Operating Systems

6. CPU: Scheduling (1)

Hyunchan, Park

http://oslab.chonbuk.ac.kr

Division of Computer Science and Engineering

Chonbuk National University

Contents

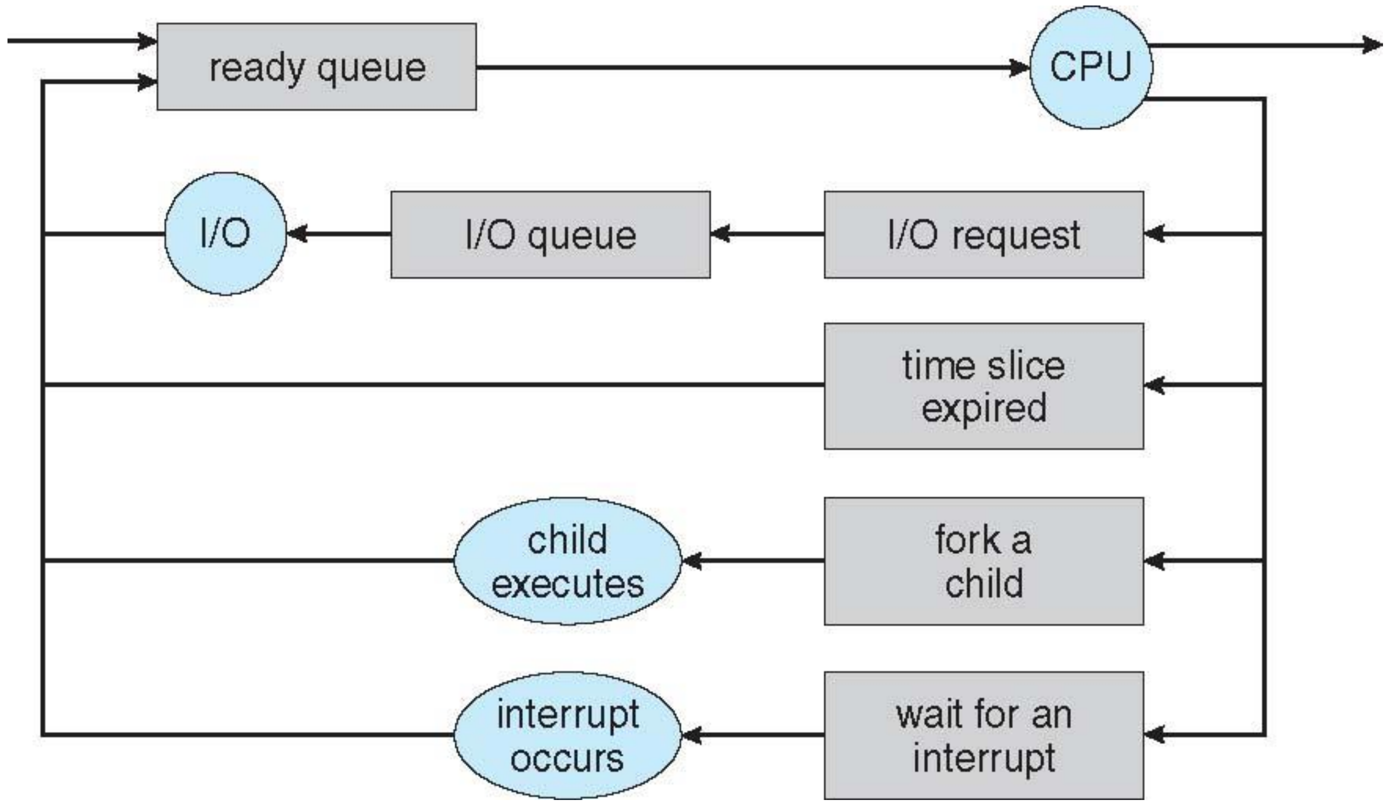
- Basic Concepts
  - Process scheduling
  - CPU and I/O bursts
  - Histogram of CPU-burst times
  - Schedulers: short, mid, and long-term
- Scheduling Criteria and Algorithms
  - First-Come, First-Served Scheduling
  - Shortest-Job-First Scheduling
  - Priority Scheduling
  - Round-Robin Scheduling
  - Multilevel Queue Scheduling
  - Multilevel Feedback Queue Scheduling

