Operating Systems: CPU Scheduling

Neerja Mhaskar

Department of Computing and Software, McMaster University, Canada

Acknowledgements: Material based on the textbook Operating Systems Concepts (Chapter 5)

Basic Concepts

- OS schedules almost all resources available to the system.
- CPU being the primary resource, scheduling processes to execute on the CPU is central to OS design.
- Maximizing CPU utilization, is feasible as processes alternate using CPU and waiting for I/O

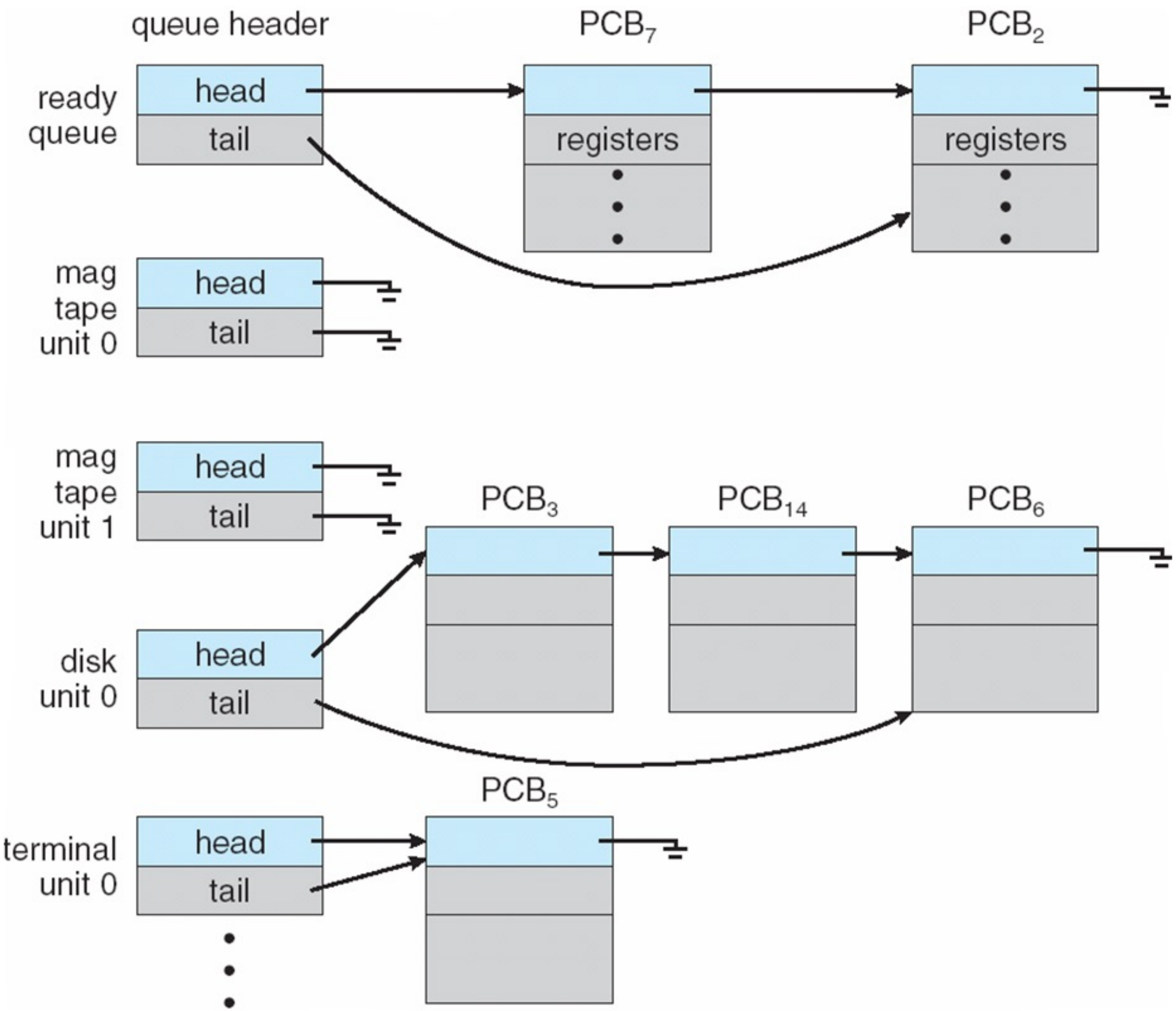
CPU burst and I/O burst

- Process execution consists of a **cycle** of CPU execution (**CPU burst**) and I/O wait (**I/O burst**).
