University of Glasgow | School of Computing Science

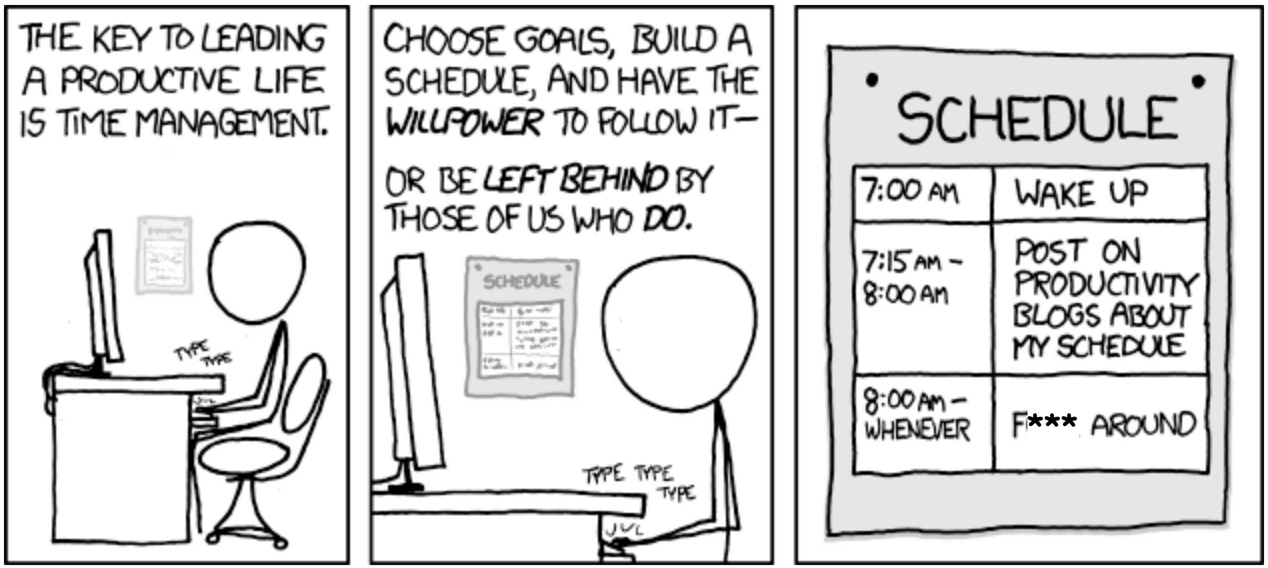
Networks & Operating Systems Essentials

Dr Angelos Marnerides

<*angelos.marnerides@glasgow.ac.uk*>

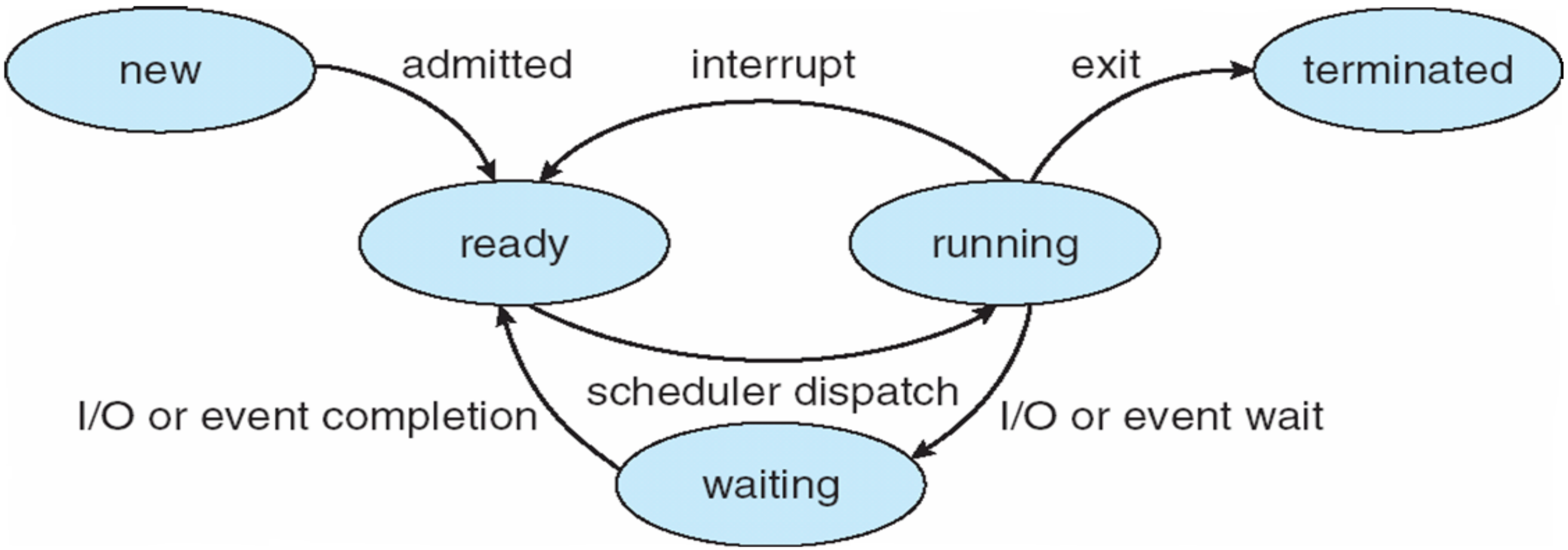
School of Computing Science

Coming up next…



@https://xkcd.com/874/

Diagram of Process States



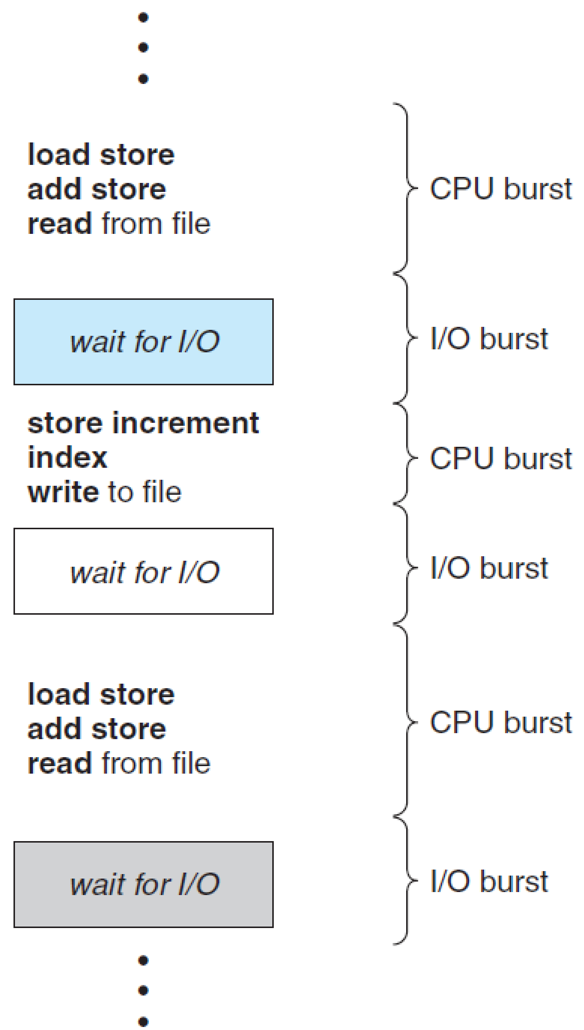
Source: A. Silberschatz, "Operating System Concepts", 9th Ed., 2012.

Process Scheduling

- Key concepts:
  - Multiprogramming: have some process running at all times
  - Time sharing: Switch among processes quickly enough to give the impression of parallel execution
- CPU burst → IO burst cycle

Typical lifecycle of a process

CPU bound?



Source: A. Silberschatz, "Operating System Concepts", 9th Ed., 2012.

