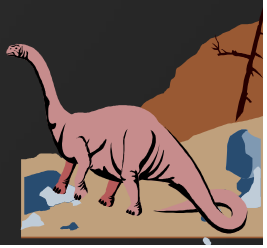
Alternating Sequence of CPU and I/O Bursts





Histogram of CPU-burst Times





Types of Scheduler

1. **Long-term (job) scheduler**
   – admits more jobs when the resource utilization is low and blocks the incoming jobs from entering ready queue when utilization is high.

2. **Medium-term scheduler (swapper)**
   – releases suspended processes from memory by swapping them out when the memory becomes over committed.

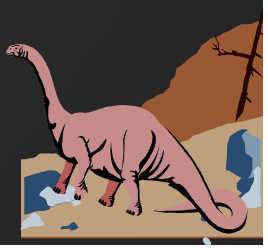
Both schedulers (long and medium) controls the level of multiprogramming and avoid overloading the system by many processes.

Schedulers (cont.)

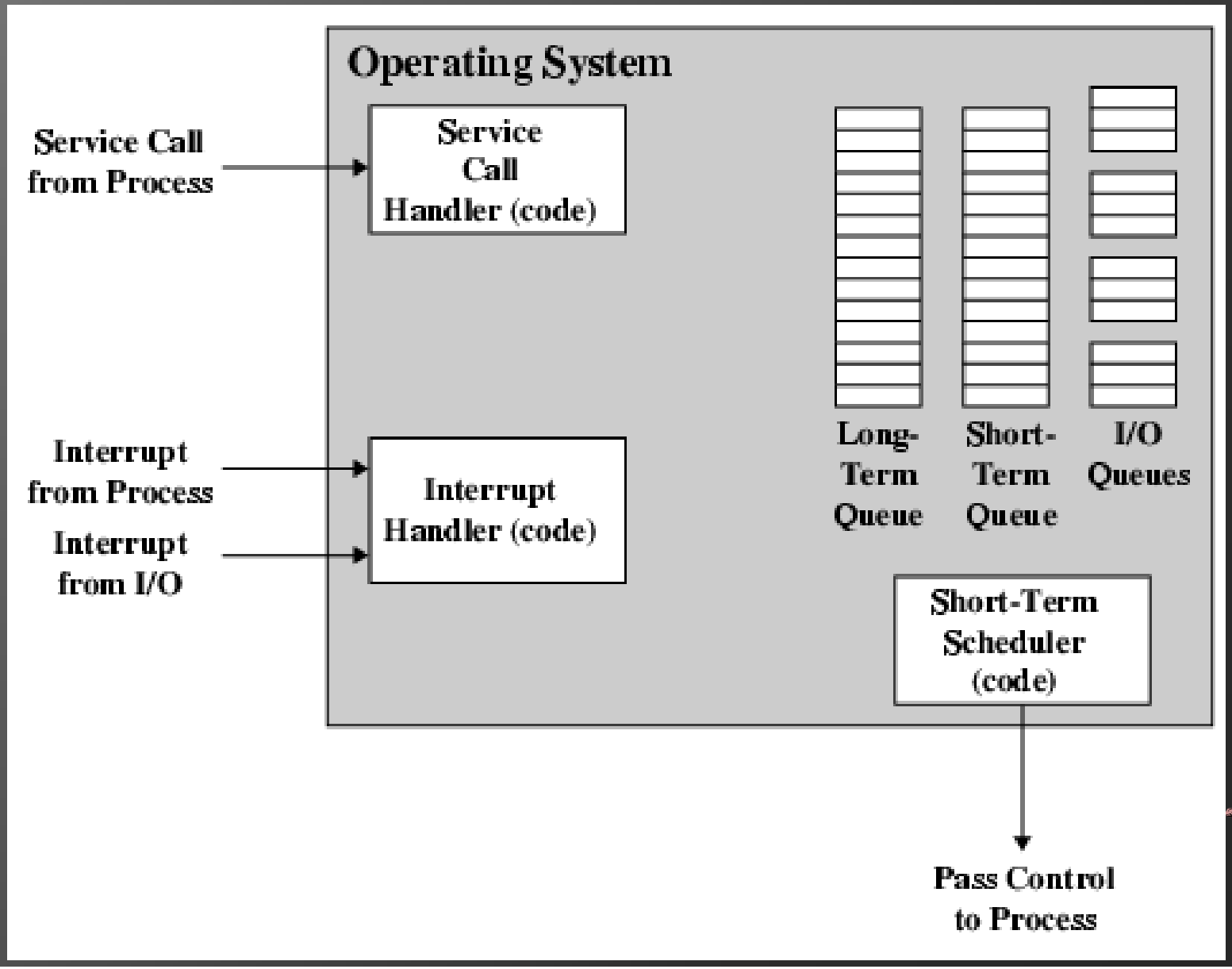
3. **Short-term (CPU) scheduler** (dispatcher) – controls the CPU sharing among the "ready" processes.

Usually, the next process to execute is selected under the following circumstances:

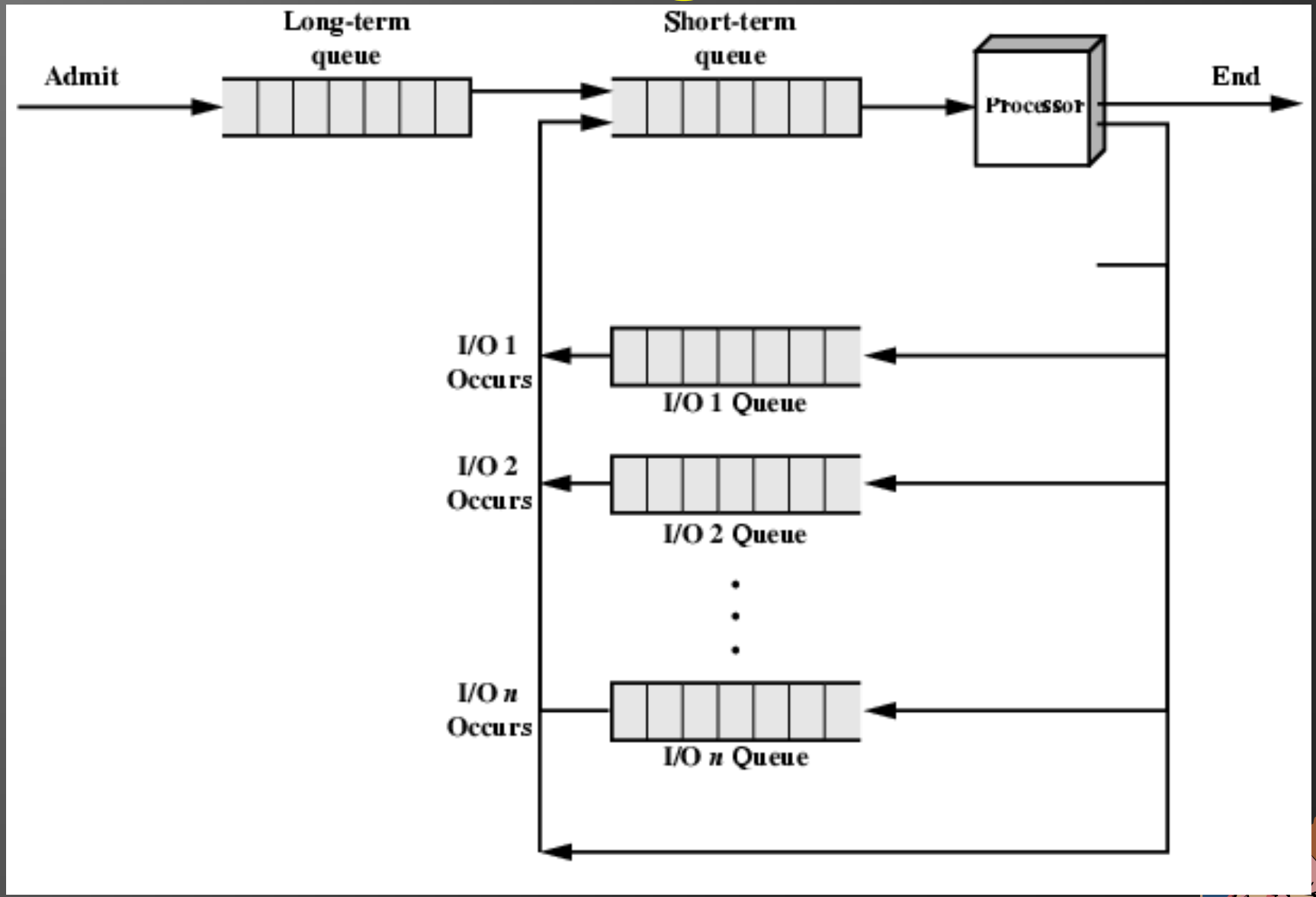
- When a process must wait for an event
- When an event occurs (I/O completion, quantum expired)



Key Elements of O/S



Process Scheduling

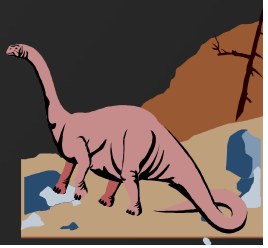


Scheduling Criteria

- *Goal:* Optimize the system performance, and yet-provide responsive service.
- Criteria used:

1. **CPU Utilization** – the percentage of time that the CPU is busy.

$$\text{CPU Utilization} = \frac{\text{total processes burst times}}{\text{total CPU execution time of all processes}} \times 100\%$$



Scheduling Criteria

2. **System throughput** – number of processes completed per time unit.