- An I/O-bound program typically has many short CPU bursts.
- A CPU-bound program might have a few long CPU bursts
- This distribution is important in the selection of an appropriate CPU-scheduling algorithm.

Process Scheduling

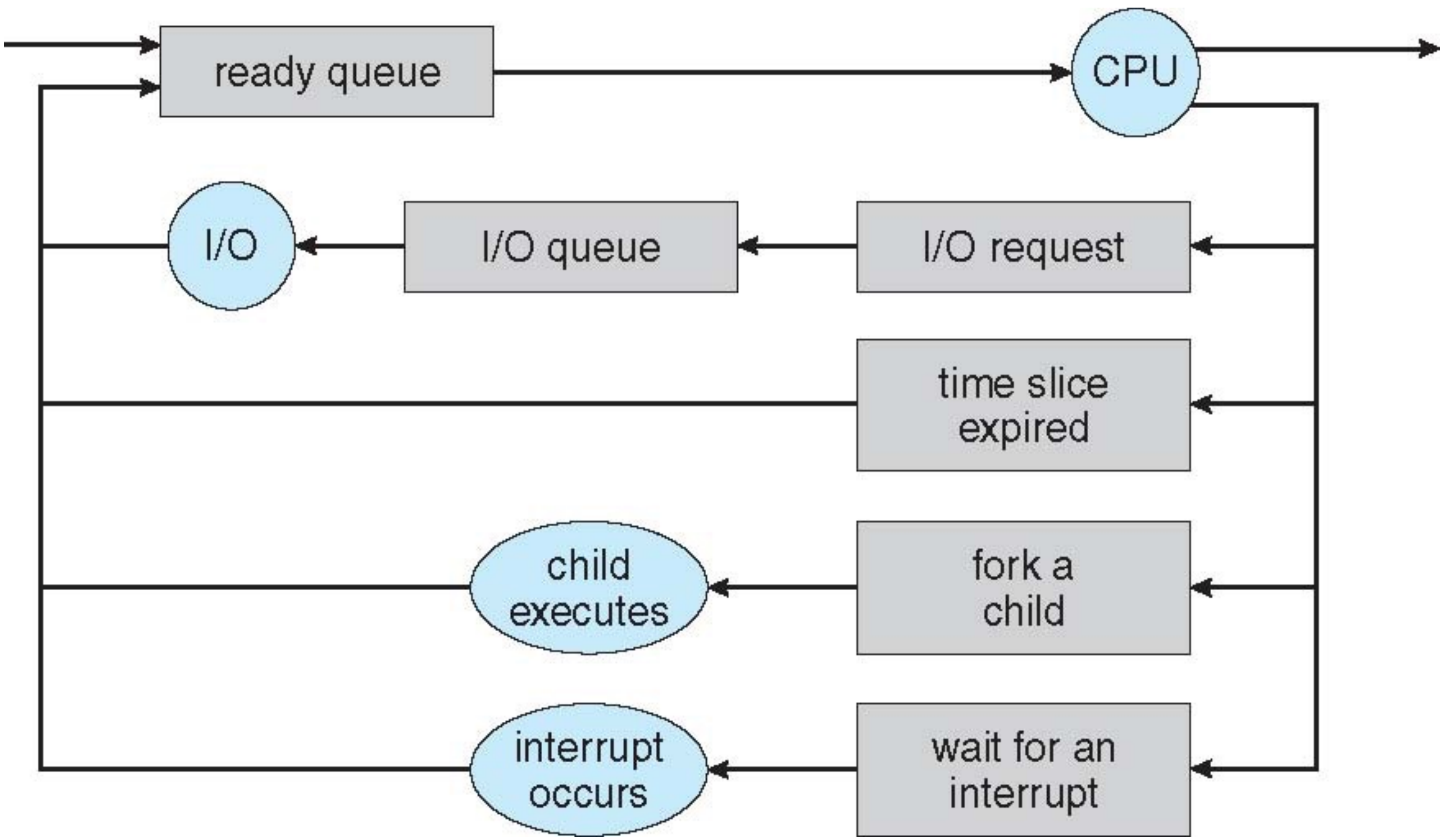
- Operating System maintains **scheduling queues** of processes
  - **Job queue** – set of all processes in the system
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - **Device queues** – set of processes waiting for an I/O device. Usually a separate device queue for each device.
- Processes migrate among the various queues
- **Process scheduler:** selects a process from a queue by implementing appropriate scheduling algorithm.