IO bound?

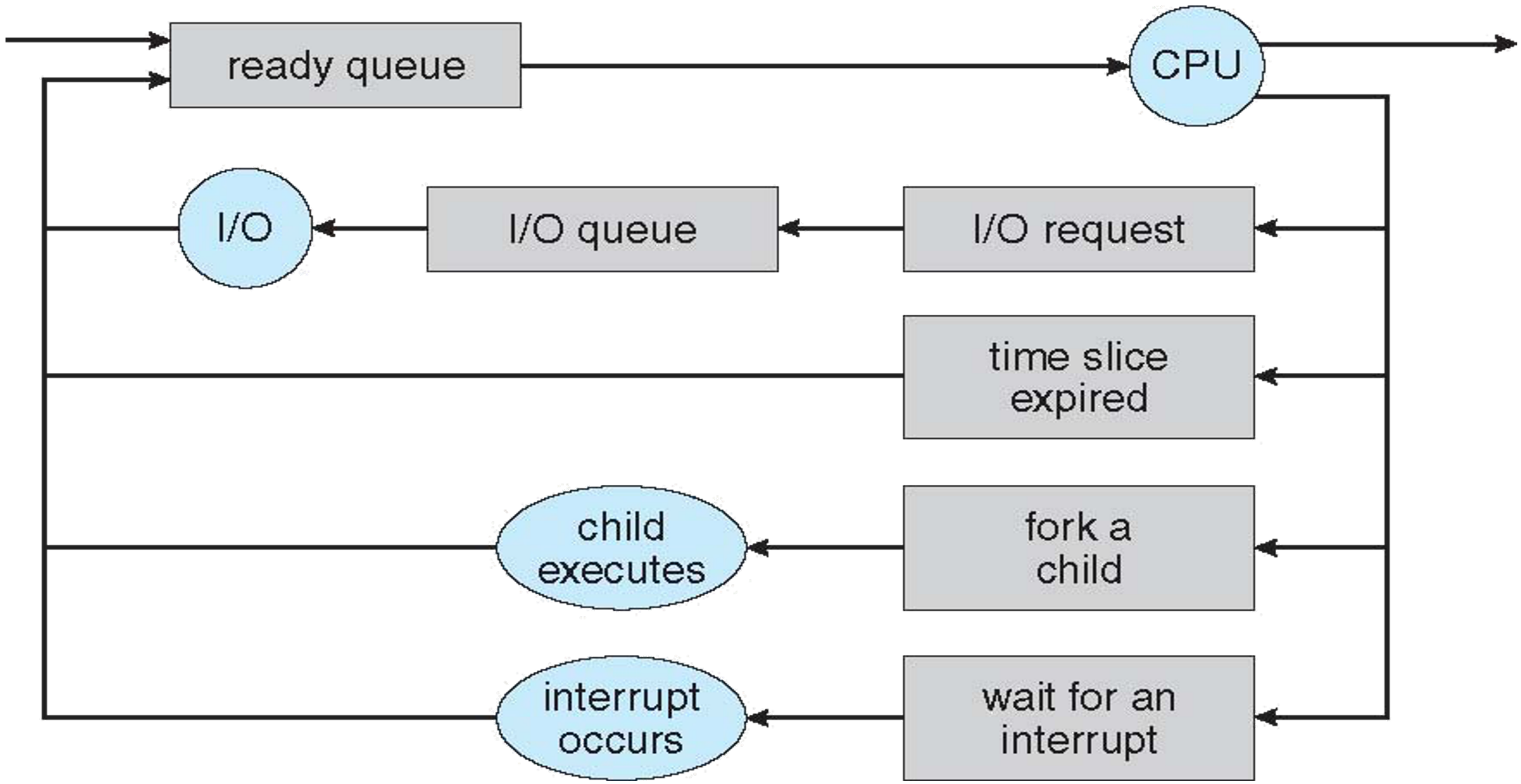
Schedulers

- Long-term scheduler (or job scheduler)
  - Loads programs into memory (i.e., turns a program into a process)
  - Strives for good mix of IO bound and CPU bound processes
- Medium-term scheduler
  - Freezes and unloads (swaps out) a process, to be reintroduced (swapped in) at a later stage
  - E.g., when there is not enough RAM for all processes
- Short-term scheduler (or CPU scheduler)
  - Selects which process to execute next

