Process Scheduling

Operating Systems

Assoc. Prof. Milos Jovanovik, PhD

Lecture Goals

- Process scheduling
- Performance metrics
- Scheduling differences
  - Batch systems
  - Interactive systems
  - Real-time systems

Scheduling

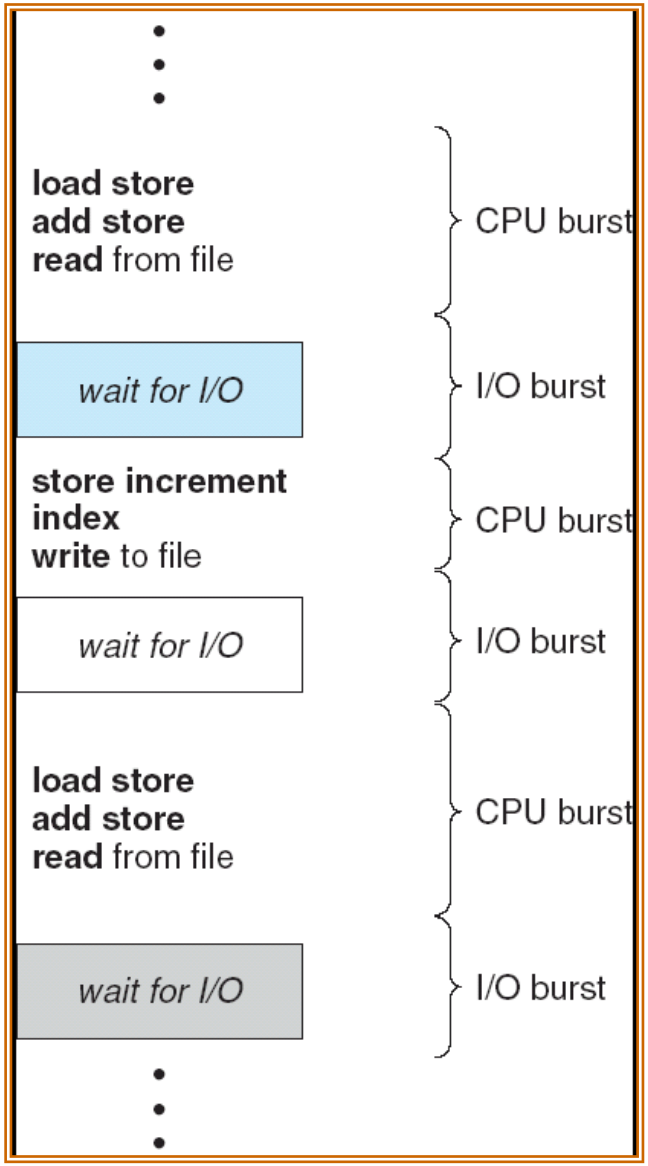
- When more than one process is in the ready state, a decision must be made about which process will get the CPU next
- The part of the OS responsible for this decision is the scheduler
- The scheduler operation is based on some scheduling algorithm

Scheduling

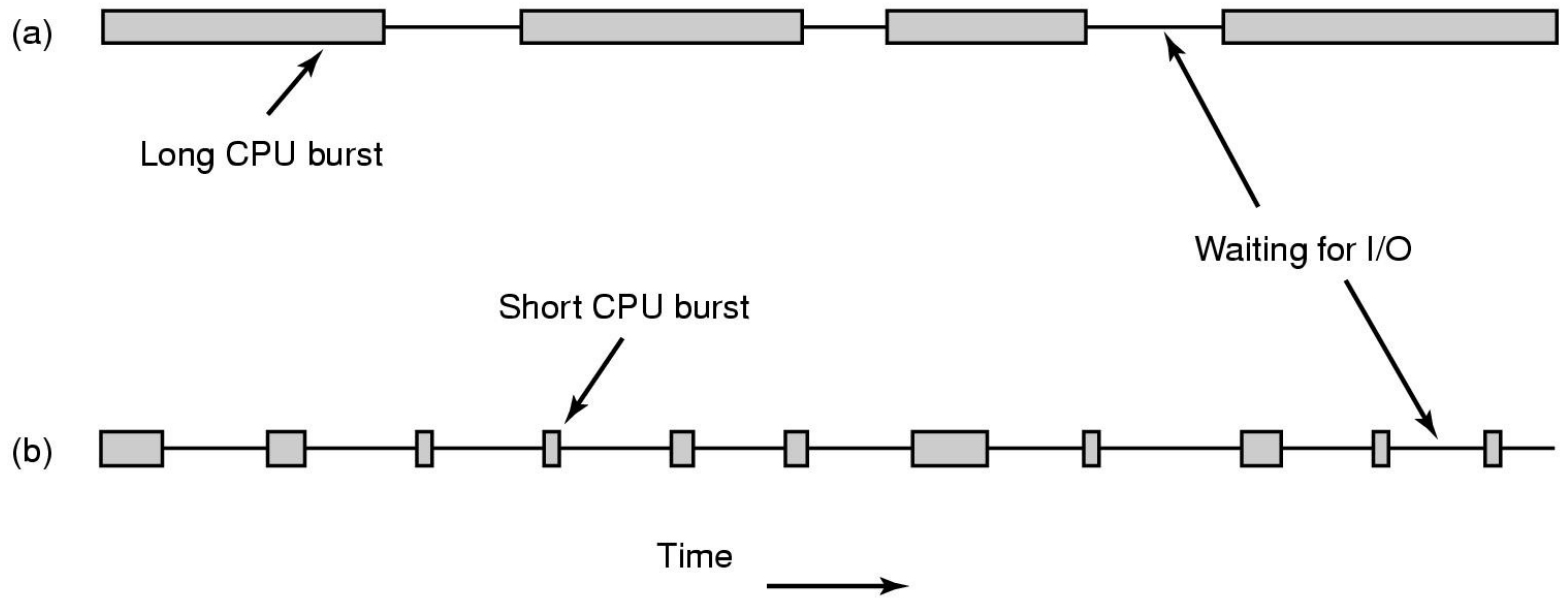
- Simplest form of scheduling:
  - Old batch systems – the next job is run
- More complex scheduling
  - Multiprogramming systems – the scheduling is more complex because there are multiple users waiting for service.
- Importance of the scheduler
  - Picking the right process for better perceived performance and user satisfaction in systems where CPU time is a scarce resource
  - Picking the frequency of process switching in systems where CPU time is an abundant resource
    - Switching is an expensive process

Process Behavior

- Nearly all processes alternate bursts of computing with I/O requests
- During I/O requests the process is waiting



Process Classes



- CPU bound process
- I/O bound process
  - Processes tend to get more I/O bound because CPUs are improving much faster than disks.