Basic Concepts: Process Scheduling

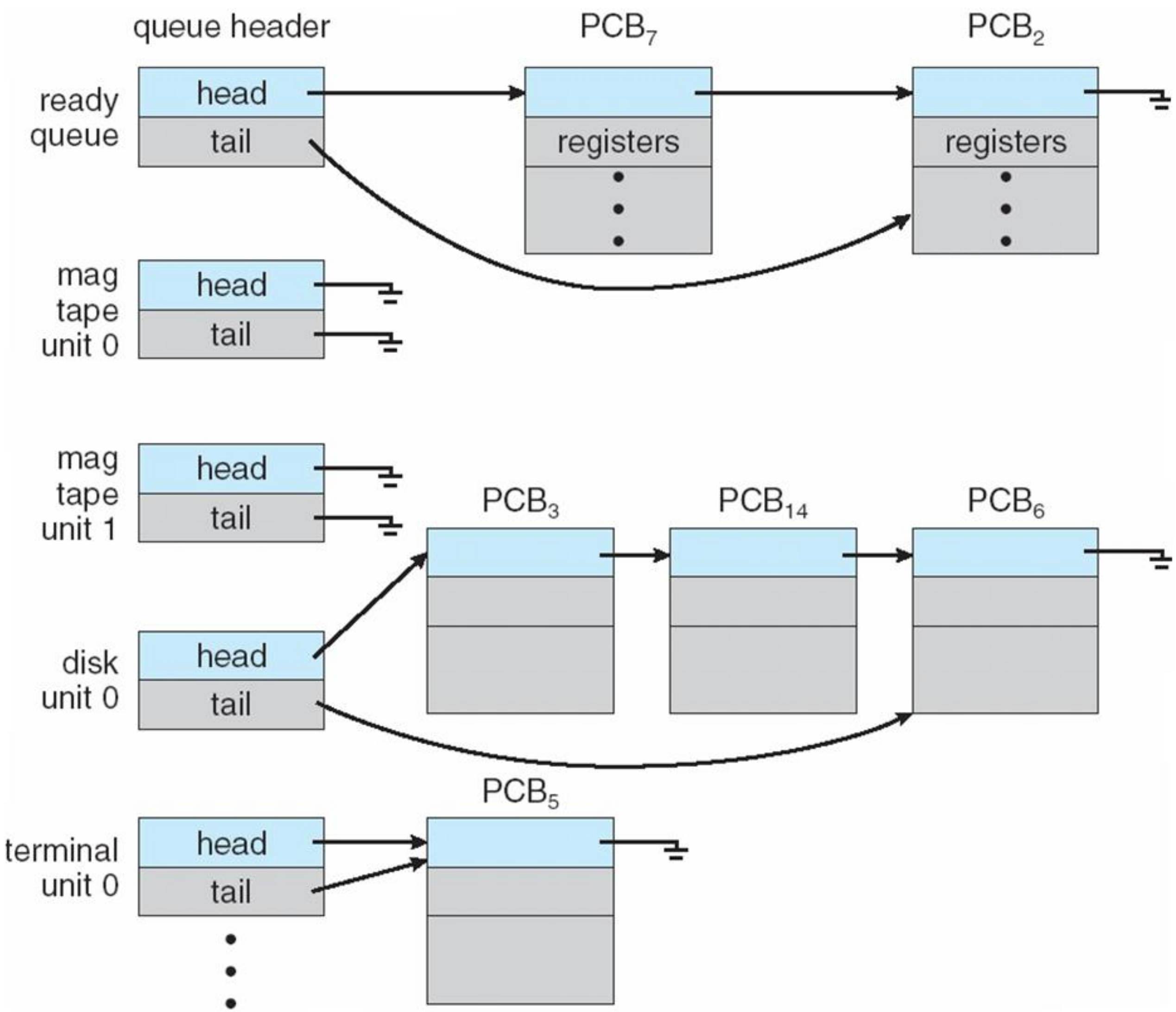
- CPU scheduling: "How to allocate CPU for processes?"
  - To maximize CPU use, quickly switch processes onto CPU for time sharing
- Process scheduler selects among available processes for next execution on CPU
- Maintains scheduling queues of processes
  - Job queue – set of all processes in the system
  - Ready queue – set of all processes residing in main memory, ready and waiting to execute
  - Wait queue – set of processes waiting for an I/O device (or device queue)
  - Processes migrate among the various queues