Ready Queue And Various I/O Device Queues



Representation of Process Scheduling

Queueing diagram represents queues, resources, flows



CPU scheduler

- **CPU scheduler (or Short-term scheduler)** – selects from among the processes in ready queue, and allocates the CPU to one of them (see next slide)
  - Sometimes the only scheduler in a system
  - Short-term scheduler is invoked frequently (milliseconds) $\Rightarrow$ (must be fast)

Non-preemptive Vs. Preemptive

- **Non-preemptive scheduling** - a running process is executed till completion without interruption.

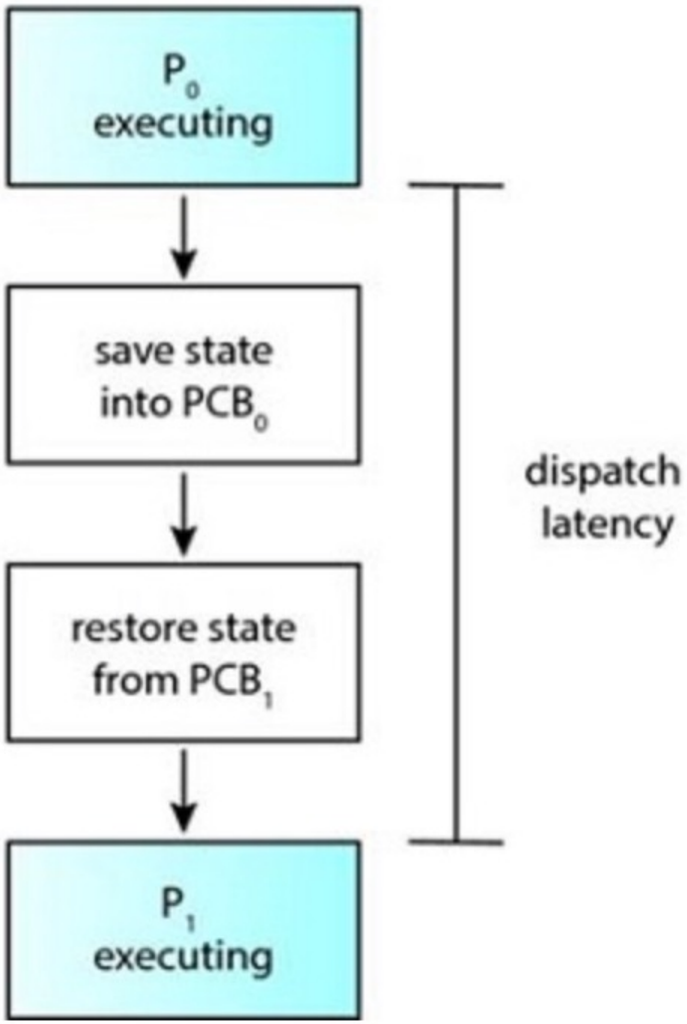
- **Preemptive scheduling** – a running process may be interrupted and moved to the Ready queue by the OS.