When to Schedule?

- A process creates a child process
- A process terminates
- A process goes from running to a blocked/waiting state (blocks on I/O, on a semaphore, waits a child process, etc.)
- A process goes from a running to a ready state (an interrupt occurred, etc.)
- A process goes from blocked/waiting into a ready state (the I/O operation finished, etc.)

Types of Resources

- Resources can be divided into two categories
- The category guides how the OS will manage them

1. **Nonpreemptible Resources** (cannot be taken away)
   - Once a resource has been assigned, it cannot be used until it's released
   - We need more instances from this resource
   - Example: A disk block
   - OS management: allocation
     - Decides which process gets the resource;
2. **Preemptible Resources** (can be taken away)
   - The resource can be taken away, then returned back
   - There can be only one instance of this resource
   - Example: CPU
   - OS management: scheduling
     - Decides in which order will the resource be used by the processes;
     - Decides how long a process can hold the resource;

How Does Scheduling Work?

- At specific time instances (system clock interrupts)
- **Non-preemptive**: the process executes until it blocks on its own, or until it ends
- **Preemptive**: the process can use the CPU only in a specified maximum time period

Performance Metrics for the Scheduling Algorithms

- Throughput: Number of processes finished in a given time
- Turnaround time: The time elapsed from starting the process until its completion
- CPU utilization: Percentage of time during which the CPU is busy
- Waiting time: The time spent in the ready queue (waiting for the CPU)
- Response time: The time from the first request until the first result (first response of the CPU)
  - Used for interactive systems, where the turnaround time is not an ideal metric

Categories of Different Scheduling Algorithms based on System Type

- Batch
  - Interest calculations, processing cases in an assurance company, etc.
  - No users
  - Non-preemptive or preemptive with long time intervals
- Interactive
  - Many users
  - Preemptive
- Real-time
  - Deadline matters

Scheduling Algorithm Goals

- All systems
  - Fair: each process must receive appropriate CPU time
  - Implementation of scheduling policy
  - Balance: keeping all parts of the system busy
- Batch systems
  - Throughput maximization
  - Minimization of turnaround time (time between submission and termination)
  - CPU utilization: keep the CPU busy all the time

Scheduling Algorithm Goals

- Interactive systems
  - Response time: the response should be quick
  - Proportionality: meet users' expectations
- Real-time systems
  - The deadlines should be met, without losing any data
  - Predictability: avoid quality degradation in multimedia systems

Interactive Systems: Delay & User Reaction

Delay & User Reaction	
0 – 16ms	People are exceptionally good at tracking motion, and they dislike it when animations aren't smooth. Users perceive animations as smooth so long as 60 new frames are rendered every second. That's 16ms per frame, including the time it takes for the browser to paint the new frame to the screen, leaving an app about 10ms to produce a frame.
0 – 100ms	Respond to a user action within this time window and users feel like the result is immediate. Any longer, and the connection between action and reaction is broken.
100 – 300ms	Users experience a slight perceptible delay.
300 – 1.000ms	Within this window, things feel part of a natural and continuous progression of tasks. For most users on the web, loading pages or changing views represents a task.
1.000+ms	Beyond 1 second, the user loses focus on the task they are performing.
10.000+ms	The user is frustrated and is likely to abandon the task; they may or may not come back later.

Batch-System Scheduling

- First-Come First-Served
- Shortest Job First
- Shortest Remaining Time Next

First-Come First-Served

- Simple algorithm
- Easy to program
  - Uses a linked list
- Disadvantages
  - Many I/O bound processes will take too long to execute, and their time in the CPU will be wasted waiting on I/O instead of actual CPU-related calculations and work

First-Come First-Served

- Each process which goes into the ready state, is added at the end of the linked list of ready processes
- The process at the front of this list, the one which arrived first, will be executed first
- The average waiting time of the processes is usually quite large with FCFS scheduling

Shortest Job First

- With this algorithm, the CPU is assigned to the process with the shortest expected time of execution
- It gives priority to shorter processes
- If two or more processes have the same length (in expected execution time), FCFS is used
- This algorithm can give optimal results when processes arrive almost simultaneously

Shortest Job First

- The execution time for each job should be known in advance
- Example:

- Average turnaround time
  - a) (A–8, B–12, C–16, D–20) 14 min.
  - b) 11 min.
- SJF gives best execution times if the scheduler knows all the tasks that need to be executed

Shortest Remaining Time Next

- Preemptive version of SJF
- It should know the execution time
- It calculates the remaining execution time

Shortest Remaining Time Next

- It chooses the process with the shortest remaining time
- When a new process arrives in the ready state, its execution time is compared to the remaining time of the process which is currently executing in the CPU
- If the new process needs less time to finish, compared to the current process, the scheduler will switch the two processes and the new one will start executing in the CPU