Representation of Process Scheduling

- Queueing diagram represents queues, resources, flows

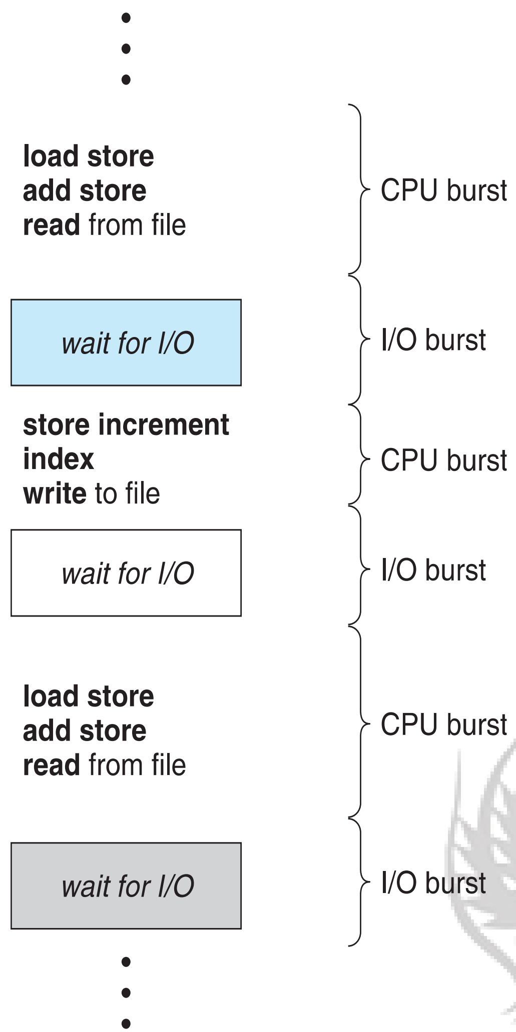


Ready Queue And Various I/O Device Queues



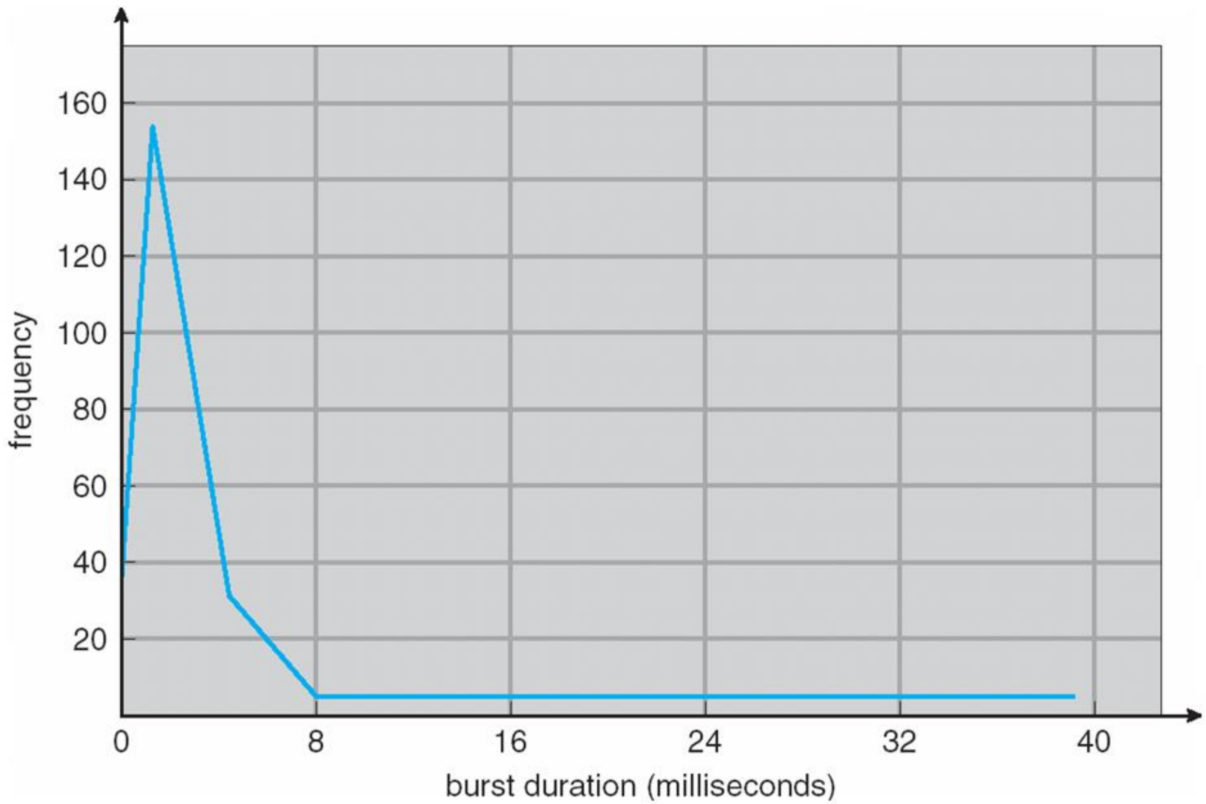
Basic Concepts: Process Scheduling

- Maximum CPU utilization obtained with multiprogramming
- CPU–I/O Burst Cycle: Process execution consists of a **cycle** of CPU execution and I/O wait
- **CPU burst** followed by **I/O burst**
  - CPU burst distribution is of main concern
- Processes can be described as either:
  - I/O-bound process – spends more time doing I/O than computations, many short CPU bursts
  - CPU-bound process – spends more time doing computations; few very long CPU bursts



Histogram of CPU-burst Times

- Common characteristics
  - A large number of short CPU bursts
  - A small number of long CPU bursts

