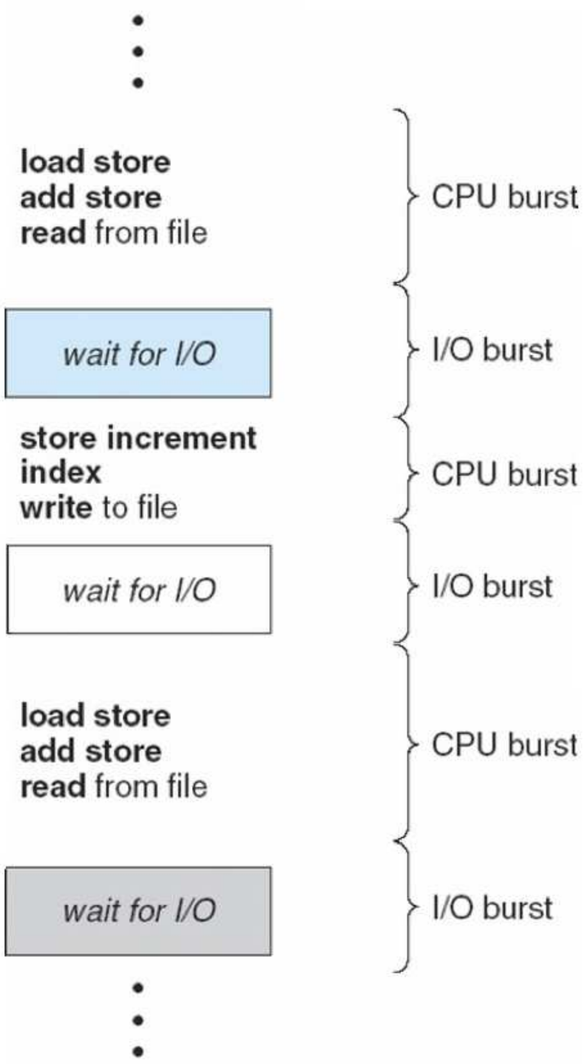
CPU Scheduling

CPU scheduling is the problem of picking a ready process/thread to run

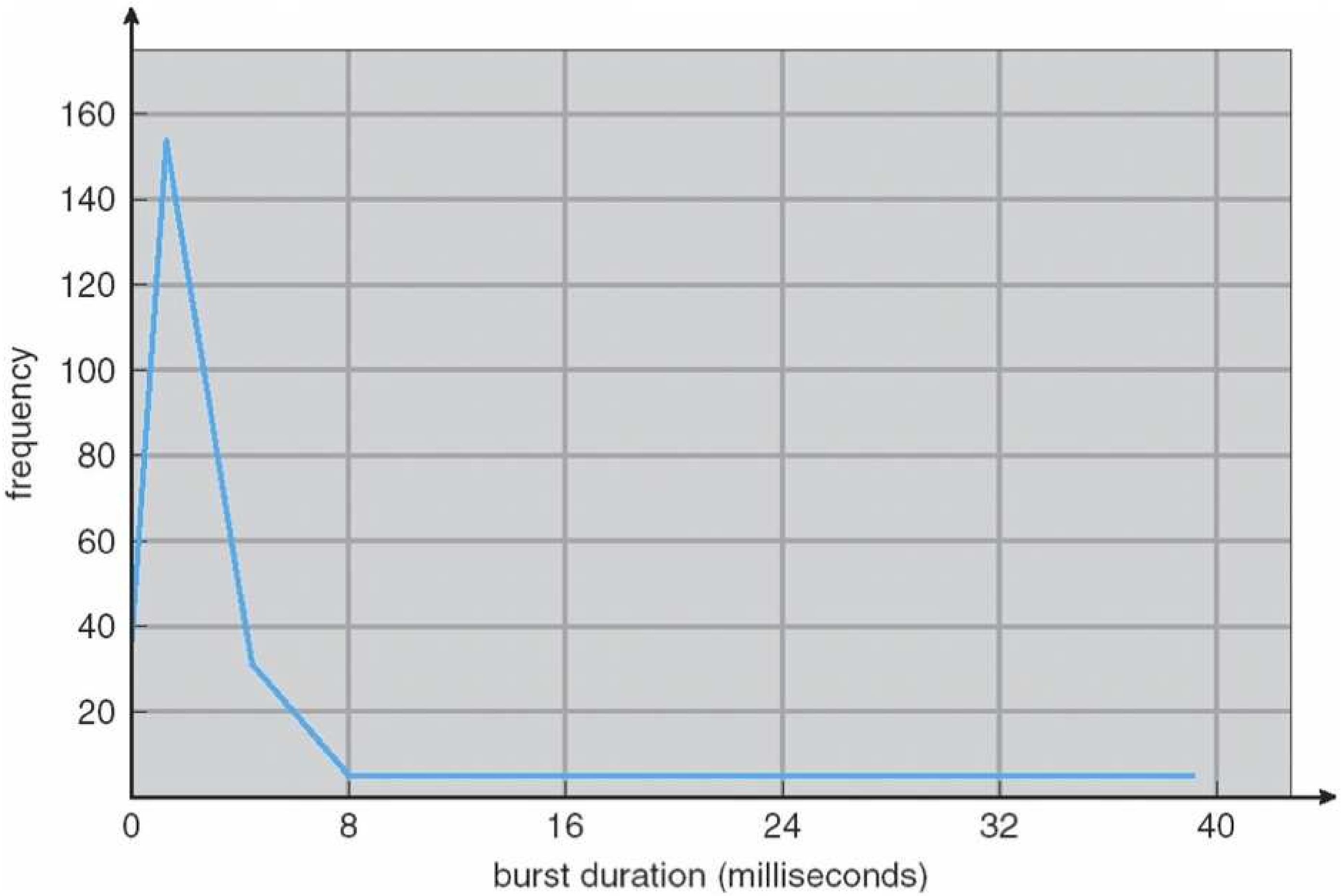
- **Non-preemptive** — process decides when it’s ok to switch
- **Preemptive** — OS decides when to switch