Scheduling in Interactive Systems

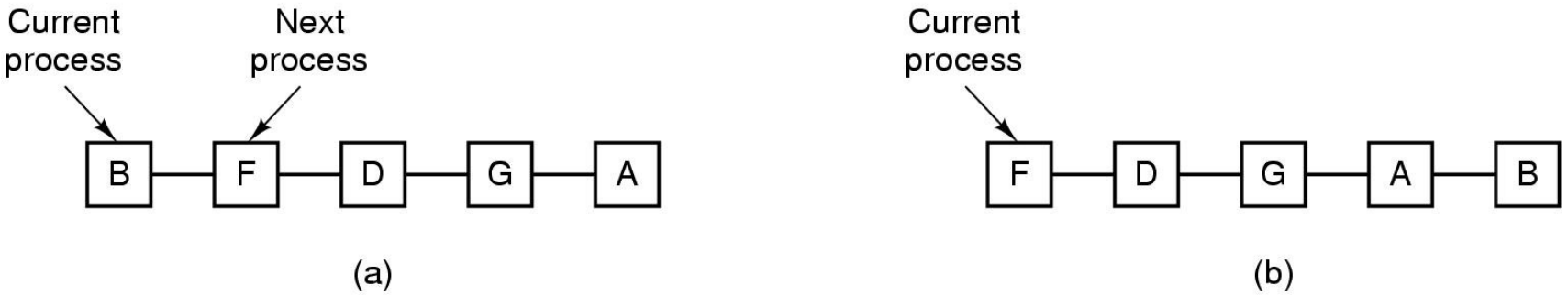
- Round-Robin
- Priority Scheduling
- Multiple Queues
- Shortest Process Next
- Guaranteed Scheduling
- Lottery Scheduling
- Fair-Share Scheduling

Round-Robin (RR)

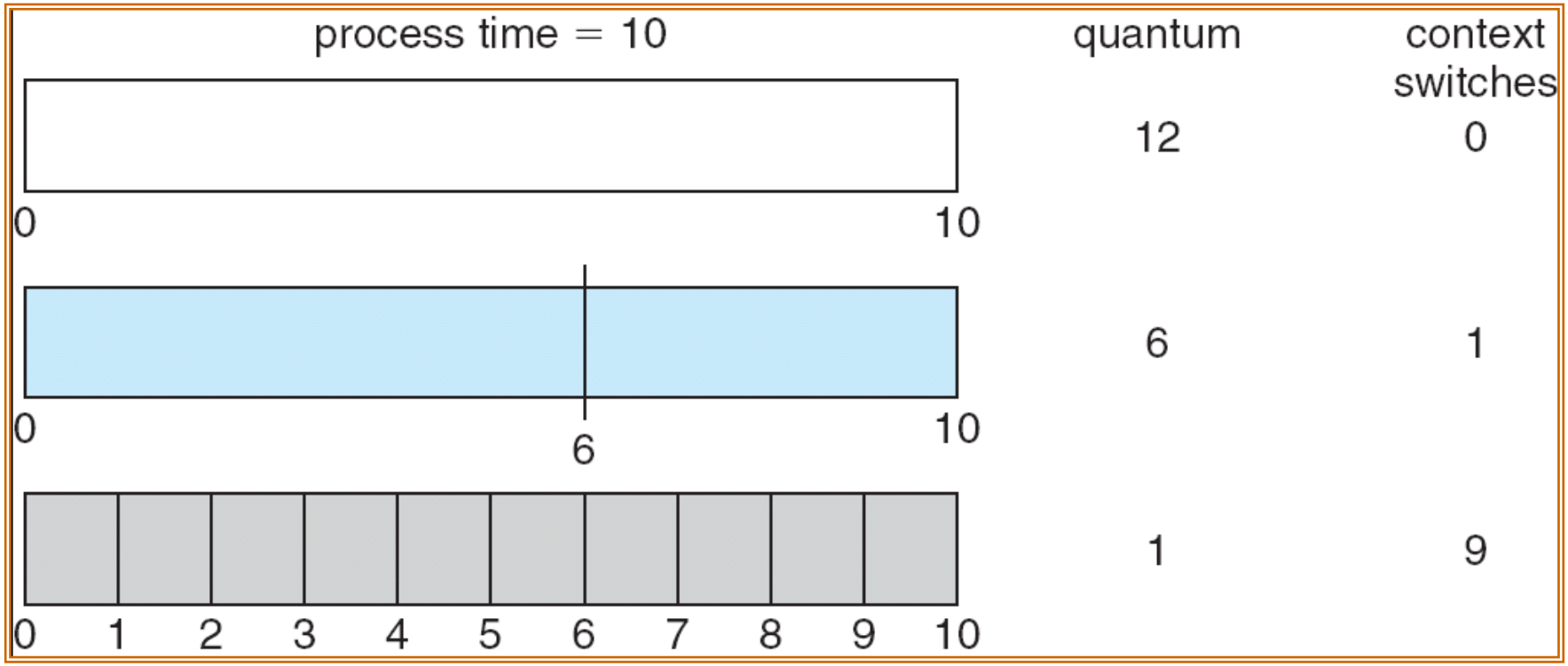
- Each process receives a small amount of CPU time (time quantum)
  - After the time quantum is up, the process is preempted from the ready queue and it is put on the end of the queue
  - Processes shorter than 1 time quantum voluntarily release the CPU.
- If there are n processes in the ready list and the quantum is q, then each process gets $1/n$ from the CPU time in portions (each no bigger than q time units). No process waits more than $(n-1)q$ time units.
- Performances
  - q big $\Rightarrow$ FIFO
  - q small $\Rightarrow$ q must be big enough compared to the time needed for context switch

Round-Robin

- a) List of ready processes
- b) List of ready process, after process B has used its quantum