   Throughtput = $\frac{\text{number of processes}}{\text{total CPU execution time of all processes}}$

3. **Turnaround Time** – elapsed time from the time the process is submitted for execution to the time it is completed.

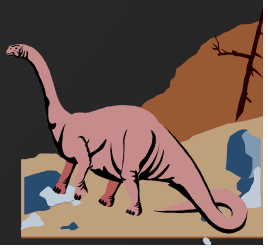
   TAT = end time – arrival time in ready queue (original)

Scheduling Criteria

4. **Waiting time** – amount of time a process has been waiting in the ready queue.

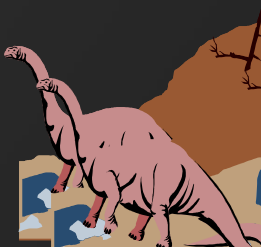
   WT = start of execution – arrival time in RQ

5. **Response Time** – amount of time it takes from when a process is submitted until the first response is produced, not output (for time sharing environment). This is the time taken by the system to respond to a user input.



Scheduling Criteria

6. **Balanced Utilization** – percentage of time all resources are utilized
7. **Predictability** – consistency
8. **Fairness** – the degree to which all processes are given equal opportunity to execute. Avoidance of starvation
9. **Priorities** – give preferential treatment to processes with higher priorities.



Note:

- Different algorithms may have different properties and may favor one type of criterion over the other. So, the design of a scheduler usually involves a careful balance of all requirements and constraints.

Optimization Criteria

- Max CPU utilization (increased)
- Max throughput (increased)
- Min turnaround time (decreased)
- Min waiting time (decreased)
- Min response time (decreased)

Scheduling Policies

- **Non-preemptive scheduling**
  - Process's execution cannot be preempted or suspended.
- **Preemptive scheduling**
  - Force the current executing processes to release the CPU on certain events such as clock interrupt, I/O interrupts or a system call. (with suspension).

Common Scheduling Algorithms

Non-Preemptive	Preemptive
FCFS	**RR**

Both Preemptive and Non-preemptive:

SJF & PRIORITY

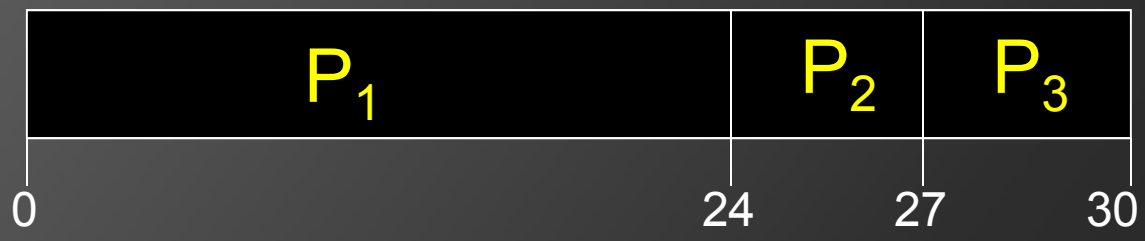
First-Come, First-Served (FCFS) Scheduling algorithm

- Also known as **First in First out (FIFO).** Simplest scheduling policy
- Arriving jobs are inserted into the tail of the ready queue and the process to execute next is removed from the head of the queue.
- Jobs are scheduled in the order in which they are received.

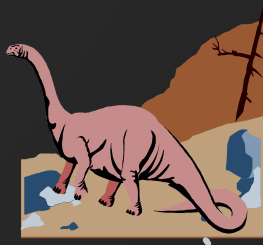
FCFS Scheduling algorithm

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

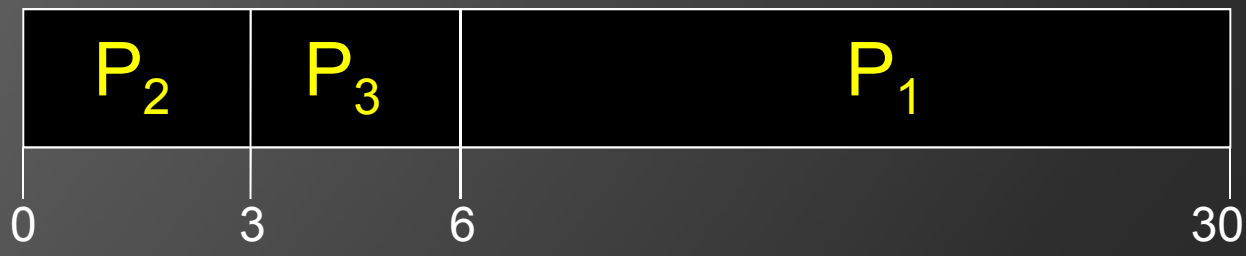


FCFS Scheduling (Cont.)

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
- Note: Better than the previous case

Lessons learned:

- Long CPU-bound job may hog the CPU and may force shorter jobs to wait prolonged periods.
- This may lead to a lengthy queue of ready jobs (**convoy effect**)

Examples:

1. Suppose that the following processes arrive for execution, what is the CPU Utilization, throughput and average TAT and WT for these processes with the FCFS scheduling algorithm

a)

Process	BT
A	24
B	10
C	3

b)

Process	BT
P1	4
P2	1
P3	8

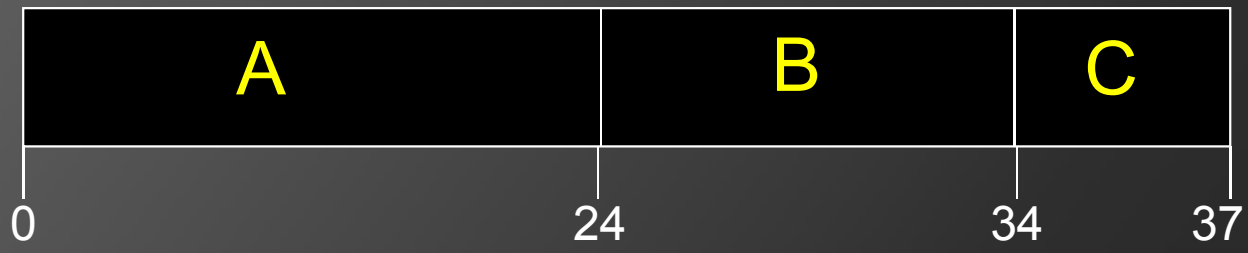
Solution:

a)

Process	BT	WT	TAT
A	24	0	24
B	10	24	34
C	3	34	37

CPU = 100%

Thr = 3 / 37

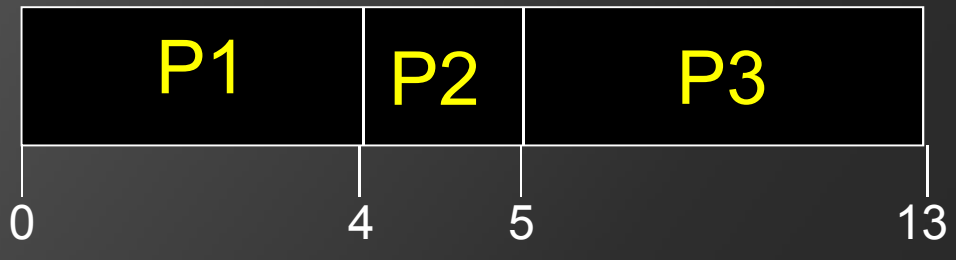


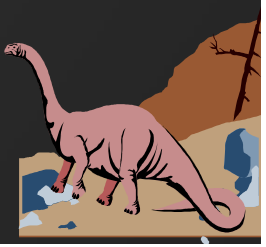
b)

Process	BT	WT	TAT
P1	4	0	4
P2	1	4	5
P3	8	5	13

CPU = 100%

Thr = 3 / 13



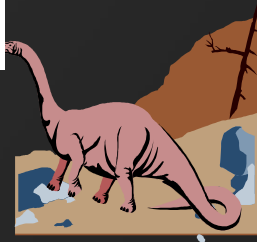


Examples:

2. Calculate the TAT and WT of each process. Consider the process arrival and burst times of these processes as shown below.