Common strategies exploit the pattern of CPU vs. I/O waiting in a program

Bursts


load store
add store
read from file
CPU burst
wait for I/O
I/O burst
store increment
index
write to file
CPU burst
wait for I/O
I/O burst
load store
add store
read from file
CPU burst
wait for I/O
I/O burst

Bursts


frequency
160
140
120
100
80
60
40
20
0
8
16
24
32
40
burst duration (milliseconds)

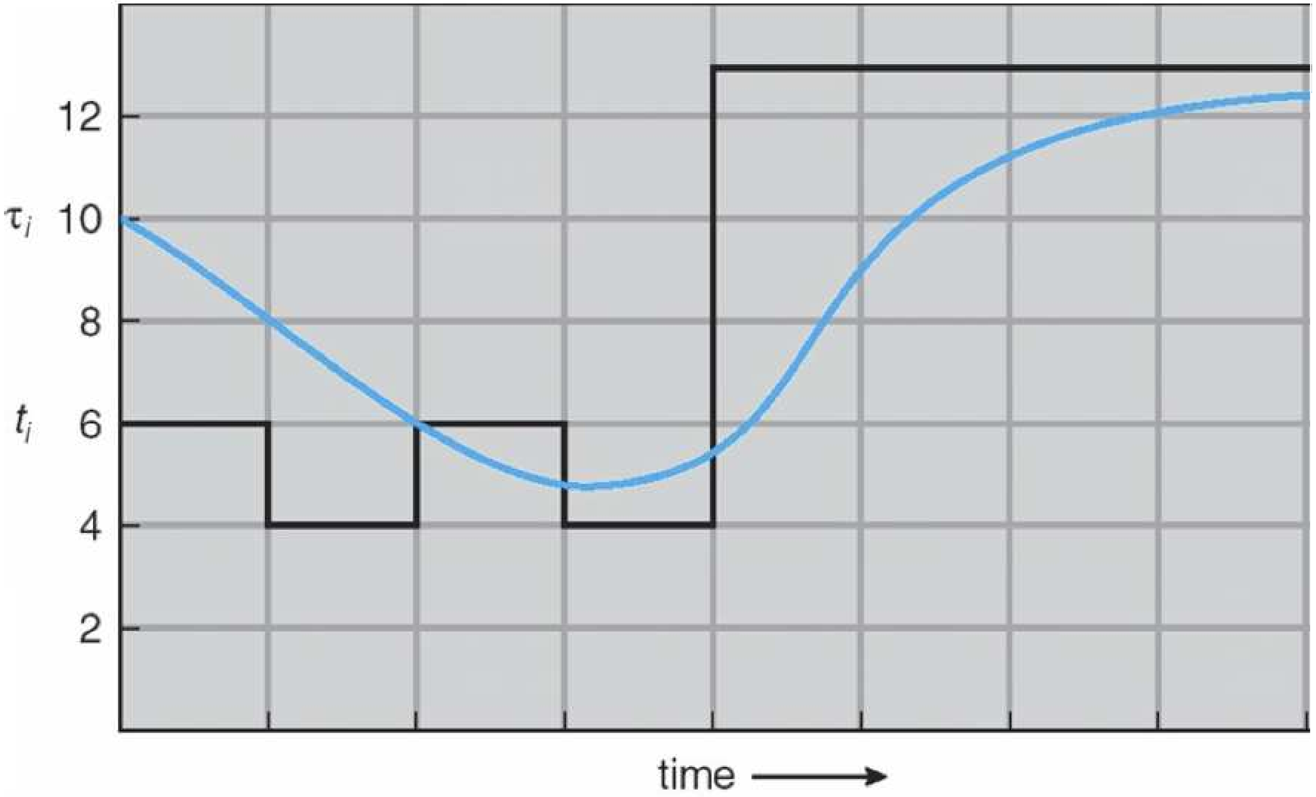
Predicting Bursts

Exponential average:

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$$

- τ_{n+1} = predicted length of burst $n+1$
- t_n = actual length of burst n
- $0 < \alpha < 1$

Predicting Bursts



CPU burst (t_i)		6	4	6	4	13	13	13
"guess" (τ_i)	10	8	6	6	5	9	11	12

Scheduler Inputs

- Available processes/threads
- Bursts (maybe predicted)
- Dispatch latency (i.e., time to switch contexts)

Possible Scheduler Goals

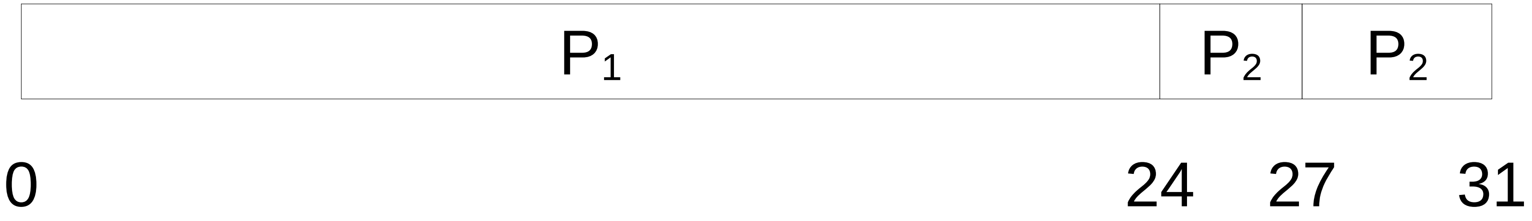
- **CPU utilization** — how busy the CPU stays
- **Throughput** — rate of process completion

- **Turnaround time** — from ready to done for each process
- **Waiting time** — turnaround time minus inherent time
- **Response time** — time from ready to first output

First-Come First-Served (FCFS)

Process	Burst time
P_0	24
P_1	3
P_2	3

Arrive in order P_1 P_2 P_3:

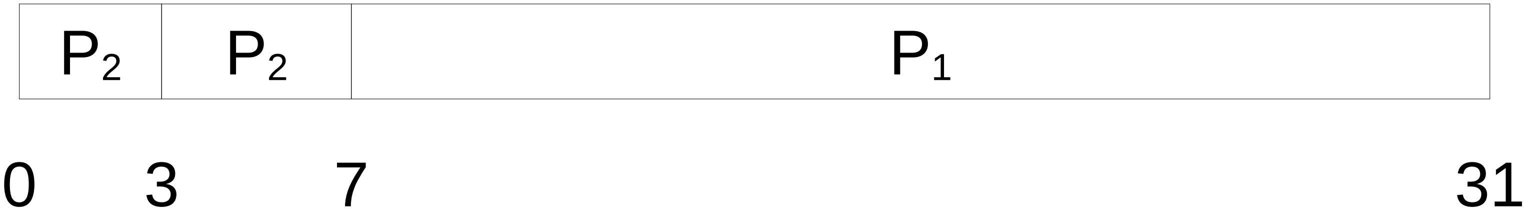


Avg waiting time: (0 + 24 + 27) / 3 = 17

First-Come First-Served (FCFS)

Process	Burst time
P_0	24
P_1	3
P_2	3

Arrive in order P_2 P_3 P_1:



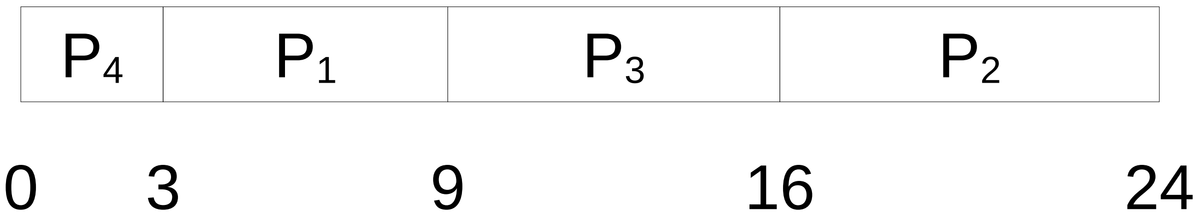
P_2 P_2 P_1

0 3 7 31

Avg waiting time: (0 + 3 + 6) / 3 = 3

Shortest-Job First (SJF)