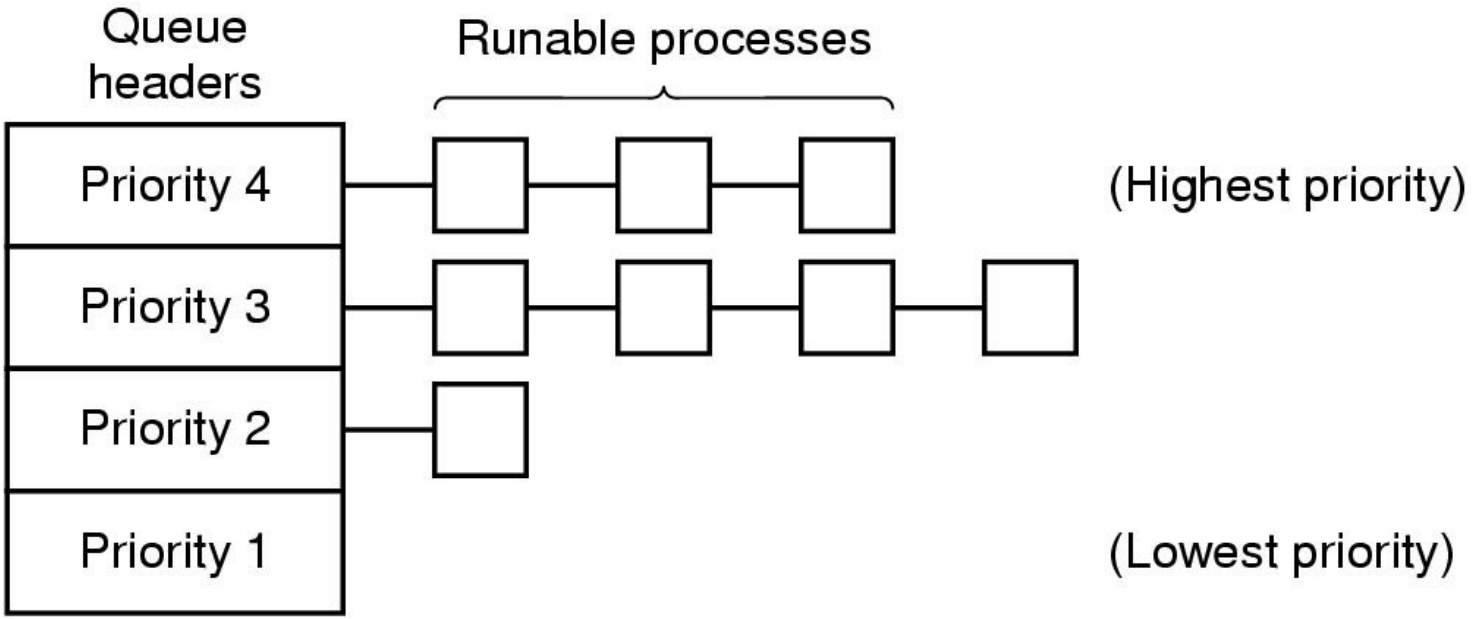
Time Quantum and Time Needed for Context Switch



- Typically q is 10–100 ms
- Context switching time is less than 10 ms

Priority Scheduling

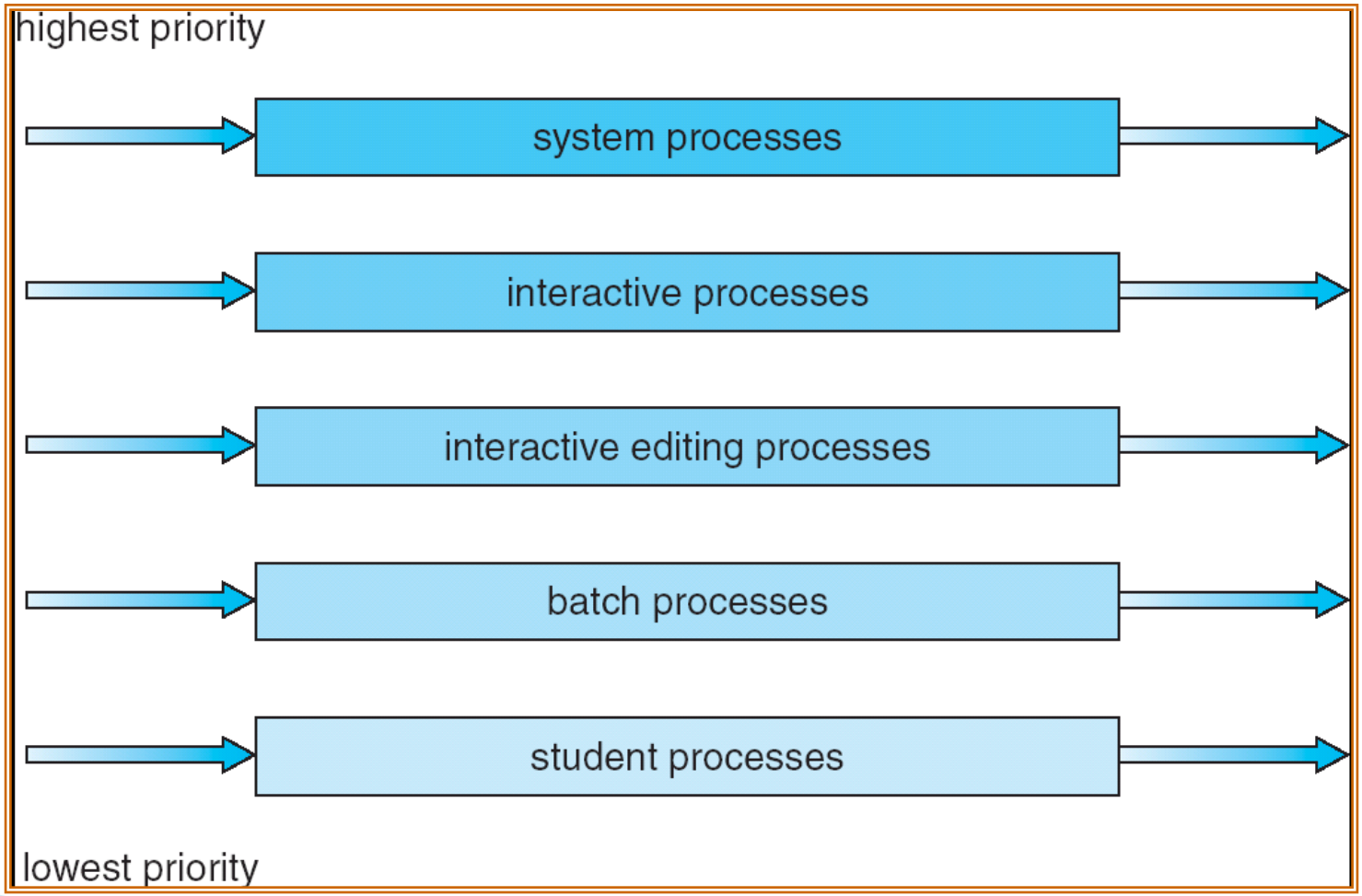
- Different priorities are assigned for different processes (depending on their importance)