Process Scheduling

- Scheduling's main priority: Maximize CPU utilization
- Several queues of processes need be maintained:
  - Job queue (all processes in the system)
  - Ready queue (processes in RAM and ready to execute)
  - Device queues (processes waiting on some device; one such queue per device)
  - Processes can migrate among the various queues

Queue-based Representation of Process Scheduling



Source: A. Silberschatz, “Operating System Concepts”, 9th Ed., 2012.

- Grey boxes: queues
- Blue circles: queue servers

CPU Scheduler

- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state (e.g., IO request, sleep, etc.)
  2. Switches from running to ready state (e.g., interrupt occurs)
  3. Switches from waiting to ready (e.g., completion of IO)
  4. Terminates

- Non-preemptive scheduling: Let the process give up (yield) the CPU (e.g., cases 1 and 4)
  - Some definitions will only consider case 4
- Preemptive: The Scheduler decides when a process yields the CPU (e.g., all cases)

Dispatcher

- Gives control of the CPU to the process selected by the short-term scheduler
- Needs to:
  - Switch context (load registers, PCB, etc.)
  - Switch to user mode
  - Jump to the proper instruction in the user program to continue execution
- Dispatch latency?
  - Note: context switch is pure overhead

Criteria for comparing CPU Schedulers

- CPU utilization (more is better)
  - Keep the CPU as busy as possible (40% - 90% a good range)
- Throughput (more is better)
  - Number of processes completed per time unit
- Waiting time (less is better)
  - Total time spent in the READY queue (i.e., loaded but not having the CPU)
- Turnaround time (less is better)
  - Time from submission of a process to its completion
- Response time (less is better)
  - Time from submission of a process till first response produced

CPU Schedulers

- First-Come, First-Served (FCFS or FIFO)
- Priority based (preemptive and non-preemptive)
- Shortest Job First (SJF)
- Shortest Remaining Time First (SRTF)
- Round-Robin (RR)

Running example

Process ID (PID)	Arrival Time	CPU Burst Time
P1	0	6
P2	4	14
P3	5	10

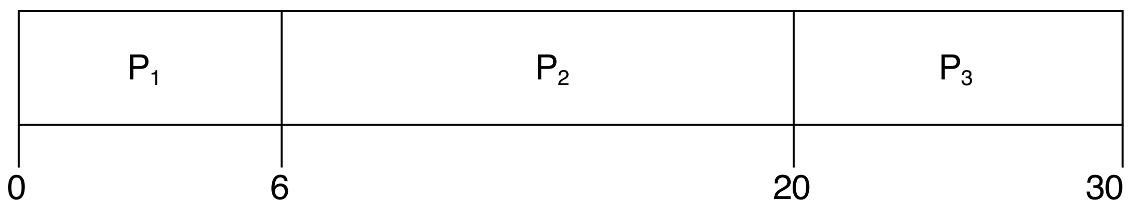
- Consider the above processes
  - Draw the Gantt Chart for the schedule
  - Compute average waiting time, average turnaround time, etc.