Process	Burst time
P_0	6
P_1	8
P_2	7
P_3	3



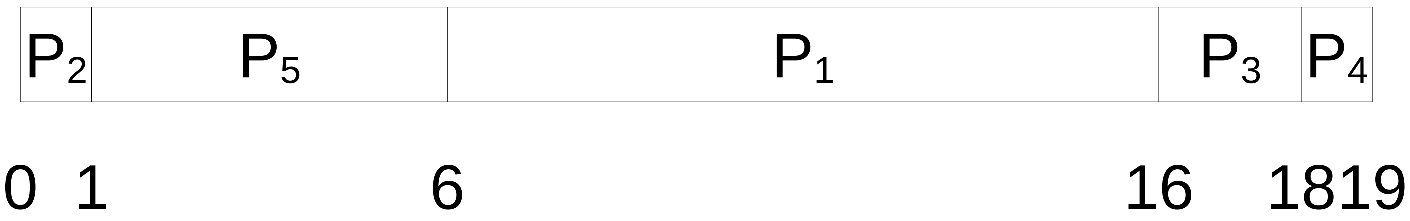
Avg waiting time: (0 3 6 7) / 4 = 4

But how do you know the next burst?

Priority Scheduling

Process	Burst time	Priority
P_0	10	3
P_1	1	1
P_2	2	4
P_3	1	5
P_4	5	2

lower number is higher priority



P_2 P_5 P_1 P_3 P_4

0 1 6 16 1819

Round-Robin (RR)

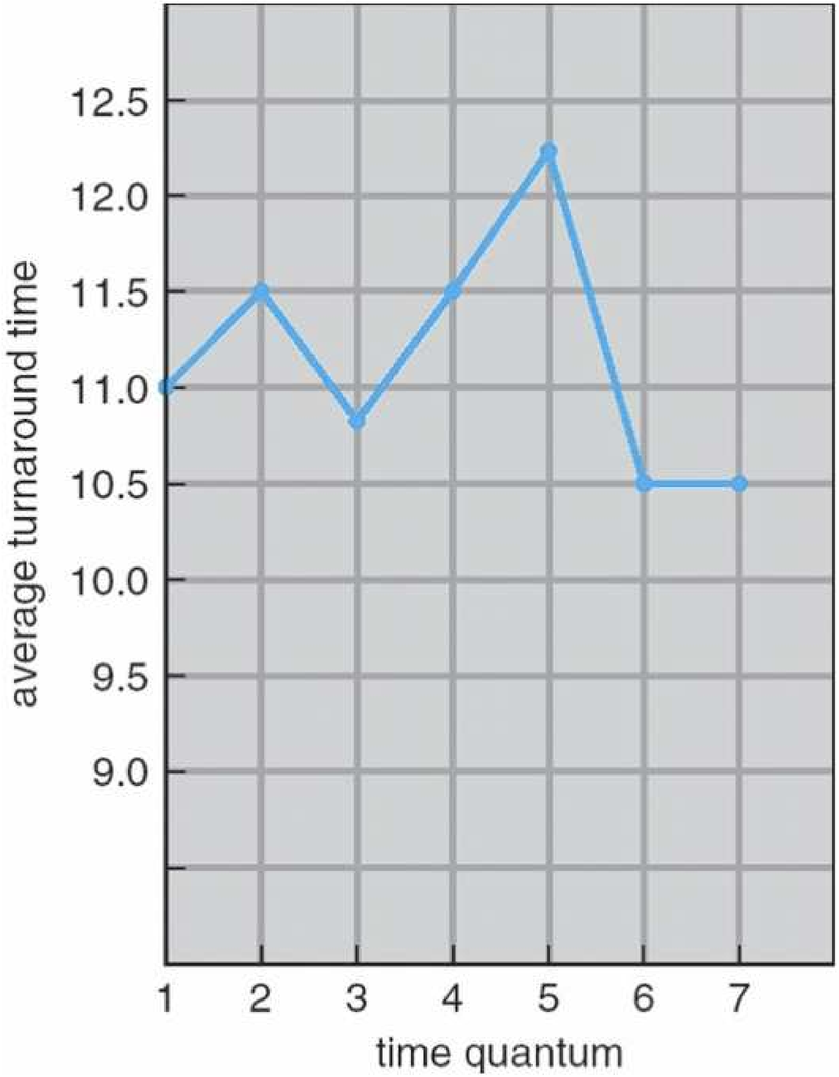
Process	Burst time
P_0	24
P_1	3
P_2	3

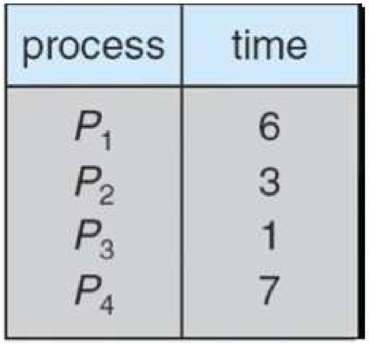
Quantum of 4:

P_1	P_2	P_3	P_1	P_1	P_1	P_1	P_1

0 4 7 10 14 18 22 26 30

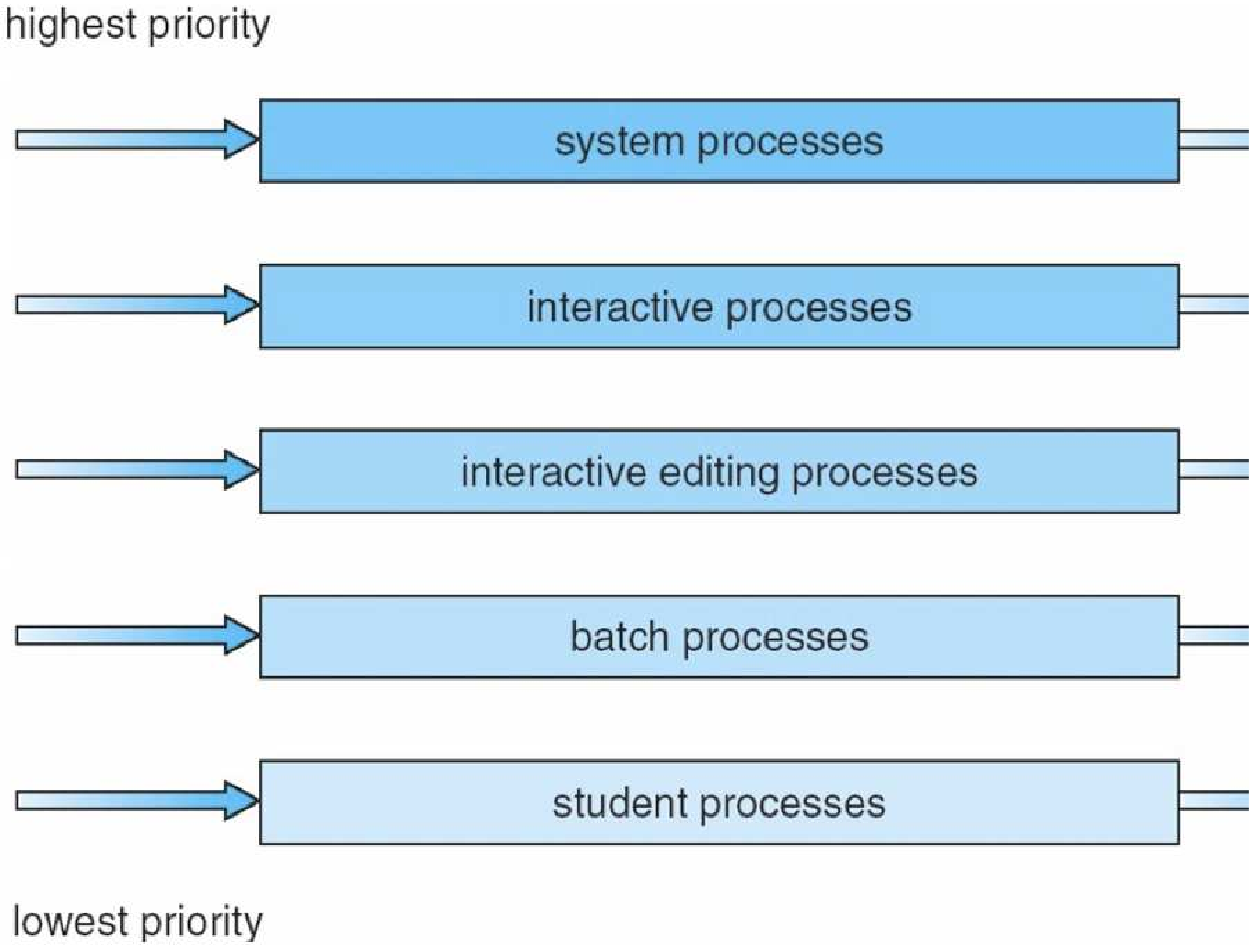
Round-Robin (RR)