Priority Scheduling

- An integer number is assigned to each process (it depicts its priority)
- The CPU is assigned to the process with the highest priority (smallest number ≡ highest priority)
  - SJF is a priority scheduling, where the priority is obtained from the predicted time for the next execution of the process
- Problem: Starvation
  - Processes with smallest priority can starve
- Solution: Aging
  - As time passes by, the process priority is increased

Multiple Queues



Priority with Multiple Queues

- It reduces the amount of process switching
- Priority classes are defined and each process receives different time for its execution
  - As a process uses the quantum from one level, it is placed into a lower priority queue with more time quanta;

Priority	Periods of Execution
Highest	1
Highest-1	2
Highest-2	4
Highest-3	8

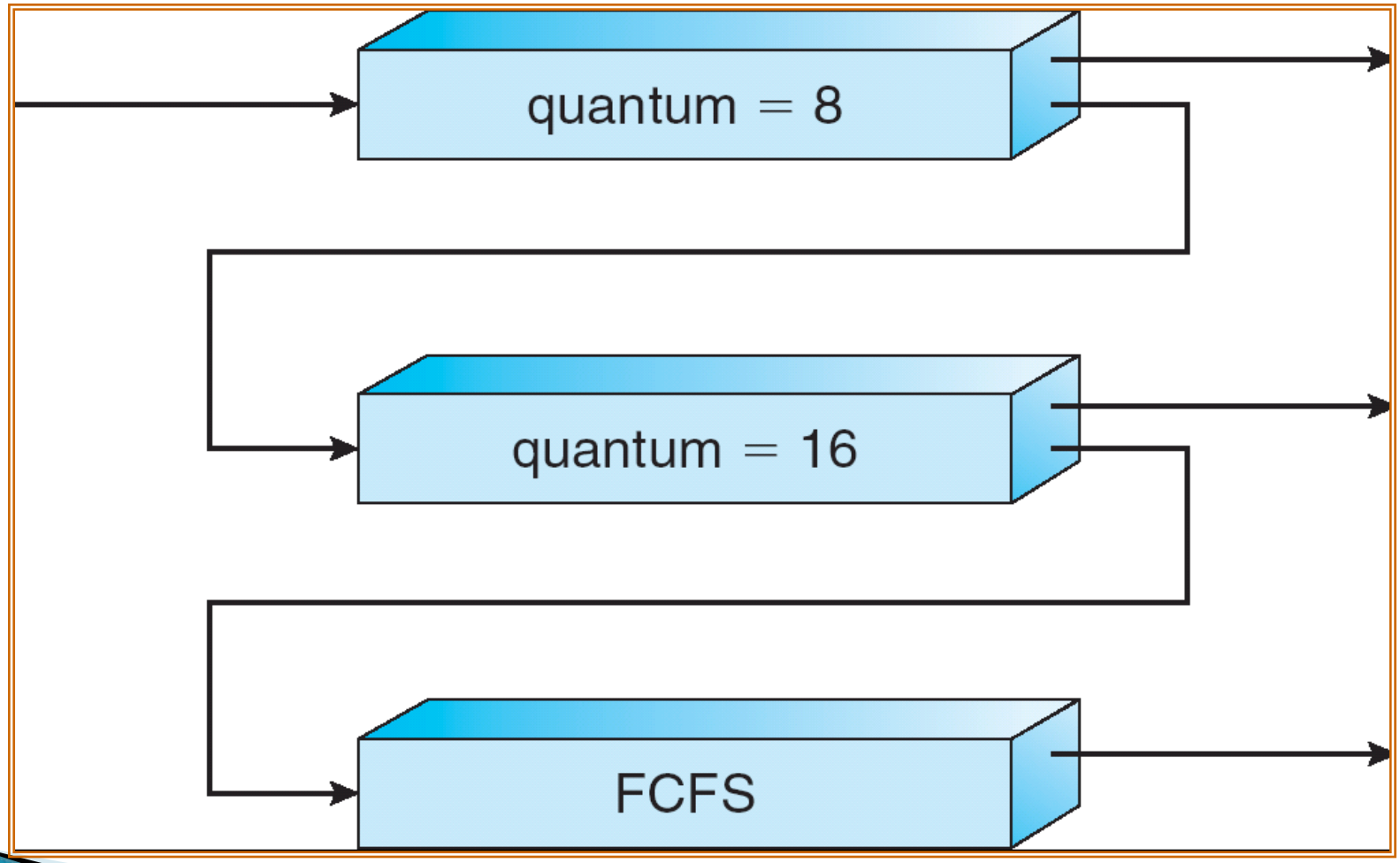
Scheduling with Multiple Queues

- The process can have variable priority
- The scheduler with multilevel feedback queues works using the following parameters:
  - Number of queues
  - Scheduling algorithm for each queue
  - A method that gives a higher priority to a process
  - A method that gives a lower priority to a process
  - A method that decides in which queue the process should be

Example

- 3 queues:
  - Q_0 – RR with 8 ms quantum
  - Q_1 – RR with 16 ms quantum
  - Q_2 – FCFS
- Scheduling
  - A new process enter in queue Q_0 and gets 8 ms CPU time to finish the task
  - If it needs more time, it is put in Q_1 and gets and gets additional 16 ms, with lower priority
  - If it needs more time, it goes to Q_2.