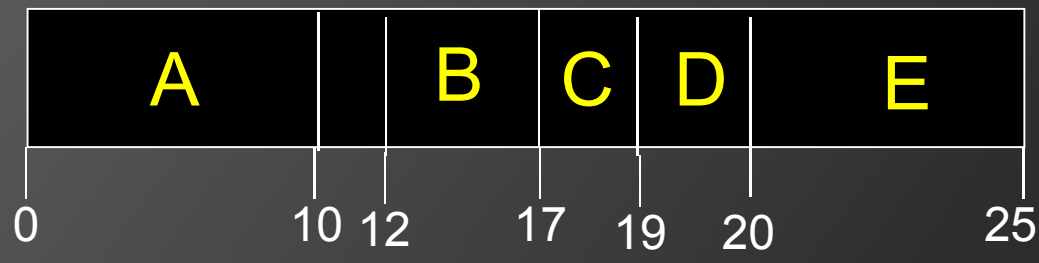
Process	BT	AT
A	10	0
B	5	12
C	2	13
D	1	18
E	5	20

Process	BT	AT
P1	8	0.0
P2	4	0.4
P3	3	8.2
P4	2	14.6
P5	1	16.0



Solution #2A:

Process	BT	AT	WT	TAT
A	10	0	0-0	10-0
B	5	12	12-12	17-12
C	2	13	17-13	19-13
D	1	18	19-18	20-18
E	5	20	20-20	25-20



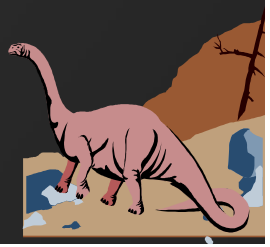
CPU = 23 / 25 * 100%

Thr = 5 / 25

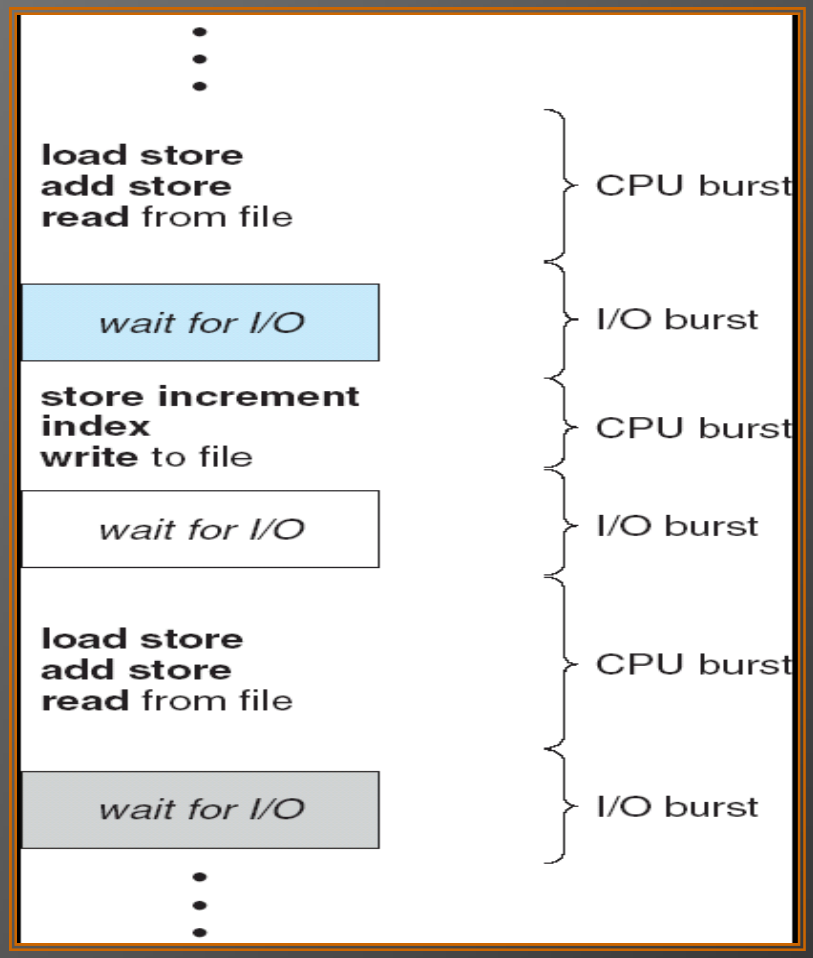
Seatwork

- Calculate the CPU Utilization, average TAT and WT for the execution of the process below. Consider their arrival times and their burst times

Process	AT	BT
J1	13	4
J2	5	6
J3	3	6
J4	24	10
J5	15	7
J6	18	7
J7	14	8



Process' CPU and I/O requests



- CPU and I/O requests use any of the CPU scheduling algorithm.
- After CPU burst, the I/O burst follows.
- CPU burst will always be the LAST to perform for the completion of the execution.

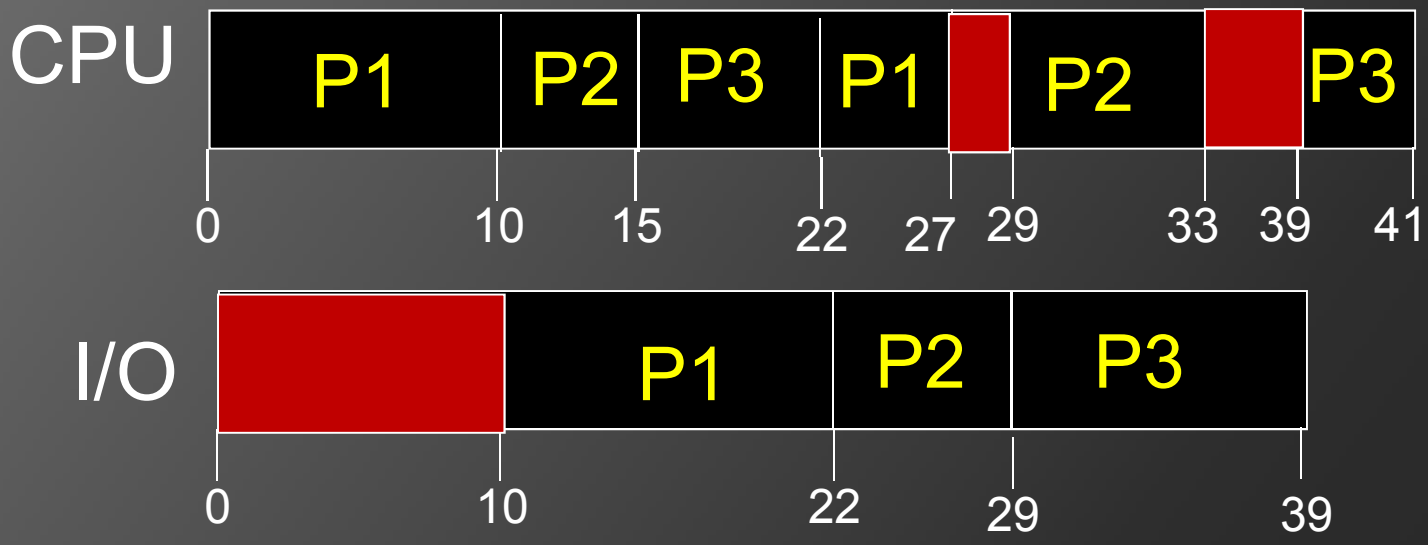
Sample problem using CPU and I/O

- With jobs in the ready queue and I/O queue all scheduled using FCFS, show the Gantt chart showing the execution of these processes. Determine CPU and I/O utilization

Job ID	AT	CPU	I/O	CPU
P1	0	10	12	5
P2	3	5	7	4
P3	10	7	10	2

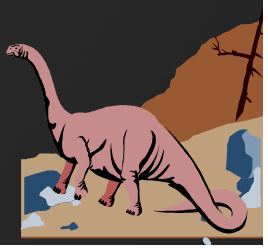
Solution

Job ID	AT	CPU	I/O	CPU
P1	0	10	12	5
P2	3	5	7	4
P3	10	7	10	2



CPU = 33 / 41 * 100%

I/O = 29 / 39 * 100%



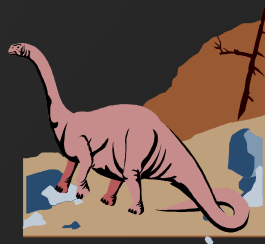
Seatwork

- With jobs in the ready queue and I/O queue all scheduled using FCFS, show the Gantt chart showing the execution of these processes.

Process	AT	CPU	I/O	CPU
J1	0	4	10	5
J2	5	6	10	4
J3	16	5	5	2
J4	14	20	2	5

Shortest-Job-First Scheduling

- Selects job with the shortest (expected) burst time first. Shorter jobs are always executed before long jobs
- One major difficulty with SJF is the need to know or estimate the burst time of each job



Shortest-Job-First Scheduling

- Another difficulty is long running times may starve because the CPU has a steady supply of short jobs
- But SJF is optimal – gives minimum WT for a given set of processes.
- Preemptive SJF is also known as Shortest Remaining Time First (SRTF)

Two schemes:

- **Non-Preemptive** – once the CPU has given to the process it cannot be preempted until it completes its CPU burst. (SJF non-pre)

- **Preemptive** – if a new process arrives with CPU burst less than the remaining time of current executing process, preempt. (SRTF)

Exercise:

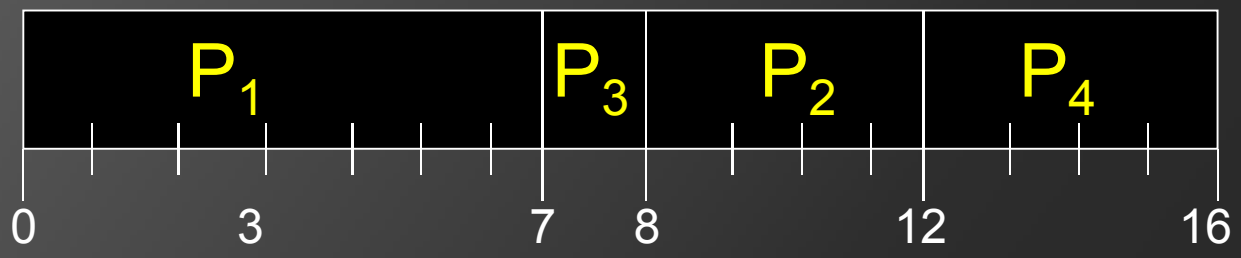
Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

Draw the gantt charts for NP-SJF and SRTF.
Solve the average WT.