process	time
P_1	6
P_2	3
P_3	1
P_4	7

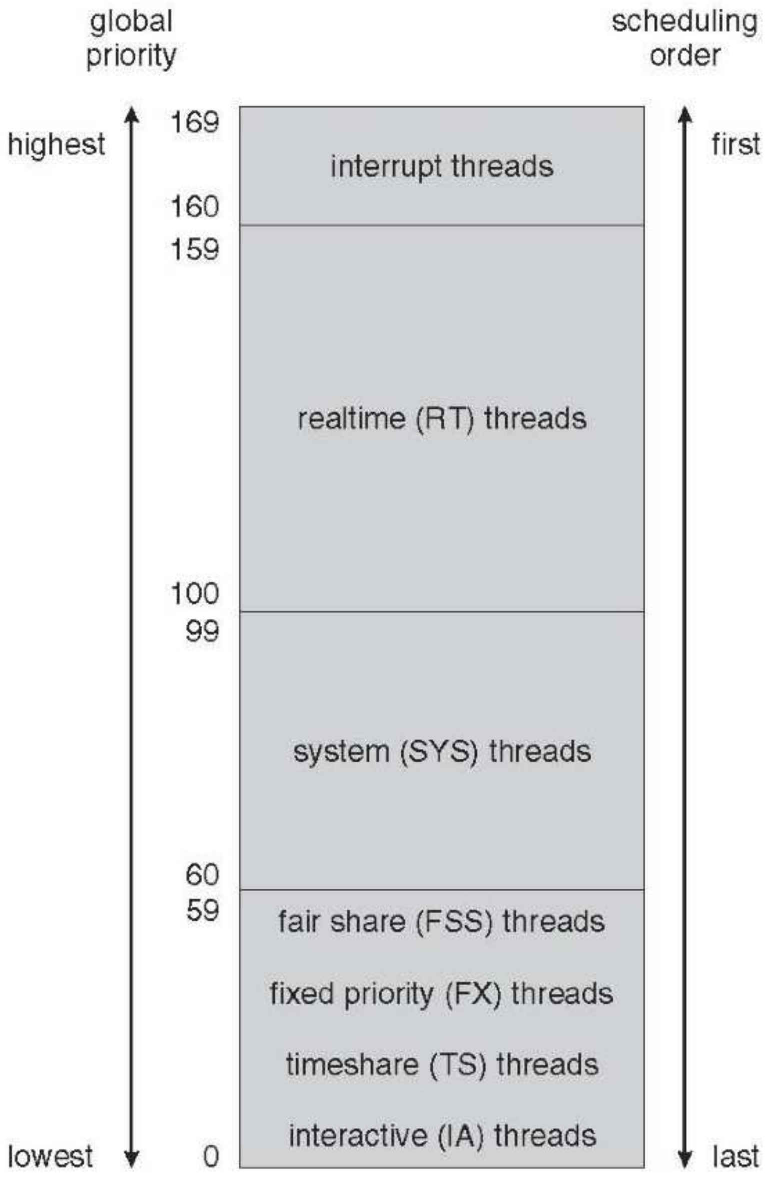
Multilevel Scheduling



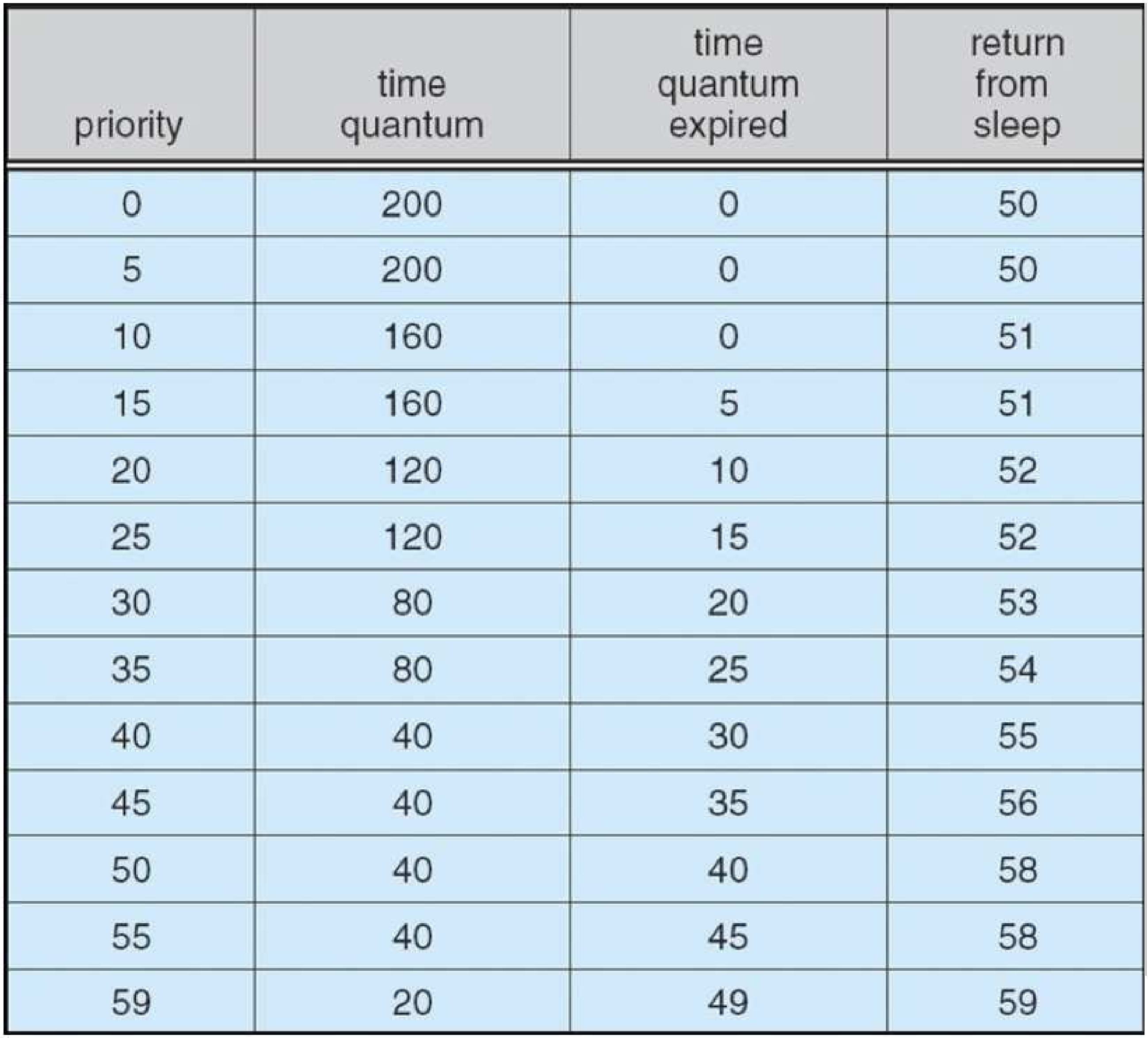
Combinations

- Round-robin within a priority
- Round-robin with priority-based quanta
- Various rules to adjust priority

Solaris