FCFS / FIFO

- Algorithm:
  - Processes (or their PCBs) are added to a queue in order of arrival
  - Picks the first process in the queue and executes it
  - If the process needs more time to complete, places it at the end of the queue (i.e., last)
- Very simple to implement
- Average waiting time can be quite high

First-Come, First-Served (FCFS) Scheduling

PID	Arrival Time	CPU Burst Time
P1	0	6
P2	4	14
P3	5	10



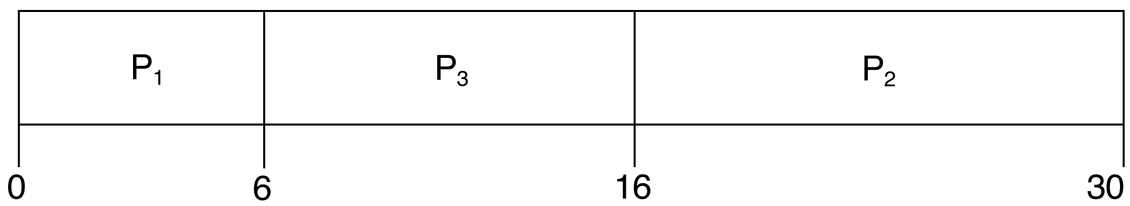
- Execution times:
  - P1: 0 - 6
  - P2: 6 - 20
  - P3: 20 - 30
- Waiting times (= execution - arrival):
  - P1: 0 - 0 = 0
  - P2: 6 - 4 = 2
  - P3: 20 - 5 = 15
- Average waiting time:
  - (0 + 2 + 15)/3 = 5.66…
- Turnaround times (=completion - arrival):
  - P1: 6 - 0 = 6
  - P2: 20 - 4 = 16
  - P3 : 30 - 5 = 25

Shortest Job First (SJF)

- Algorithm:
  - Non preemptive
  - Processes (or their PCBs) are added to a queue, ordered by (next) CPU burst time
  - Picks the first process in the queue and executes it
  - If multiple processes have the same CPU burst time, use FCFS among them
- Fairly simple to implement
- Provable minimum average waiting time
- Unfortunately unrealistic…

Shortest Job First (SJF)

PID	Arrival Time	CPU Burst Time
P1	0	6
P2	4	14
P3	5	10



- Execution times:
  - P1: 0 - 6
  - P2: 16 - 30
  - P3: 6 - 16
- Waiting times (= execution - arrival):
  - P1: 0 - 0 = 0
  - P2: 16 - 4 = 12
  - P3: 6 - 5 = 1
- Average waiting time:
  - (0 + 12 + 1)/3 = 4.33…
- Turnaround times (=completion - arrival):
  - P1: 6 - 0 = 6
  - P2: 30 - 4 = 26
  - P3 : 16 - 5 = 12

Shortest Remaining Time First (SRTF)

- Algorithm:
  - Preemptive version of SJF
  - Processes (or their PCBs) are added to a queue, ordered by remaining CPU burst time
  - Scheduler is also called when new processes arrive
  - Picks the first process in the queue and executes it
  - If multiple processes have the same CPU burst time, use FCFS among them
- Fairly simple to implement
- Provable minimum average waiting time
- Unfortunately unrealistic…

Shortest Remaining Time First (SRTF)

PID	Arrival Time	CPU Burst Time
P1	0	6
P2	4	14
P3	5	10

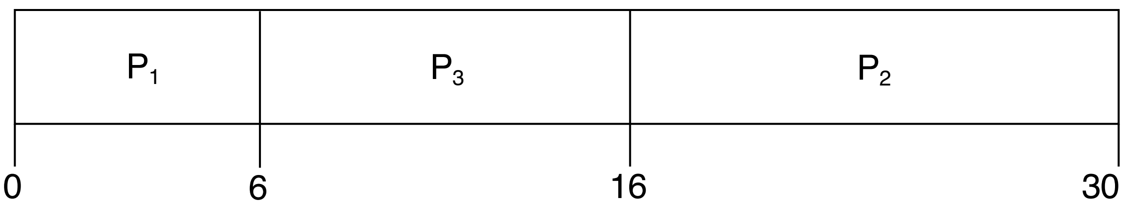
- Execution times:
  - P1: 0 - 6
  - P2: 16 - 30
  - P3: 6 - 16
- Waiting times (= execution - arrival):
  - P1: 0 - 0 = 0
  - P2: 16 - 4 = 12
  - P3: 6 - 5 = 1
- Average waiting time:
  - (0 + 12 + 1)/3 = 4.33…
- Turnaround times (=completion - arrival):
  - P1: 6 - 0 = 6
  - P2: 30 - 4 = 26
  - P3 : 16 - 5 = 12