Multilevel Feedback Queues



Shortest Process Next

- A SJF version for interactive systems
- Problem: which of the current processes is the shortest?!?
- Each command is considered as a separate job and its execution time is measured
- Based on the prediction, the shortest command is executed as first

Simple Assessment of the Execution Time (Aging)

- The execution time can be estimated based on the past behaviour
- Suppose that T_n is the length of the n^{th} run of that process (fresh information, recent history), F_{n+1} it is our predicted value for the next run (F_n is past history)

- We define a recurrent link:

$$F_{n+1} = \alpha \cdot T_n + (1 - \alpha) \cdot F_n,$$

(with α we control the influence of the recent history)

Simple Assessment of the Execution Time (Aging)

- Assessment of the next run of the process:
- If we have two values for the process run, T_0 and T_1

  $F_1 = T_0$, $F_2 = \alpha \cdot T_1 + (1 - \alpha) \cdot F_1$,

  $F_K = \alpha \cdot T_{K-1} + (1 - \alpha) \cdot F_{K-1}$, $K = 3, 4, \ldots$

  for $\alpha = 1/2$ (recent history and "past" history have same weight)

  T_0, $T_0/2 + T_1/2$, $T_0/4 + T_1/4 + T_2/2$, $T_0/8 + T_1/8 + T_2/4 + T_3/2$, ...

  (the future is influenced less and less by the history)
- SJF is optimal (for the waiting time) when all the processes come in the same moment

Guaranteed Scheduling

- Need for guaranty that each user will get a certain amount of CPU time
- It measures the time spent by each user
- The process priority is based by the ratio of the actual CPU time consumed to the entitled CPU time

Lottery Scheduling

- For simpler implementation of scheduling in which some predefined percentage of CPU time will be assigned
- Each process/user gets some number of tickets which allow usage of some resource
- Whenever a scheduling decision has to be made, a lottery ticket is chosen at random (an integer value between 1 and the number of tickets), and the process holding that ticket gets the resource.
- Processes have several tickets.
- Its number is proportional with the probability that they will be chosen (i.e. will get the resource)

Fair-Share Scheduling

- It solves the problem when a certain user will start more processes in order to get more CPU time
- The idea is to run only one process from each user
- If one user has 4 processes A,B,C,D, and another user has a process E, then the processes will be executed in the following order:
  - AEBECEDEAE...
- If the first user needs to get twice more CPU, then the execution will be as follows:
  - ABECDEABECDE...

Scheduling in Real-Time Systems

- The main goal is the job to be finished in a given time limit
- There must be some measurement if the system performance can meet the limit
  - If there are m periodical events
  - Event i has a period P_i and it needs C_i seconds to be serviced
- The load can be tolerated if:

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

Exercises

Process scheduling tasks and exercises

Exercise 1

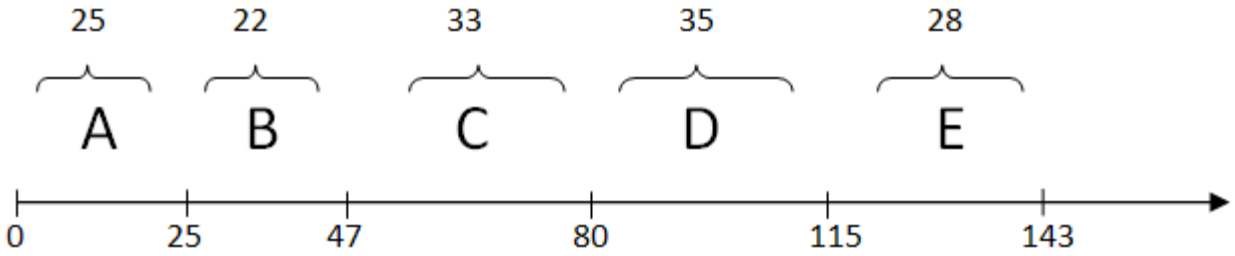
- There are 5 processes: A, B, C, D and E and each of them has expected execution time of: 25, 22, 33, 35 и 28, respectively. Assume that the processes arrive almost simultaneously.
- Draw the Gantt chart (time diagram) for each of the scheduling algorithms given in the table on the next slide and fill the corresponding fields in the table.

Exercise 1

Process	Execution Time	Time of Arrival	Scheduling Algorithms					
			FCFS		**SJF**		**Round Robin** with quantum=10 ms	
			Waiting time	Total time in the system	Waiting time	Total time in the system	Waiting time	Total time in the system
A	25	0						
B	22	0						
C	33	0						
D	35	0						
E	28	0						

Solution:

- Gantt chart for FCFS:



- Total waiting time for the processes is:

Process	Waiting time
A	0
B	25
C	47
D	80
E	115

Average waiting time: 53.4 ms

Solution:

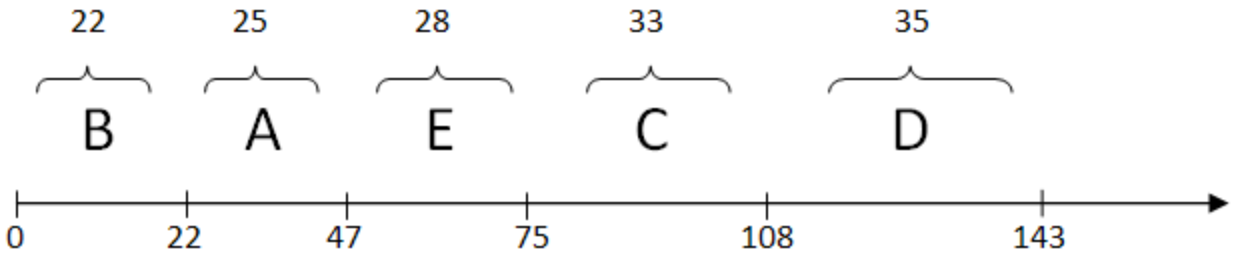
- The total time in the system for a given process is the time that has passed from their arrival till they terminate

Process	Total time in the system
A	25
B	47
C	80
D	115
E	143

The average total time in the system is: 82 ms

Solution:

- The Gantt chart for SJF is:



- The total waiting time for the processes is:

Process	Waiting time
A	22
B	0
C	75
D	108
E	47

The average waiting time is: 50.4 ms

Solution:

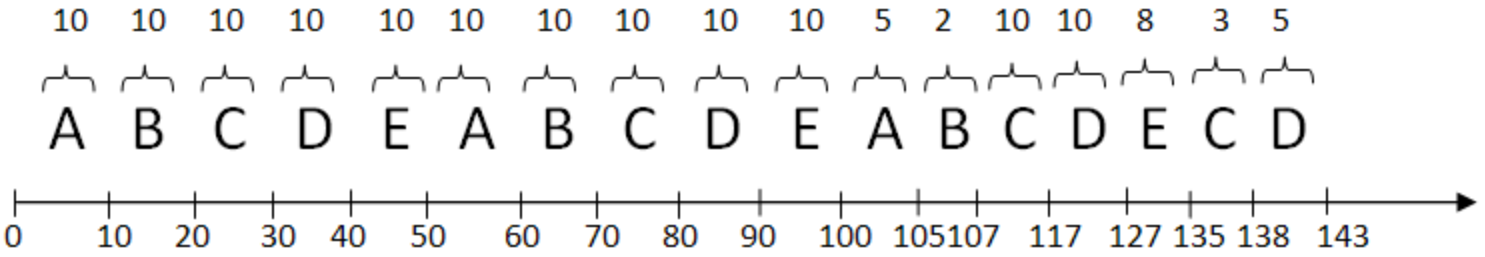
- Total time in the system for SJF:

Process	Total time in the system
A	47
B	22
C	108
D	143
E	75

The average total time in the system is: 79 ms

Solution:

- Gantt chart for RR with quantum of 10 ms is:



- The total waiting time for the processes is:

Process	Waiting time
A	0+(50-10)+(100-60) = 80
B	10+(60-20)+(105-70) = 85
C	20+(70-30)+(107-80)+(135-117) = 105
D	30+(80-40)+(117-90)+(138-127) = 108
E	40+(90-50)+(127-100) = 107

The average waiting time is: 97 ms

Solution:

- The total time in the system

Process	Total time in the system
A	105
B	107
C	138
D	143
E	135

The average total time in the system is: 125.6 ms

Solution:

- The solution table is given as:

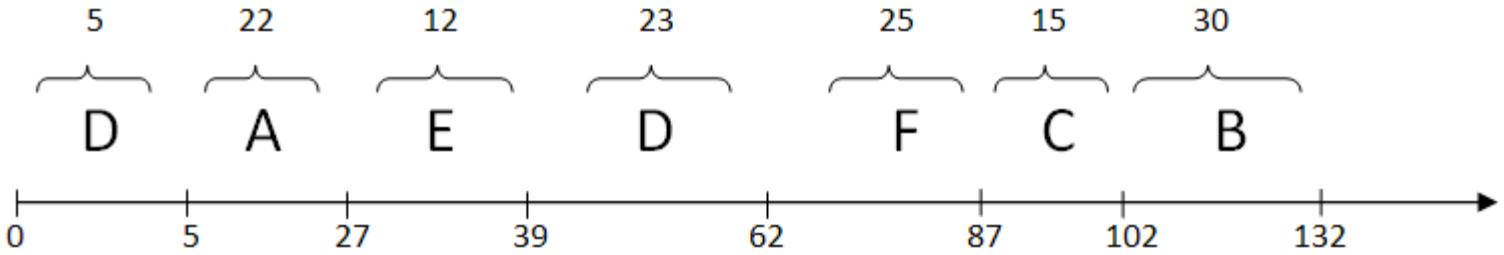
Process	Execution time	Arrival time	Scheduling algorithms					
			FCFS		**SJF**		**Round Robin** with quantum=10 ms	
			Waiting time	Total time in the system	Waiting time	Total time in the system	Waiting time	Total time in the system
A	25	0	0	25	22	47	80	105
B	22	0	25	47	0	22	85	107
C	33	0	47	80	75	108	105	138
D	35	0	80	115	108	143	108	143
E	28	0	115	143	47	75	107	135
Avg.	-	-	**53.4**	**82**	**50.4**	**79**	**97**	**125.6**

Exercise 2

- Using the SRTN algorithm, schedule the following 6 processes: A, B, C, D, E and F, with following execution times 22, 30, 15, 28, 12 and 25, respectively. Assume that their arrival times are: 5, 10, 80, 0, 15 и 50 ms, respectively.
- Draw the Gantt chart and calculate the response time for each of the processes.

Solution:

- The Gantt chart for SRTN is:



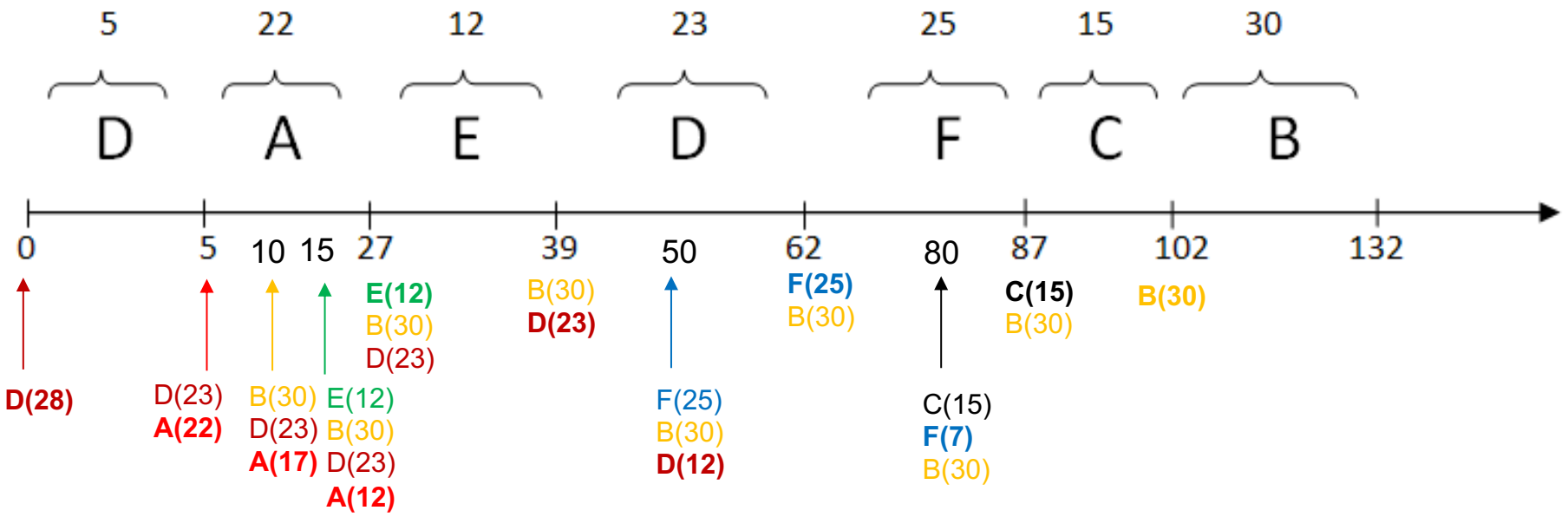
- The response time is calculated from the time of the first request till the first obtained results from the process.
  - We will assume that the first results is produced immediately after the process becomes active.