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

Dispatcher

- **Dispatch latency** – time taken by the dispatcher to stop one process and start put another process onto the CPU.



Other Scheduler

- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
  - Long-term scheduler is invoked infrequently (seconds, minutes) $\Rightarrow$ (may be slow). Therefore, can use sophisticated scheduling algorithms.
  - Typically, seen on a process intensive systems.
  - The long-term scheduler controls the **degree of multiprogramming and** good ***process mix***

Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – # of processes that complete their execution per time unit
- **Turnaround time** – amount of time to execute a particular process
  - Turnaround time for a process = waiting time + CPU burst time
- **Waiting time** – amount of time a process has been waiting in the ready queue

Scheduling Algorithm Optimization Criteria

- Maximize CPU utilization and throughput
- Minimize turnaround time and waiting time

Gantt Chart

Gantt Chart: is a bar chart that illustrates a particular process schedule, including the start and finish times of each of the participating processes.

Gantt Chart Example

Suppose processes P_1, P_2, P_3 have CPU burst time 24, 3, 3 respectively and arrive in the same order.

The Gantt Chart for the schedule is:

- Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$ msec
- Average waiting time: $(0 + 24 + 27)/3 = 17$ msec
- Turnaround time for a process = waiting time + CPU burst time
- Average turnaround time: $(24 + 27 + 30)/3 = 27$ msec

Scheduling Algorithms

- First Come First Serve (FCFS) Scheduling
- Shortest Job First (SJF) Scheduling
  - Shortest Remaining Time First Scheduling
- Priority Scheduling
- Round Robin Scheduling

First- Come, First-Served (FCFS) Scheduling

- Process requesting the CPU first is allocated the CPU first.
  - The implementation of the FCFS policy can be achieved with a FIFO queue.
- FCFS scheduling algorithm is non-preemptive
- **Disadvantage:** Average CPU waiting time for a process to use CPU is long.

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 $P_1 = 6$ msec; $P_2 = 0$ msec; $P_3 = 3$ msec
- Average waiting time: (6 + 0 + 3)/3 = 3 msec
- **Much better than previous case!**
- **Convoy effect** - short process behind long process
  - Results in lower resource utilization

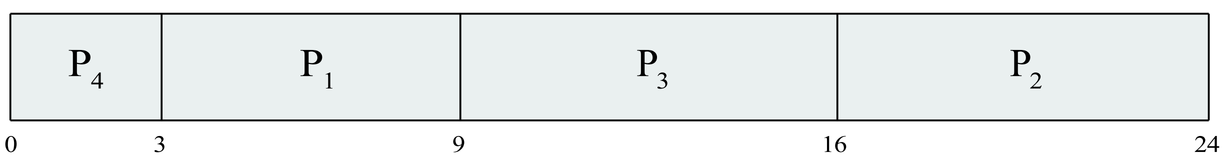
Shortest-Job-First (SJF) Scheduling

- Associated with each process is the length of its next CPU burst
- **SJF** use these lengths to schedule the process with the shortest CPU burst.
  - If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.
- **SJF is optimal** – gives minimum average waiting time for a given set of processes

Example of SJF

Process	CPU Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart



- Average waiting time = (0+ 3 + 9 + 16) / 4 = 7 msec
- Turnaround time = (3+9+16+24)/4 = 13msec

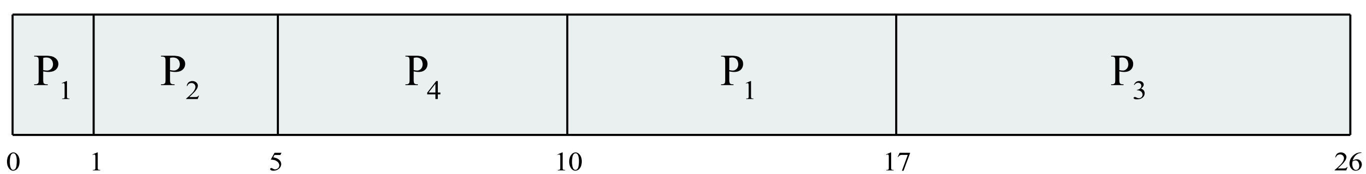
Shortest Remaining Time First

- SJF algorithm can either be preemptive or nonpreemptive
- Preemptive version of SJF is called **shortest-remaining-time-first**
  - At a given time, the process with shortest remaining time is scheduled first.
- Now we add the concepts of varying arrival times and preemption to the analysis