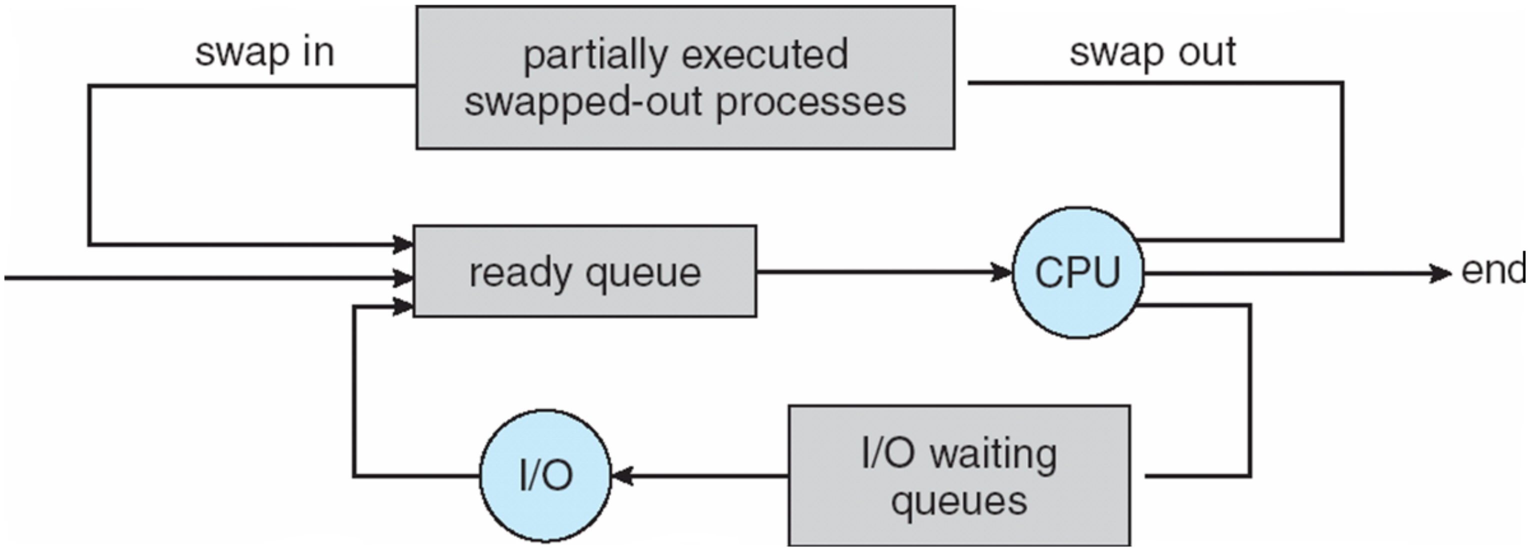


Schedulers

- Short-term scheduler (or CPU scheduler) – selects which process should be executed next and allocates CPU
  - Sometimes the only scheduler in a system
  - Short-term scheduler is invoked frequently (milliseconds) $\Rightarrow$ (must be fast)
- 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)
  - The long-term scheduler controls the degree of multiprogramming
- Long-term scheduler strives for good process mix
  - Among CPU and I/O-bound processes

Addition of Medium Term Scheduling

- Medium-term scheduler can be added if degree of multiple programming needs to decrease
  - In other words, when the memory is not enough
  - Remove process from memory, store on disk, bring back in from disk to continue execution: swapping



Short-term Scheduler (CPU Scheduler)

- Short-term scheduler selects from among the processes in read y queue, and allocates the CPU to one of them
  - Queue may be ordered in various ways

- 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

Non-preemptive and preemptive schedulers

- Non-preemptive (비선점형)
  - Scheduling under 1 and 4
  - OS cannot control the executing process

- All other scheduling is preemptive (선점형)
  - OS can control or interrupt the executing process
  - Issue: synchronization
    - Consider access to shared data
    - Consider preemption while in kernel mode
    - Consider interrupts occurring during crucial OS activities

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
- **Dispatch latency** – time it takes for the dispatcher to stop one process and start another running
  - Context switching overhead

Criteria and Algorithms

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 (waiting time is included)
- **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, not output (for time-sharing environment)

Scheduling Algorithm Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

- It’s impossible to satisfy all the criteria in the singular scheduler
- Only focuses on the main requirement depend on the system
  - Super computer: CPU utilization
  - Main frame or work station: Throughput, turnaround time
  - Personal computer: Response time

Scheduling Algorithms

- First-Come, First-Served Scheduling
- Shortest-Job-First Scheduling
- Priority Scheduling
- Round-Robin Scheduling
- Multilevel Queue Scheduling
- Multilevel Feedback Queue Scheduling

First- Come, First-Served (FCFS) Scheduling

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
- Much better than previous case
- **Convoy effect** - short process behind long process
  - Consider one CPU-bound and many I/O-bound processes