Detailed Solution:

P: A, B, C, D, E, F

d: 22, 30, 15, 28, 12, 25

T: 5, 10, 80, 0, 15, 50



Solution:

- Response time for process A:
  - The first request – 5ms
  - The first results – 5ms
  - Response time: 5 – 5 = 0ms
- Response time for process B:
  - The first request – 10ms
  - The first results – 102ms
  - Response time: 102 – 10 = 92ms
- Response time for process C:
  - The first request – 80ms
  - The first results – 87ms
  - Response time: 87 – 80 = 7ms

Solution:

- Response time for process D:
  - The first request – 0ms
  - The first results – 0ms
  - Response time: 0 – 0 = 0ms
- Response time for process E:
  - The first request– 15ms
  - The first results – 27ms
  - Response time: 27 – 15 = 12ms
- Response time for process F:
  - The first request – 50ms
  - The first results – 62ms
  - Response time: 62 – 50 = 12ms

Homework:

- Schedule the processes from exercise 2 using the following algorithms:
  - FCFS
  - SJF
  - RR with quantum 5ms
- Draw the Gantt chart for each of the algorithms and calculate the response time for each process in the system. Assume that the processes produce their first result after 1ms from the moment they become active.

Scheduling in Real-Time Systems

- Time plays a crucial role.
- Some examples are:
  - The system in CD Players, which takes the bits from the disk unit and converts them in audio in a very short interval.
    - If the calculation would need longer time, then the audio will sound awkward.
  - Systems in planes, systems that control the production in a factory, monitoring patients in hospitals, etc.

Scheduling in Real-Time Systems

- The events in this systems might be periodic (they happen in regular intervals) and aperiodic (they happen in an unpredictable fashion).
- If the real-time system can respond to multi-periodic stream of events (processes) depends on two things:
  - How much time the process needs to be executed;
  - Which is his period of occurrence;

Scheduling in real-time systems

- If we have m periodical processes and each process appears with a period P_i and needs C_i CPU seconds, then the system can control the flow of events iff:

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

- Real-time systems that satisfy this criterion are called schedulable systems.

Exercise 3:

- There is a real-time system with 3 periodic events. The processes that represent these events have periods 100, 200 и 500 msec, respectively. If these processes need 50, 30 и 100 msec CPU time, could the system control these processes, would the system be schedulable?
- If there is 4th process with period of 1 sec, what is the maximum execution time for this process in order system to be schedulable?

Solution:

- The system is schedulable if it satisfies the following inequality:

$$\sum_{i=1}^{3} \frac{C_i}{P_i} \leq 1$$

- Where $C_1 = 50$, $C_2 = 30$, $C_3 = 100$ и $P_1 = 100$, $P_2 = 200$, $P_3 = 500$, thus we have:

$$\sum_{i=1}^{3} \frac{C_i}{P_i} = \frac{50}{100} + \frac{30}{200} + \frac{100}{500} = 0.5 + 0.15 + 0.2 = 0.85 \leq 1$$

- **The system is schedulable.**

Solution:

- If there is another process with period of 1sec = 1000 msec, then in order system to be schedulable, the following inequality should hold:

$$\frac{50}{100} + \frac{30}{200} + \frac{100}{500} + \frac{x}{1000} \leq 1$$

- From which we have that $x \leq 0.15$ or that the process needs to have maximum execution time of 150 msec.

Exercise 4:

- In order to predict the process execution time the system uses aging algorithm with $\alpha=1/2$. If the previous 4 execution times of the same job were 40, 20, 40 и 15 ms, what will be the predicted time?

Solution:

- The aging algorithm calculates the next execution time as a weighted sum of the previous two executions of the same job, i.e. if T_0 and T_1 are two consecutive execution times, then T_2 is calculated as:

$$T_2 = \alpha \cdot T_1 + (1 - \alpha) \cdot T_0$$

Solution:

- The prediction of the next execution of some process (job) can be done as following:
- If we have the 2 previous values of the execution time for the process, T_0 и T_1

$$F_1 = T_0,$$

$$F_2 = \alpha \cdot T_1 + (1 - \alpha) \cdot F_1,$$

$$F_k = \alpha \cdot T_{k-1} + (1 - \alpha) \cdot F_{k-1},\ k=3, 4, \ldots$$

Solution:

- If $\alpha = 1/2$, we have the following predictions:

  T_0
  $T_0/2 + T_1/2,$
  $T_0/4 + T_1/4 + T_2/2,$
  $T_0/8 + T_1/8 + T_2/4 + T_3/2,$

- For the values in the exercise the predicted time according the aging algorithm would be:

  $T_0/8 + T_1/8 + T_2/4 + T_3/2 =$
  $40/8+20/8+40/4+15/2=25ms$

Questions?