Example of Shortest-remaining-time-first

Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Preemptive SJF Gantt Chart



- Average waiting time = [(10-1)+(1-1)+(17-2)+(5-3)]/4 = 26/4 = 6.5 msec

Priority Scheduling

- A **priority number** (integer) is associated with each process
  - Equal priority process scheduled in FCFS order.
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
- Priority scheduling can either be preemptive or non-preemptive
- Problem ≡ **Starvation** – low priority processes may never execute
  - **Solution** ≡ **Aging** – as time progresses increase the priority of the process

Example of Priority Scheduling

Process	Burst Time(ms)	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- Priority scheduling Gantt Chart

P_2	P_5	P_1	P_3	P_4

- Average waiting time = (0 + 1 + 6 + 16 + 18)/5 = 8.2 ms (milliseconds)

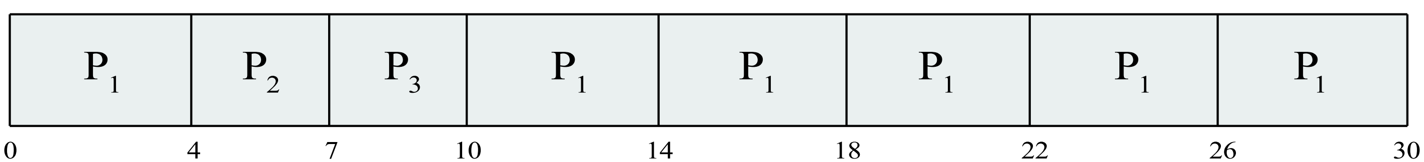
Round Robin (RR)

- Similar to FCFS scheduling, but preemption is added to enable the system to switch between processes.
- Processes are scheduled on FCFS basis from the ready queue, where each process gets a small unit of CPU time (**time quantum** *q*)
- After this time has elapsed, the process is preempted and added to the end of the ready queue.
- Timer interrupts every quantum to schedule next process
- Preemptive Scheduling

Example of RR with Time Quantum = 4

Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



- Typically, higher average *turnaround time* than SJF, but better *response time.*