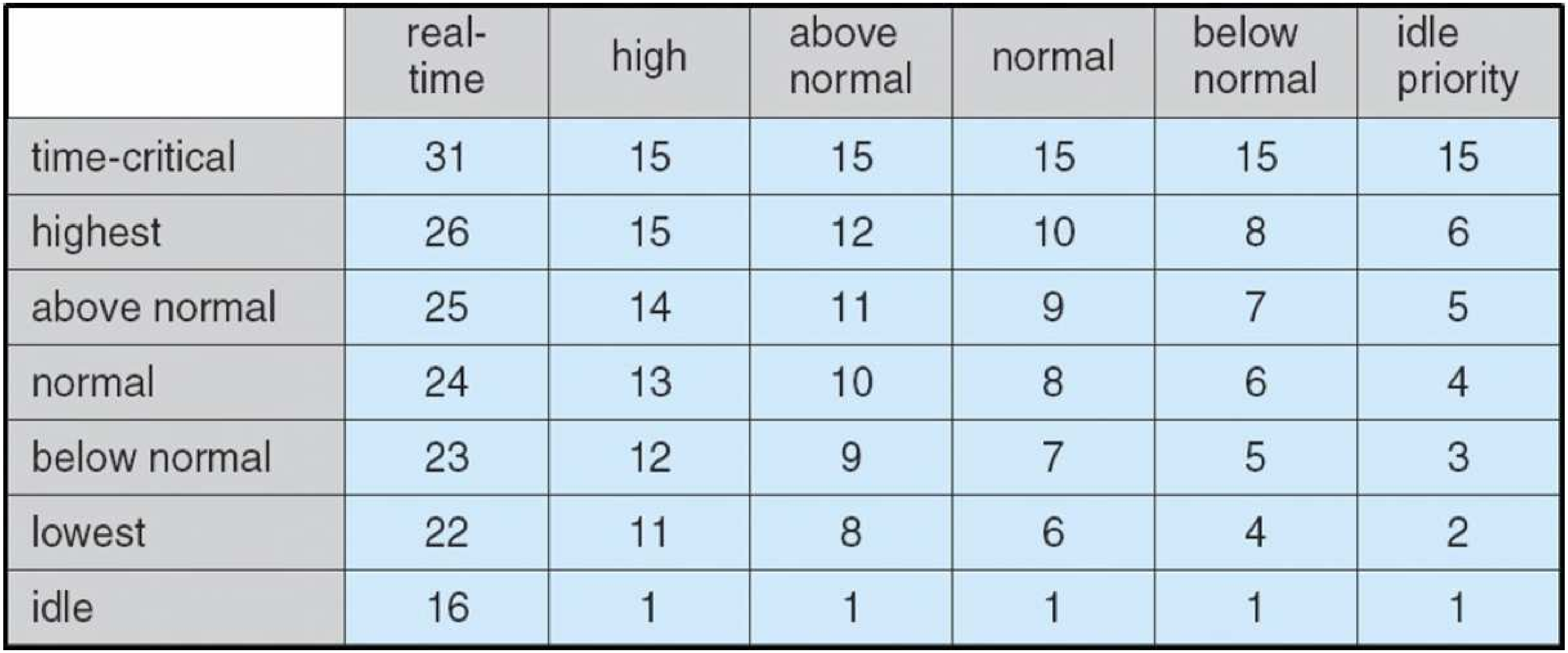

global priority
scheduling order
highest
first
169
interrupt threads
160
159
realtime (RT) threads
100
99
system (SYS) threads
60
59
fair share (FSS) threads
fixed priority (FX) threads
timeshare (TS) threads
interactive (IA) threads
lowest
0
last