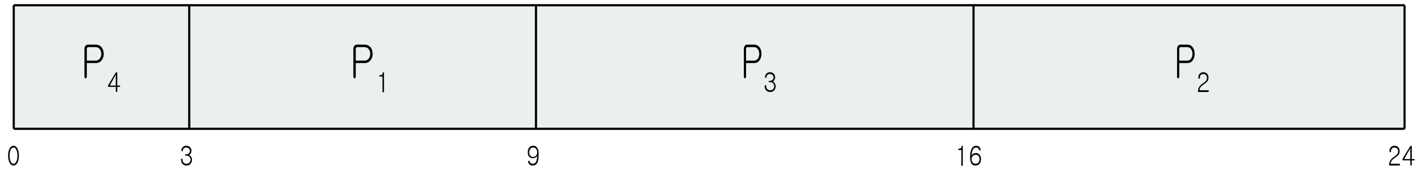
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst
  - Use these lengths to schedule the process with the shortest time
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request
  - Could ask the user

Example of SJF

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

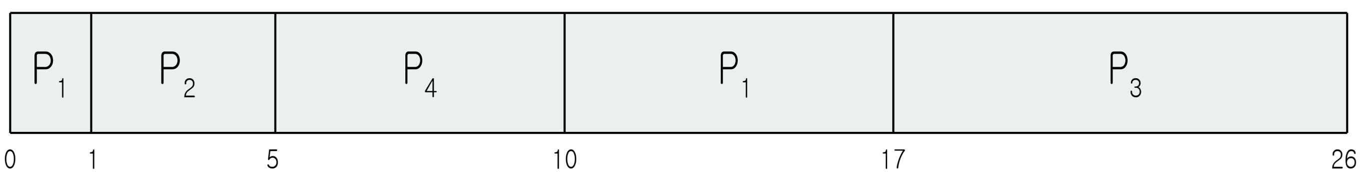
- SJF scheduling chart



- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

Example of Shortest-remaining-time-first

- Now we add the concepts of varying arrival times and preemption to the analysis
  - Process Arrival Time Burst Time
  - P1 0 8
  - P2 1 4
  - P3 2 9
  - P4 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
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
- SJF is priority scheduling where priority is the inverse of CPU burst time
- 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	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

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

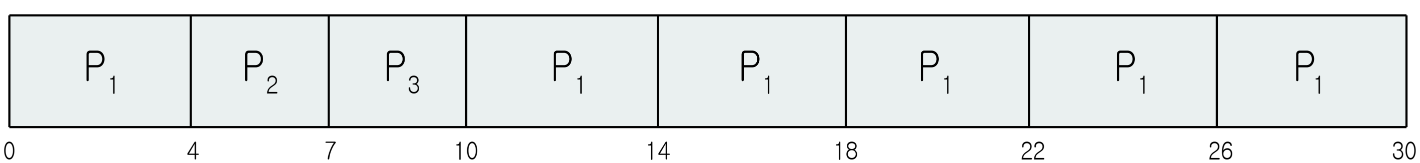
Round Robin (RR)

- Each process gets a small unit of CPU time, usually 1-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue
  - Time quantum: the smallest unit of allocation (e.g. 1 ms)
    - H/W timer interrupts every quantum to enter the kernel mode, and kernel schedules next process if it is necessary
  - Time slice: allocated (or allowed to use CPU) time for a process in a round (e.g. TS = 10 ms = 10 of time quantum)
- Performance
  - *Time slice* large ⇒ FIFO (high performance due to minimal context switches)
  - *Time slice* small ⇒ High responsiveness but low performance
  - *Time slice* must be large with respect to context switch, otherwise overhead is too high