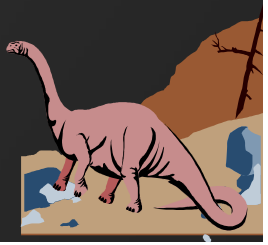
Solution:

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SJF (**Non-preemptive**)



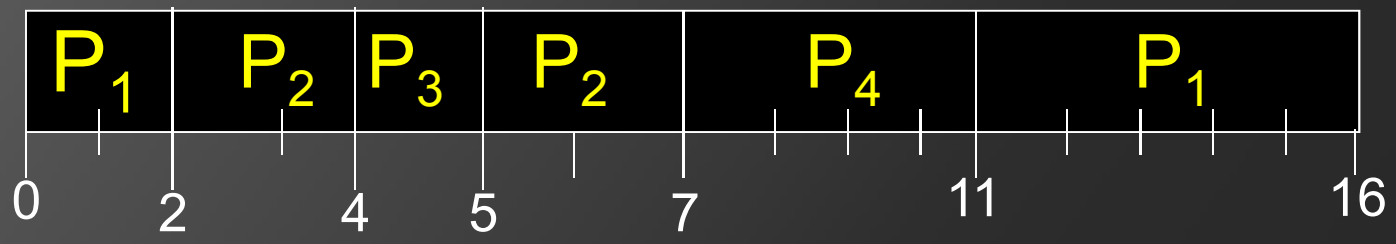
- Average waiting time = (0 + 6 + 3 + 7)/4 = 4



Solution

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SJF (**Preemptive**)



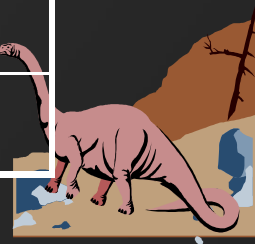
- Average waiting time = (9 + 1 + 0 +2)/4 = 3

Your turn

- Obtain the average waiting time and the turnaround time, including CPU Utilization and Gantt Chart for the following set of processes using SJF (Non-Pre), and SRTF

Process	BT	AT
A	10	0
B	6	5
C	7	7
D	3	6
E	1	3

Process	BT	AT
J1	6	11
J2	8	0
J3	12	4
J4	2	2
J5	5	5



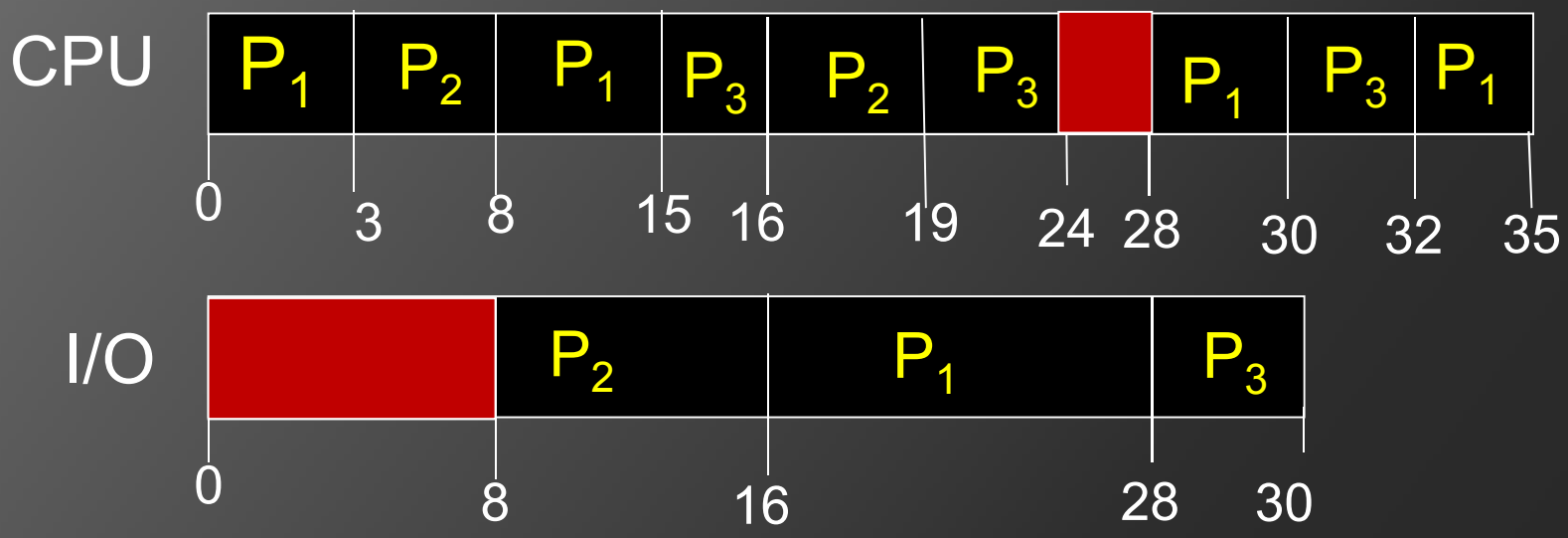
CPU and I/O request

- With jobs in the ready queue scheduled as **SRTF** and the jobs in the I/O queue scheduled as **FCFS**, show the Gantt chart showing the execution of these processes. What is the CPU and I/O utilization?

Job ID	AT	CPU	I/O	CPU
P1	0	10	12	5
P2	3	5	8	3
P3	10	6	2	2

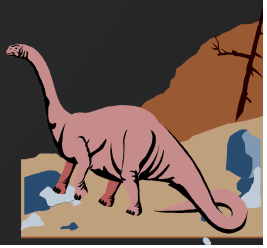
Solution

Job ID	AT	CPU	I/O	CPU
P1	0	10	12	5
P2	3	5	8	3
P3	10	6	2	2



CPU = 31 / 35 * 100%

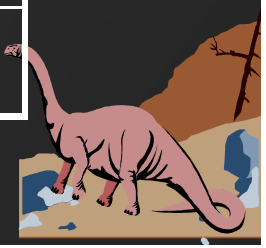
I/O = 22 / 30 * 100%



Seatwork

- With jobs in the ready queue scheduled as **SRTF** and the jobs in the I/O queue scheduled as **FCFS**, show the Gantt charts showing the execution of these processes. What is the CPU and I/O utilization?

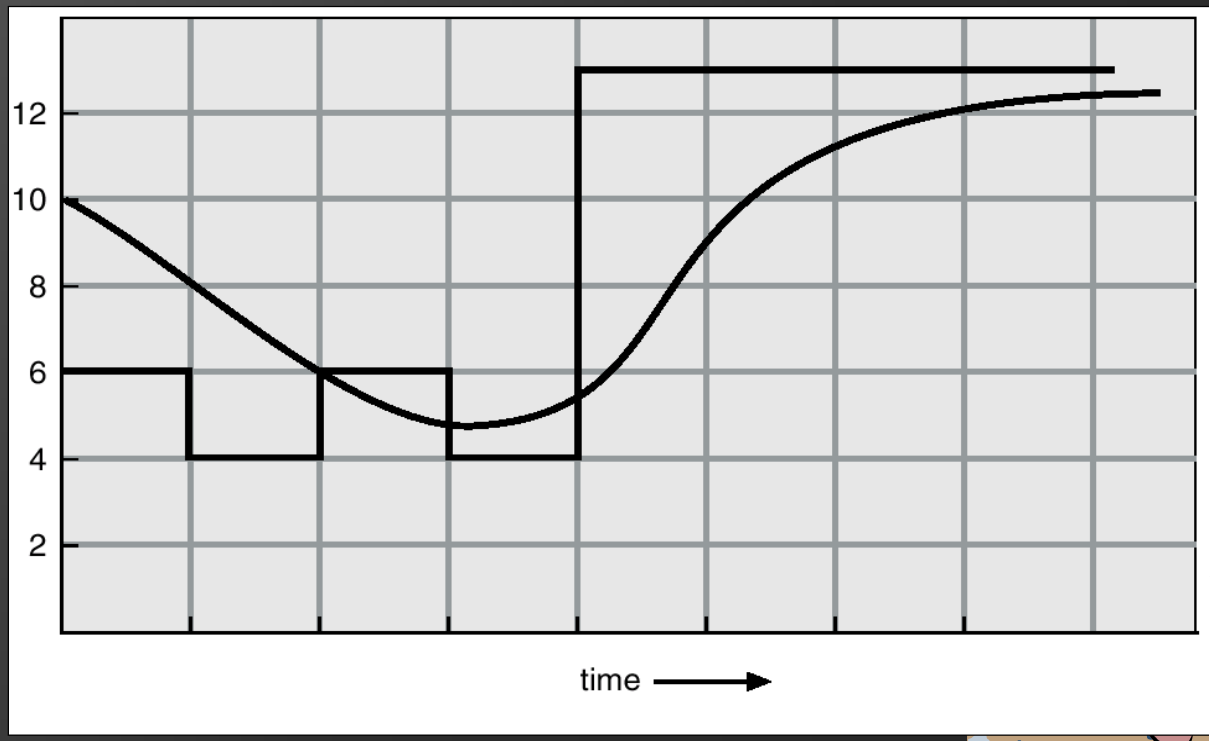
Process	AT	CPU	I/O	CPU
J1	0	5	10	6
J2	3	7	5	2
J3	7	3	4	4
J4	15	2	5	2