Solaris



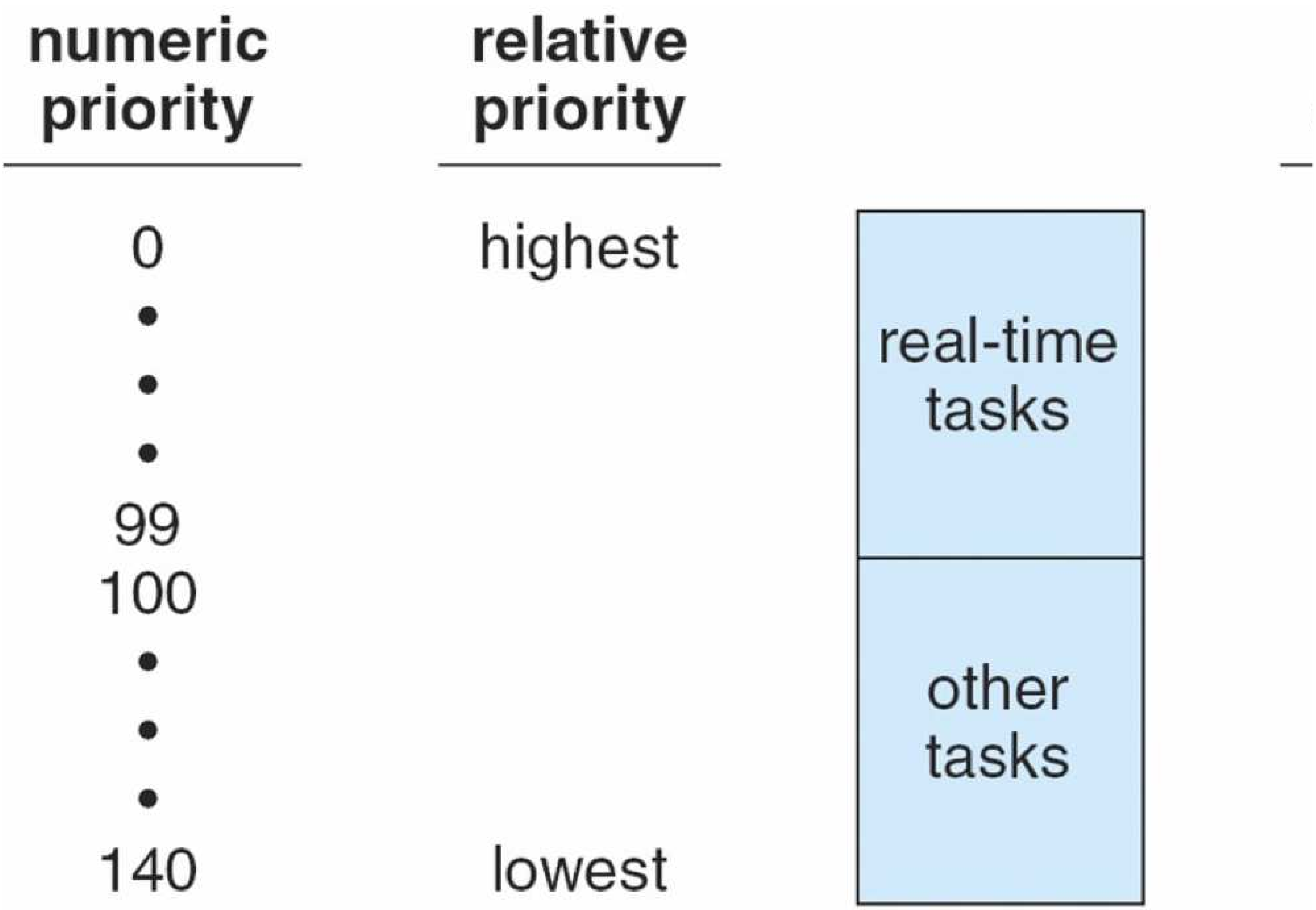
priority	time quantum	time quantum expired	return from sleep
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	58
55	40	45	58
59	20	49	59

Windows



	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

Linux


numeric priority
relative priority
0
highest
real-time tasks
99
100
other tasks
140
lowest

Other Issues

Threads: schedule within or across processes?

Manay OSes support across processes only

Multiple processors: processor affinity
vs. load balancing

Virtualization: scheduler interactions