Example of RR with Time Slice = 4

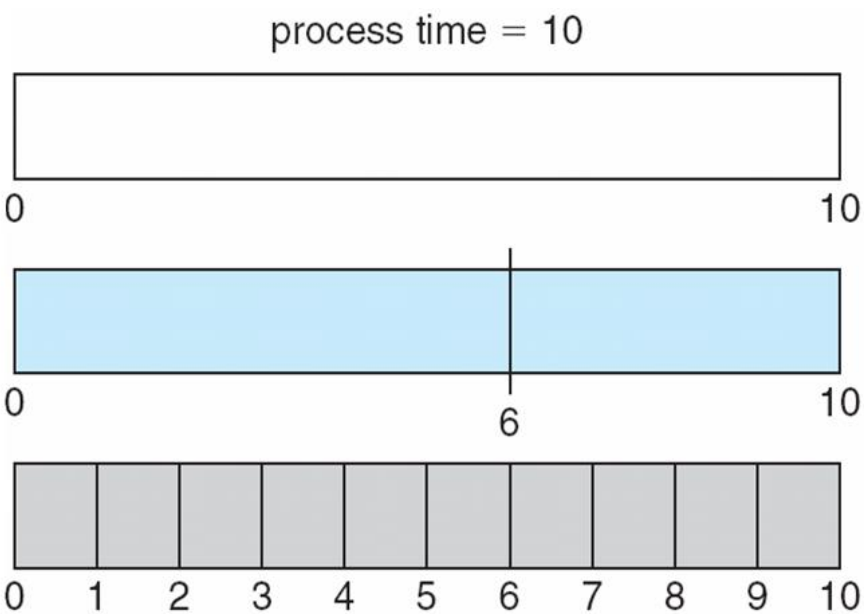
Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



- Typically, higher average turnaround, but better ***response***
- Time slice should be large compared to context switch time
- Time slice usually 1ms to 10ms, context switch < 10 usec