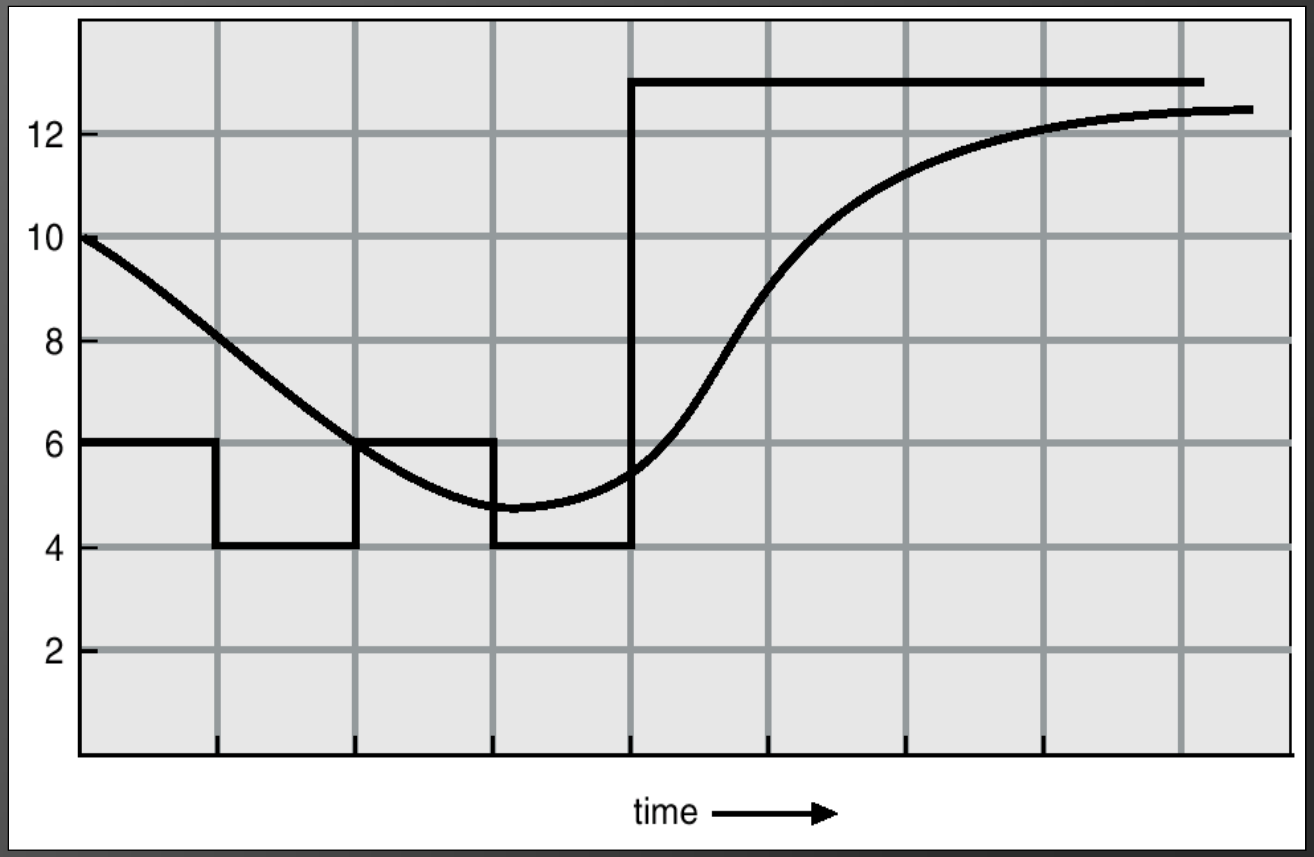
Determining Length of Next CPU Burst

- Can only estimate the length.
- Can be done by using the length of previous CPU bursts, using exponential averaging.

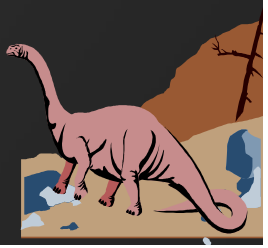
Prediction of the Length of the Next CPU Burst



Prediction of the Length of the Next CPU burst



Real Burst Time :	-	6	4	6	4	13	13	13
Guess BT :	10	8	6	6	5	9	11	**12**

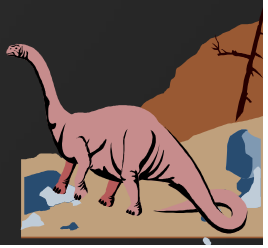


Priority Scheduling Algorithm (PSA)

- Priority number (integer) is associated with each process
- CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive PSA
  - Non-Preemptive PSA

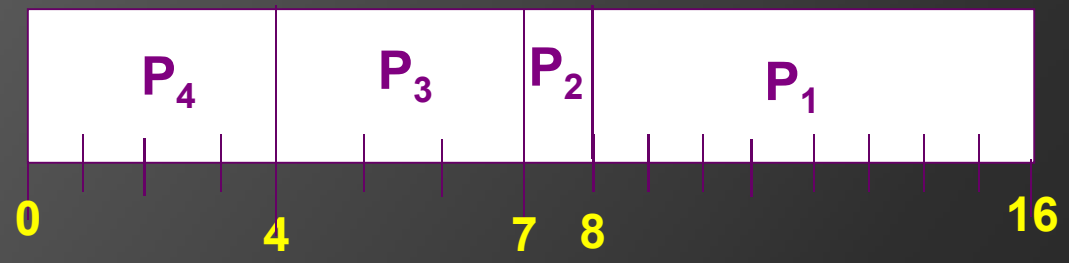
PSA

- SJF is a priority scheduling where priority is the predicted next CPU burst time
- Problem:
  - Starvation – low priority processes may never execute
- Solution:
  - Aging – as time progresses increase the priority of the process



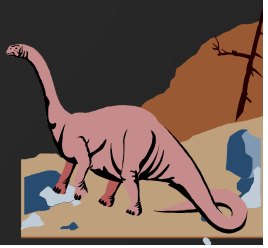
Sample Problem #1:

Process	BT	PL (Highest Priority = 1)
P1	8	4
P2	1	3
P3	3	2
P4	4	1



Average WT: (8+7+4+0) / 4 = 4.75

Average TAT: (16+8+7+4) / 4 = 8.75

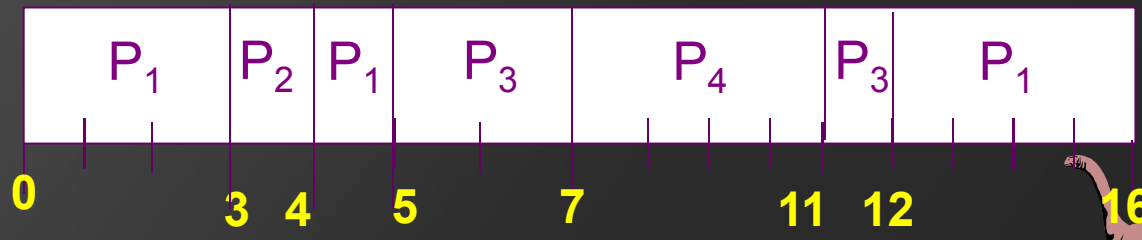


Sample Problem #2:

Process	AT	BT	PL (HPL =1)
P1	0	8	4
P2	3	1	3
P3	5	3	2
P4	7	4	1

Non-Preemptive PSA:

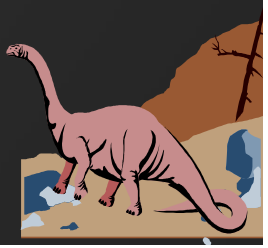
Preemptive PSA:



Round Robin (RR) Scheduling

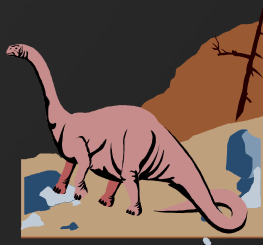
- Process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.

- If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than (*n*-1)*q* time units.