Non-preemptive Priority Scheduling

- Algorithm:
  - Non-preemptive
  - Processes (or their PCBs) are added to a queue ordered by their priorities (higher priority value = lower priority)
  - If multiple processes have the same priority, use FCFS among them
  - Picks the first process in the queue and executes it
- Fairly simple to implement
- SJF/SRTF practically a special case of priority scheduling
  - If priority = 1/(remaining/next) CPU burst time
- Can lead to indefinite blocking of low priority processes (a.k.a. starvation)
  - Can be solved through “aging”

Non-preemptive Priority Scheduling

PID	Arrival Time	CPU Burst Time	Priority
P1	0	6	2
P2	4	14	3
P3	5	10	1



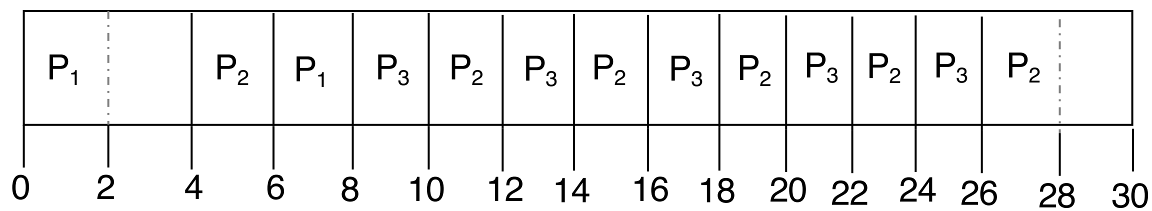
- Execution times:
  - P1: 0 - 6
  - P2: 16 - 30
  - P3: 6 - 16
- Waiting times (= execution - arrival):
  - P1: 0 - 0 = 0
  - P2: 16 - 4 = 12
  - P3: 6 - 5 = 1
- Average waiting time:
  - (0 + 12 + 1)/3 = 4.33…
- Turnaround times (=completion - arrival):
  - P1: 6 - 0 = 6
  - P2: 30 - 4 = 26
  - P3 : 16 - 5 = 12

Round-Robin (RR) Scheduling

- Algorithm:
  - Preemptive version of FCFS
  - Processes (or their PCBs) are added to a queue ordered by their time of arrival
  - Picks the first process in the queue and executes it for up to a time interval (*time quantum/time slice*)
  - If the process needs more time to complete, places it at the end of the queue (i.e., last)
  - If the process ends before its time slice elapses, moves to the next process in the queue
- Quite simple to implement
- May also lead to long waiting times
- Performance depends heavily on the length of the time slice
  - Think of extreme values…