Time slice and Context Switch Time

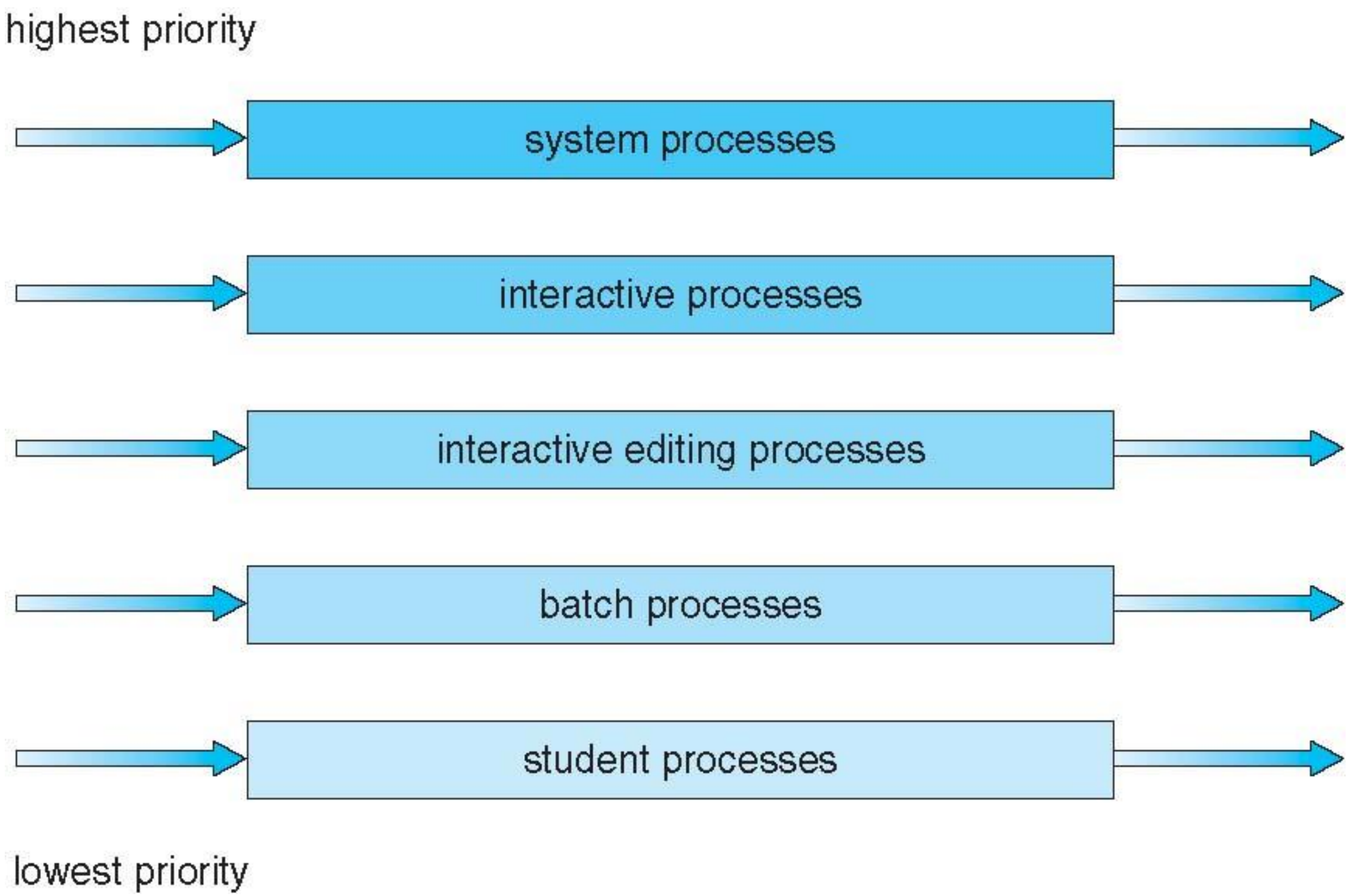


Time slice	context switches
12	0
6	1
1	9

Multilevel Queue

- Ready queue is partitioned into separate queues, eg:
  - **foreground** (interactive) w/ RR
  - **background** (batch) w/FCFS
- Process permanently in a given queue
- Each queue has its own scheduling algorithm
- Scheduling must be done between the queues:
  - Fixed priority scheduling; (i.e., serve all from foreground then from back ground). Possibility of starvation.
  - Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR
  - 20% to background in FCFS

Multilevel Queue Scheduling