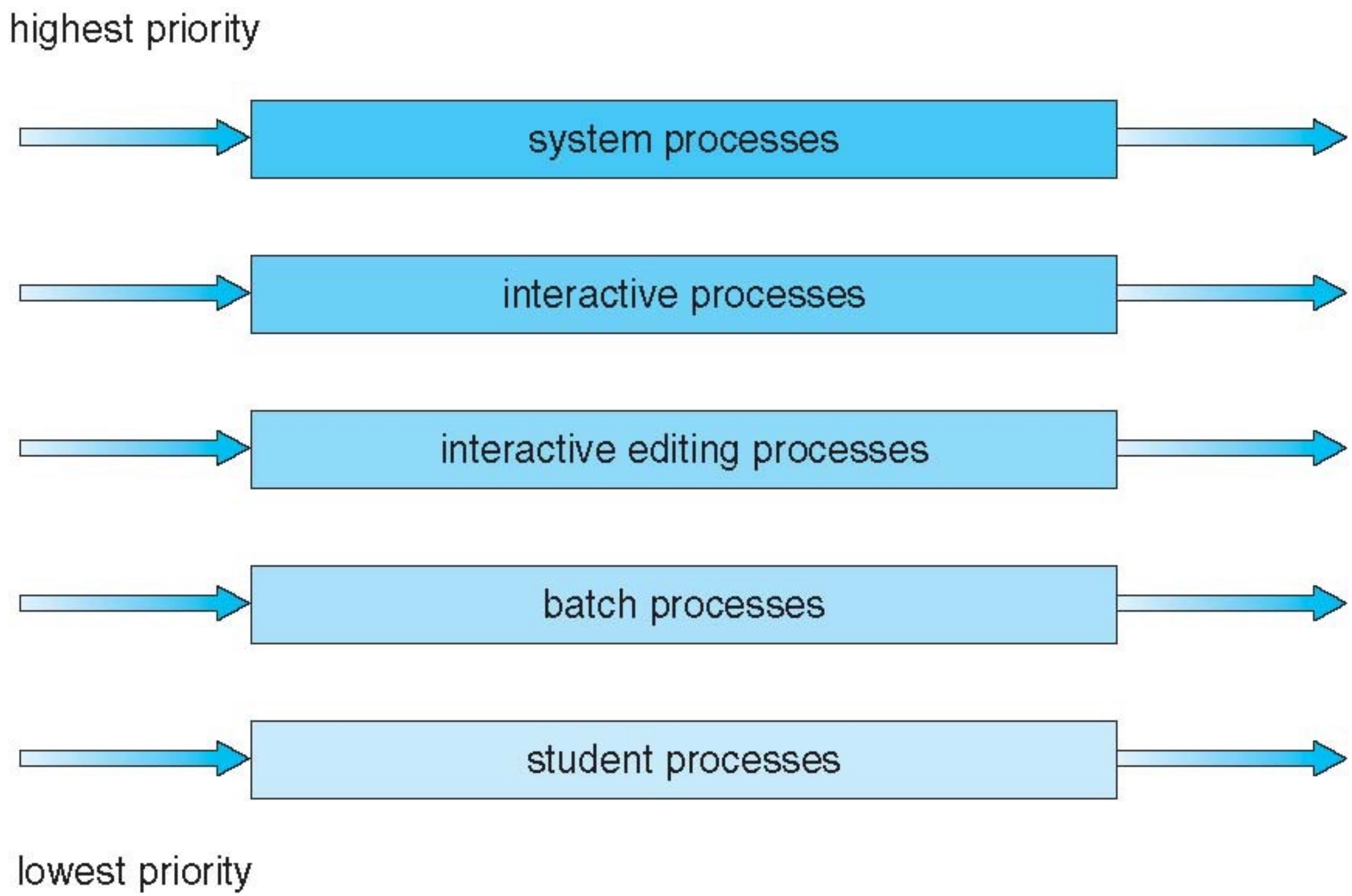
Round Robin Performance

- q large $\Rightarrow$ FIFO
- q must be large with respect to context switching time, otherwise overhead is too high.
  - q is usually 10 to 100 milli seconds and context switch < 10 micro second.

Multilevel Queue and Multilevel Feedback Queue Scheduling

- **Multilevel Queue Scheduling:** Ready queue is partitioned into various separate queues. Process resides permanently in a given queue.
  - Each queue has its own scheduling algorithm
  - Scheduling must be done between the queues:
  - ***Fixed priority preemptive scheduling:*** Serve processes of highest priority first.
    - *Possibility of starvation*.
  - ***Time slice*** – each queue gets a certain amount of CPU time which it can schedule amongst its processes.

Multilevel Queue Scheduling



Multilevel Feedback Queue Scheduling

Multilevel Feedback Queue Scheduling - Disadvantage

- Inflexible – as processes are permanently assigned to a given queue.

- In contrast, in **Multilevel Feedback Queue Scheduling** a process can move between the various queues.
  - Process in low priority queue can be moved to high priority queue, and aging can be implemented this way.

Thread Scheduling

- When threads are supported by the kernel, threads scheduled not processes
- To run on CPU user level threads must be mapped to an associated kernel level thread.
- POSIX Pthreads allows setting scheduling parameters during thread creation.