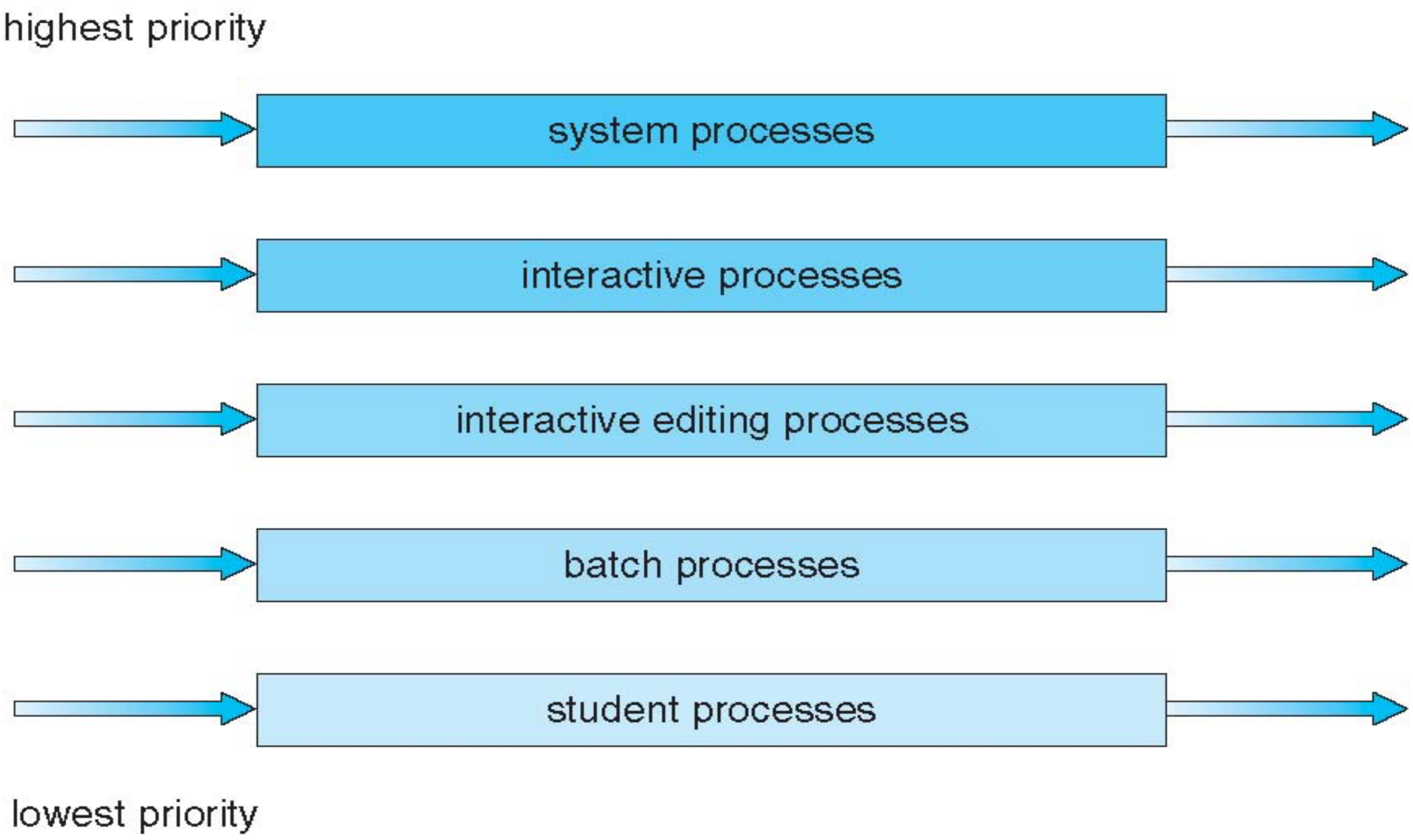
Round-Robin (RR) Scheduling (quantum = 2)

PID	Arrival Time	CPU Burst Time
P1	0	6
P2	4	14
P3	5	10



- Execution times:
  - P1: 0 - 4, 6 - 8
  - P2: 4-6, 10-12, 14-16, 18-20, 22-24, 26-30
  - P3: 8-10, 12-14, 16-18, 20-22, 24-26
- Waiting times (= execution - arrival):
  - P1: (0-0) + (6-4) = 2
  - P2: (4-4) + (10-6) + (14-12) + 18-16) + (22-20) + (26-24) = 12
  - P3: (8-5) + (12 - 10) + (16-14) + (20-18) + (24-22) = 11
- Average waiting time:
  - (2 + 12 + 11)/3 = 11.66…
- Turnaround times (=completion - arrival):
  - P1: 8 - 0 = 8
  - P2: 30 - 4 = 26
  - P3 : 26 - 5 = 21

Multilevel Queue Scheduling



Source: A. Silberschatz, “Operating System Concepts”, 9^{th} Ed., 2012.

Recommended Reading

- Silberschatz, Gagne, Galvin, “Operating Systems Essentials”, chapter 5, sections 5.1, 5.2, 5.3