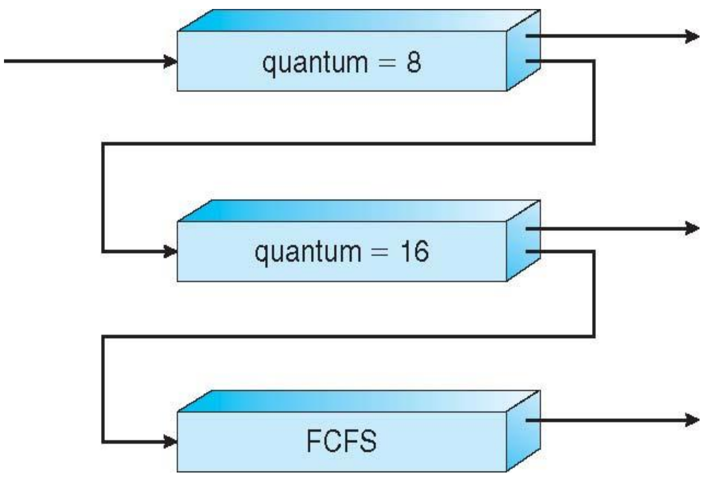


Multilevel Feedback Queue

- A process can move between the various queues; aging can be i mplemented this way
- Multilevel-feedback-queue scheduler defined by the following p arameters:
  - 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:
  - Q0 – RR with time slice 8 milliseconds
  - Q1 – RR time slice 16 milliseconds
  - Q2 – FCFS



- Scheduling
  - A new job enters queue Q0 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 Q1
  - At Q1 job is again served FCFS and receives 16 additional milliseconds
    - If it still does not complete, it is preempted and moved to queue Q2