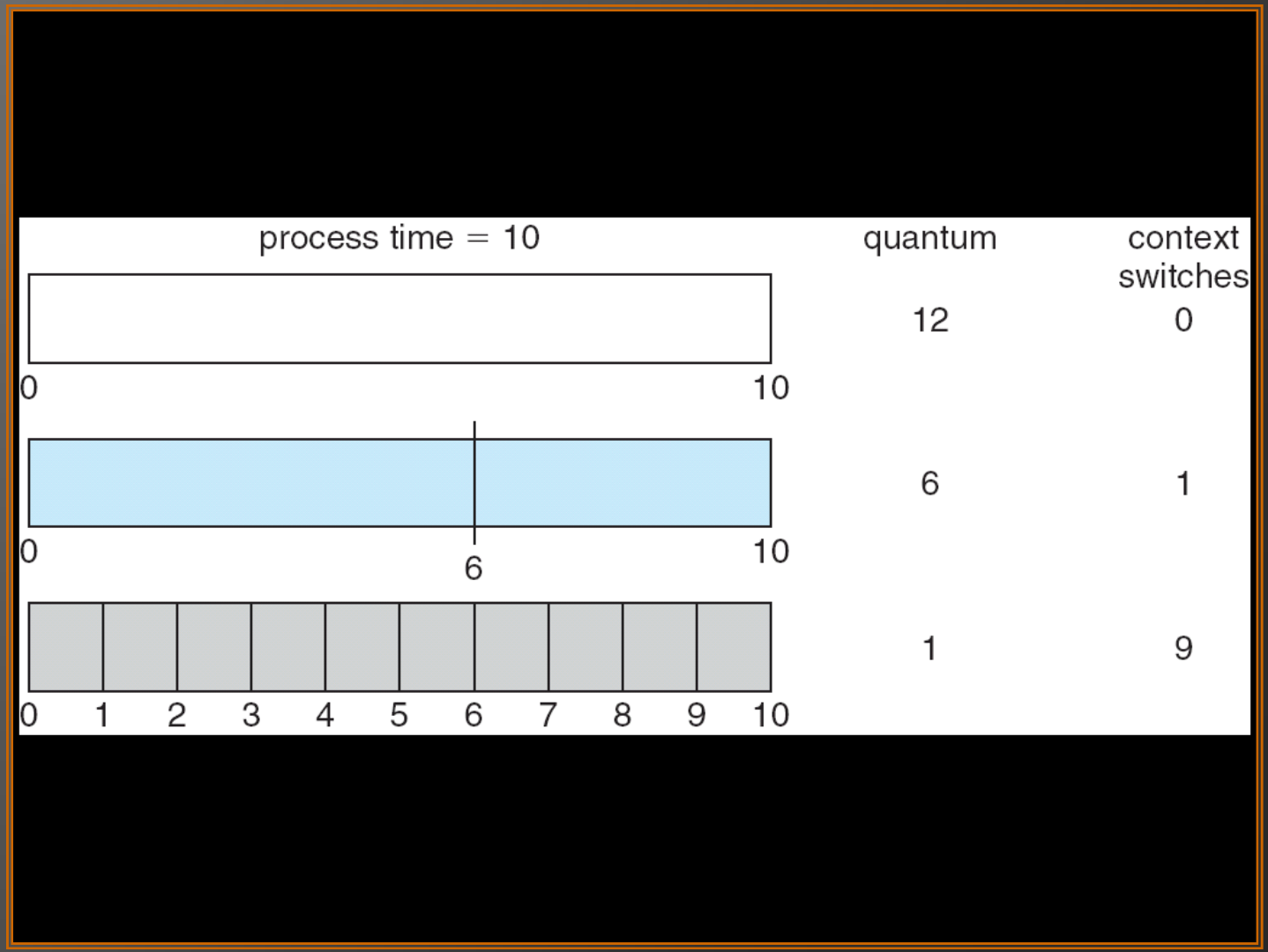


Round Robin (RR)

- Performance
  - q large $\Rightarrow$ FIFO
  - q small $\Rightarrow$ q must be large with respect to context switch, otherwise overhead is too high

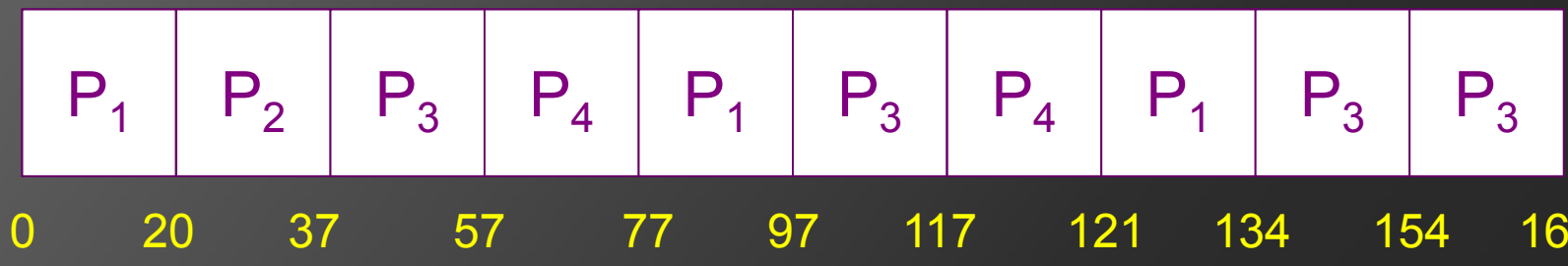


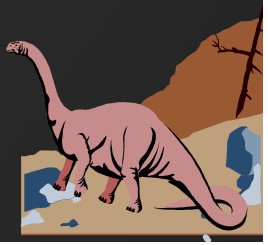
Time quantum and context switches



Sample Problem:

Process	Burst Time	
P1	53	QT = 20
P2	17	
P3	68	
P4	24	





Your turn:

- Given the following jobs, schedule using PPSA and RR (QT=4).
- Answer the queries on the next slide.

Job	AT	BT	PL HPL=1
1	0	18	3
2	8	8	2
3	10	5	2
4	17	12	1
5	25	8	3
6	30	6	1

Using PPSA, answer the following:

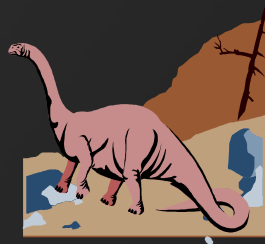
1. What job is currently allocated to the CPU @ time 18?
2. How many jobs are finished at time 24?
3. What is the last job to finish?
4. When did job 2 start executing?
5. When did job 4 start executing?
6. After job 2 totally finished, which job was allocated to the CPU?
7. What is the waiting time of job 5?
8. What is the finish time of job 6?

Using RR, answer the following:

1. When was job 2 first assigned to the CPU?
2. How many jobs are finished at time 22?
3. Which job was allocated the CPU at time 19?
4. Which job was allocated the CPU at time 29?
5. After job 3 was totally finished, which job was allocated to the CPU?
6. What is the finish time of job 3?
7. What is the last job to finish?
8. What is the turnaround time of job 1?

Comparison between PPSA and RR:

- Comparing the 2 algorithms:
  1. Which algorithm did job 2 start the earliest?
  2. Which algorithm did job 5 start the latest?
  3. Which algorithm did job 1 finish the earliest?
  4. Which algorithm did job 1 finish the latest?



Activity

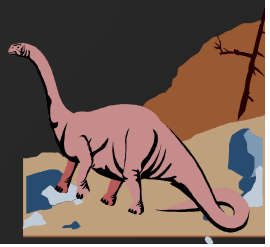
- For the following processes, schedule using:
  - FCFS - SJF - SRTF
  - NPPSA - PPSA - RR (Q=4)

J	BT	AT	P
A	12	17	1
B	15	18	4
C	7	1	2
D	11	20	2
E	4	5	3
F	24	0	5

- Rank each algorithm according to:
  a. Max CPU Utilization
  b. Min TAT
  c. Min WT

Answers:

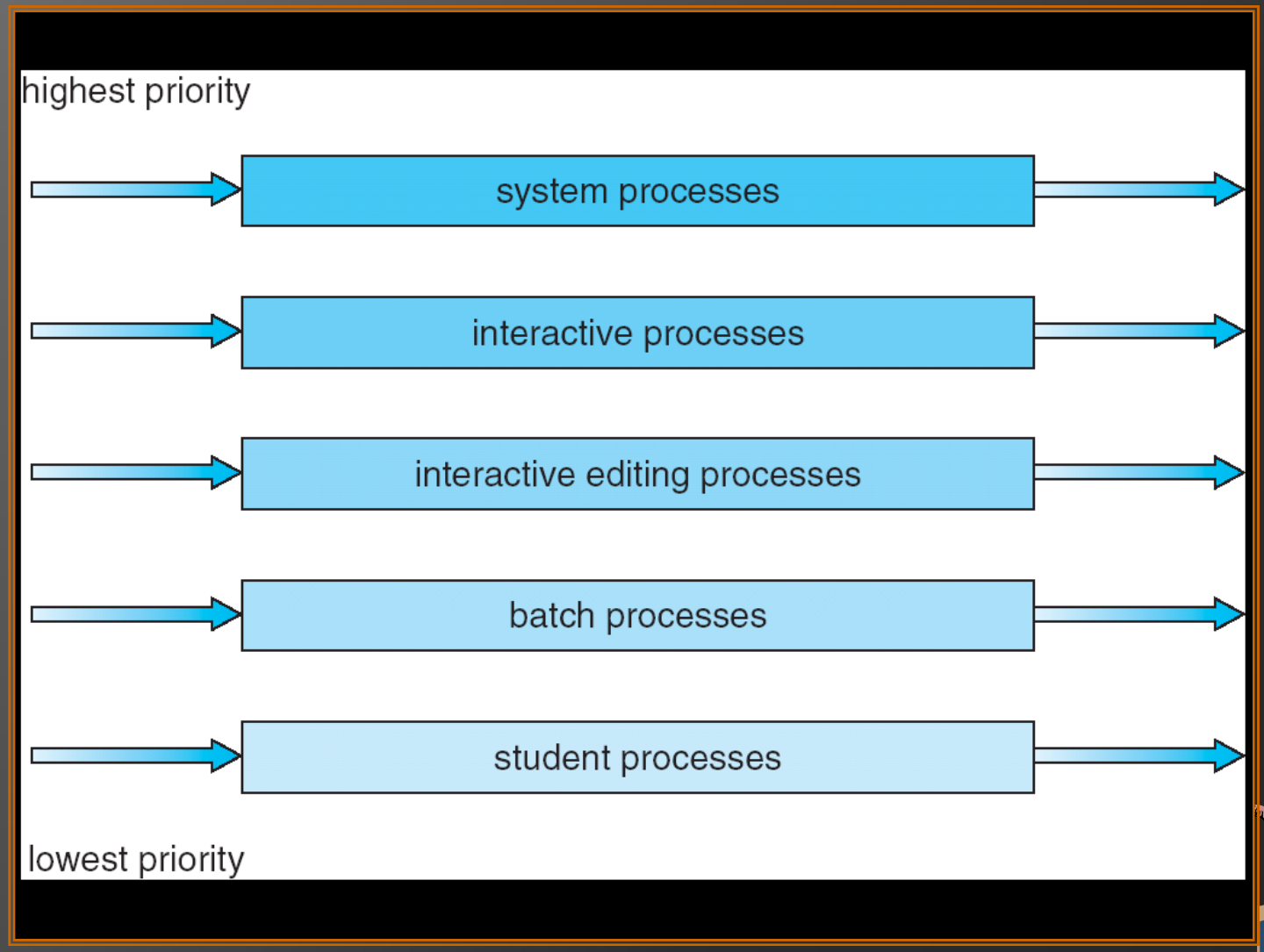
- CPU Utilization: all equal
- Turnaround time:
  - SRTF, PP, SJF, FCFS, NPP, RR
- Waiting time:
  - SRTF, PP, SJF, FCFS, NPP, RR



Multilevel Queue

- Ready queue is partitioned into separate queues:
  foreground (interactive) & background (batch)
- Each queue has its own scheduling algorithm
  - foreground – RR
  - background – FCFS
- Serves foreground first then background
  Possibility of starvation.

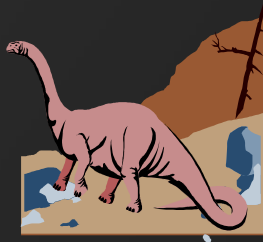
Multilevel Queue Scheduling



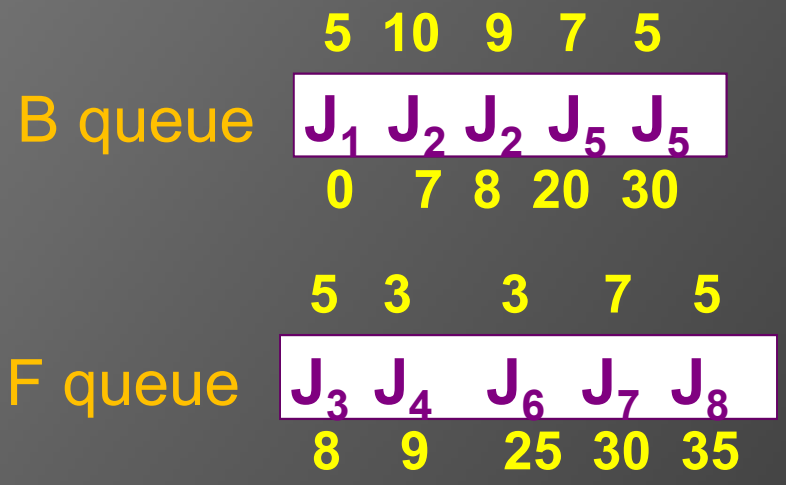
MLQ Example

- Assume that the CPU scheduler uses MLQ with 2 levels: foreground queue for type F (jobs uses SJF); and background queue for type B (jobs uses FCFS). Scheduling algorithms between queues is fixed preemptive priority (F being the highest priority)

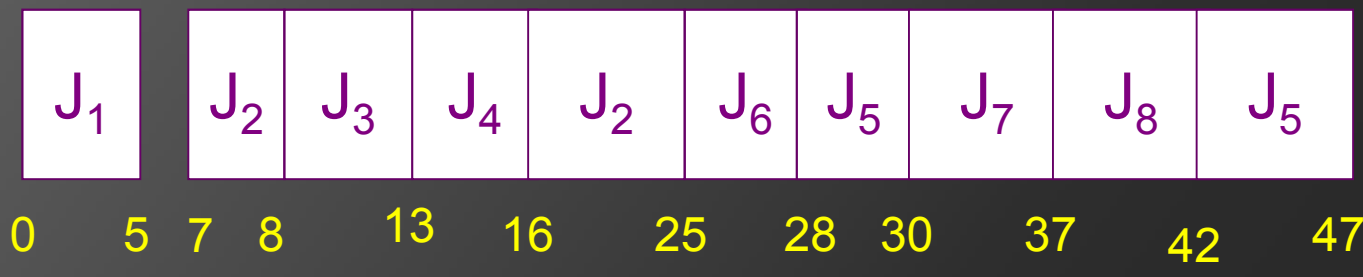
Job	AT	BT	Type
1	0	5	B
2	7	10	B
3	8	5	F
4	9	3	F
5	20	7	B
6	25	3	B
7	28	7	F
8	37	5	F

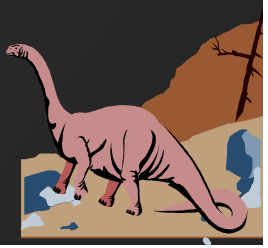


Solution



Job	AT	BT	Type
1	0	5	B
2	7	10	B
3	8	5	F
4	9	3	F
5	20	7	B
6	25	3	F
7	30	7	F
8	35	5	F





Queries on the previous example:

1. Which job was allocated to the CPU at time 8?
2. Which job was allocated to the CPU at time 26?
3. Which job was allocated to the CPU at time 35?
4. Which job was allocated to the CPU at time 43?
5. What is the turnaround time for job 5?
6. What is the waiting time for job 6?
7. What is the finish time of job 2?
8. What is the finish time of job 3?
9. What is the finish time of job 7?
10. What is the finish time of job 8?

Your Turn...

Assuming the ready queue is partitioned into 2 queues:

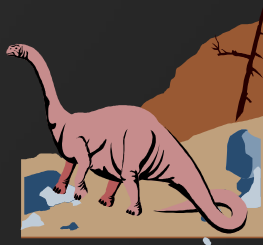
Background queue - processes with priorities from 3-5 (uses FCFS); and

Foreground queue- processes with priorities 1 and 2. (uses RR with Q=6.

Algorithm between queues is preemptive priority.

Draw the Gantt chart of the CPU.

J	BT	AT	P
A	12	17	1
B	15	18	4
C	7	1	2
D	11	20	2
E	4	5	3
F	24	0	5

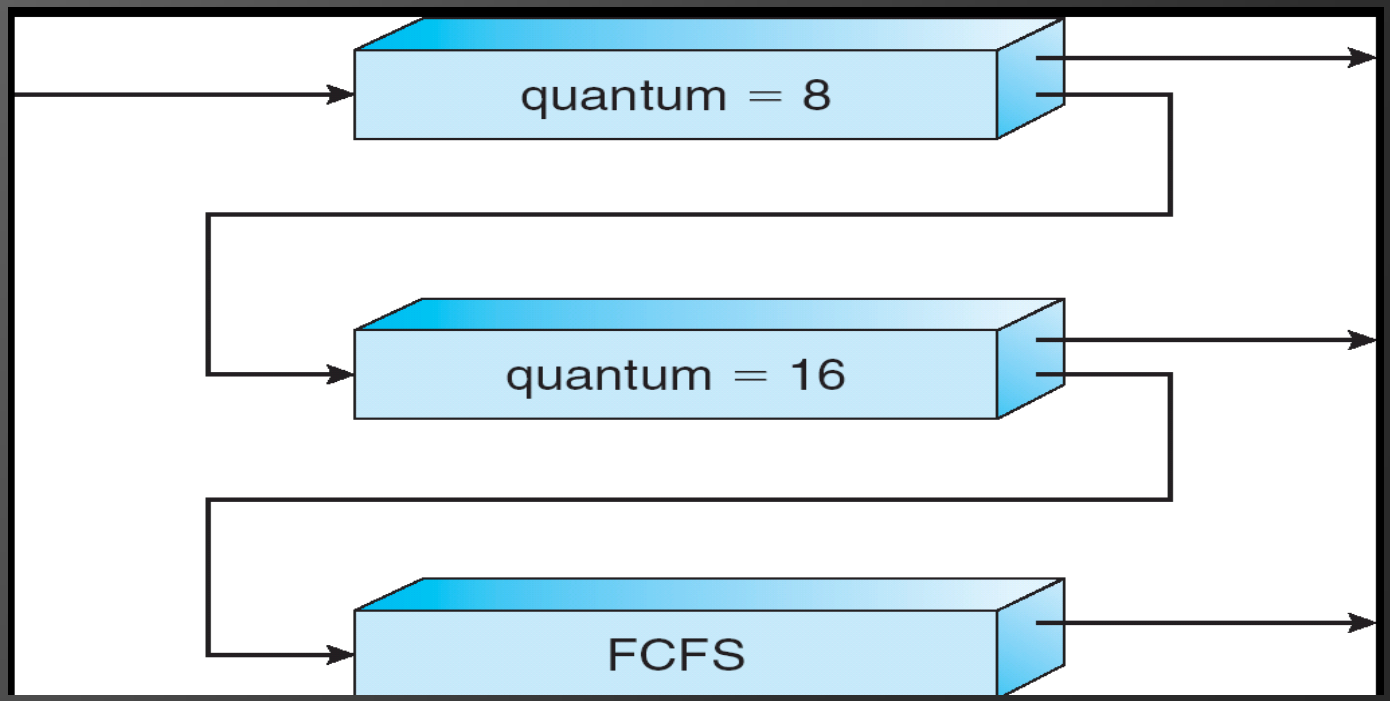


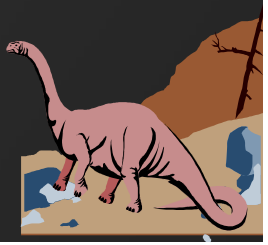
Multilevel Feedback Queue

- A process can move between the various queues; aging can be implemented this way
- Multilevel-feedback-queue scheduler defined by the following parameters:
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine when to upgrade a process
  - method used to determine when to demote a process
  - method used to determine which queue a process will enter when that process needs service

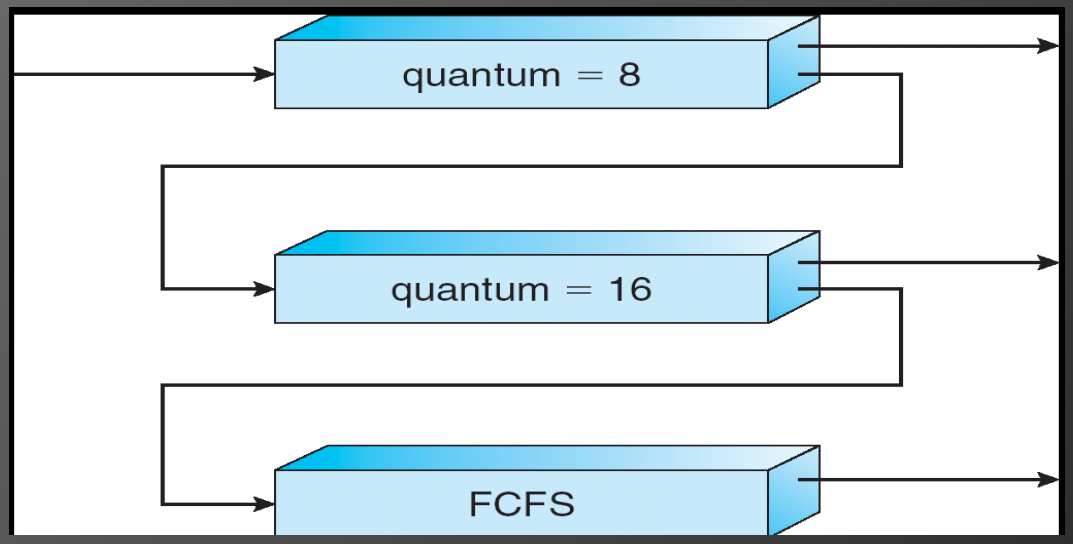
Example of Multilevel Feedback Queue

- Three queues:
  - Q_0 – RR with time quantum 8 milliseconds
  - Q_1 – RR time quantum 16 milliseconds
  - Q_2 – FCFS





Multilevel Feedback Queue: Scheduling



- New job enters queue Q_0 which is served FCFS. When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds, job is moved to queue Q_1.
- At Q_1 job is again served FCFS and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q_2.

MLFQ Example

- Schedule using an MLFQ:

  Q1: RR Q=3

  Q2: RR Q=5

  Q3: FCFS

Job	BT	AT
A	10	4
B	13	3
C	5	0
D	6	1
E	18	1

Try it!

- Schedule using an MLFQ:

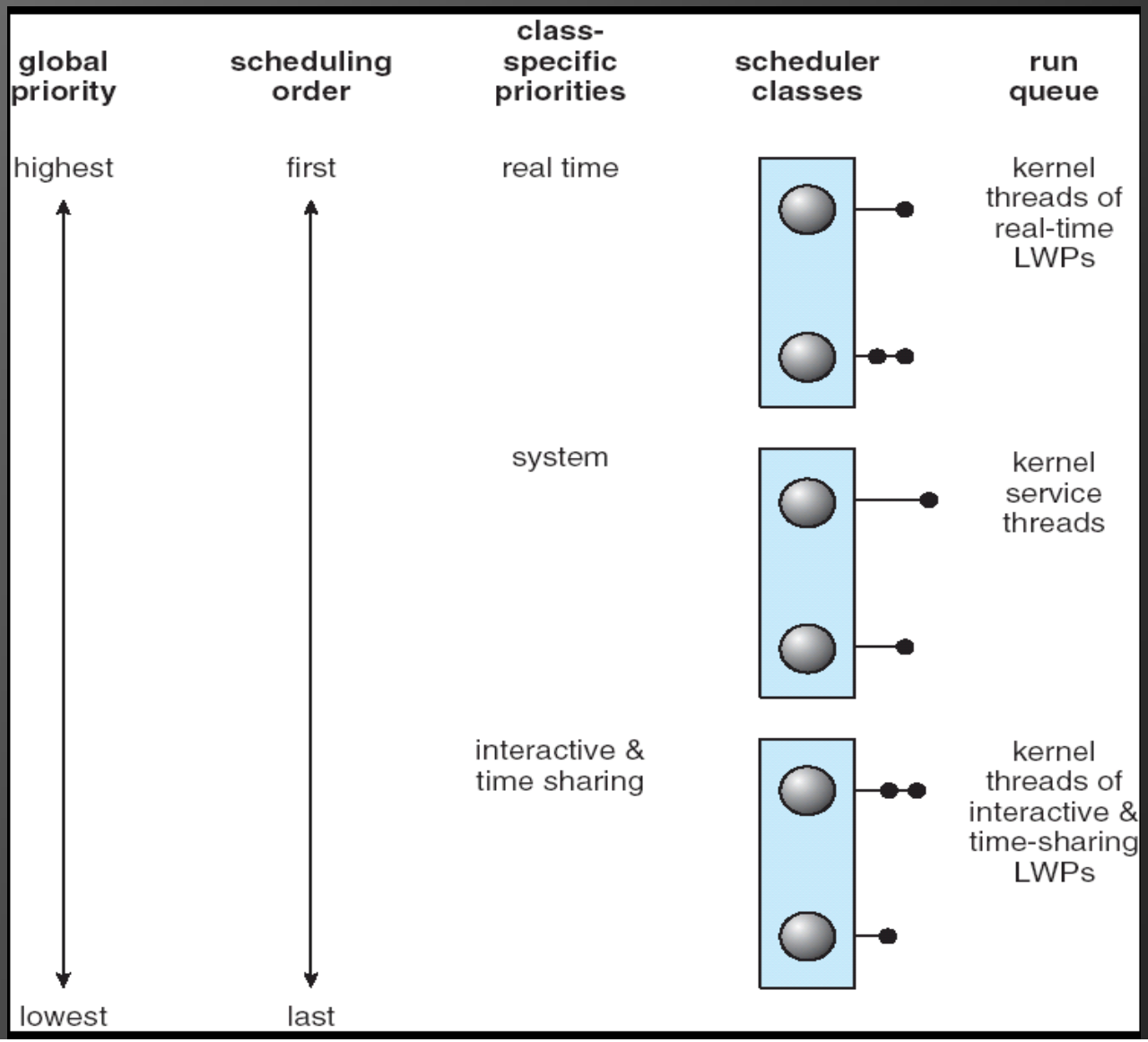
  Q1: RR with Q=4

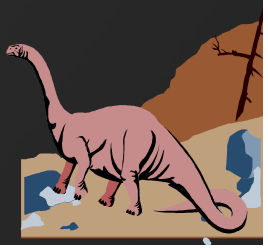
  Q2: RR with Q=10

  Q3: FCFS

Job	BT	AT
A	24	32
B	36	15
C	17	0
D	9	1
E	15	17

Solaris 2 Scheduling





Solaris Dispatch Table

priority	time quantum	time quantum expired	return from sleep
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	58
55	40	45	58
59	20	49	59

Windows XP Priorities

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1