Real-Time Operating Systems

Dr. Naira Elazab

Information Technology Dept.,
Faculty Of Computers and Information,
Mansoura University

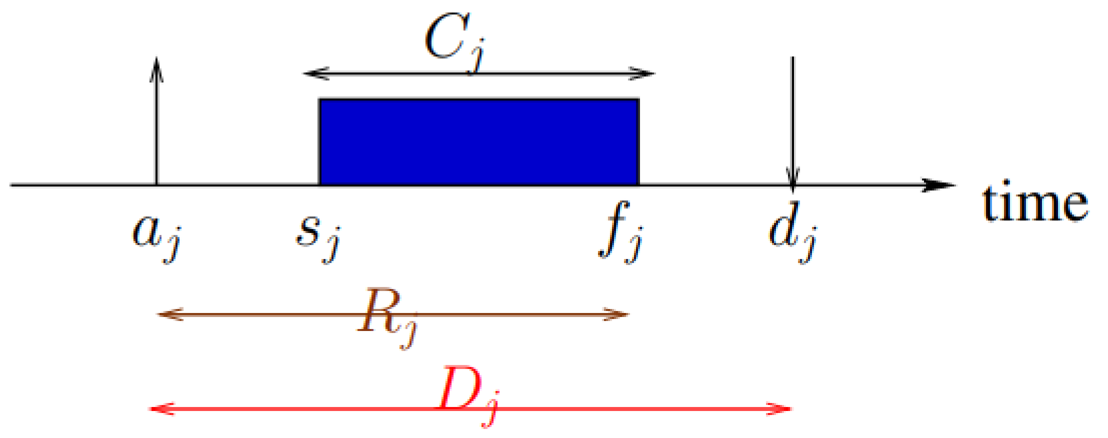
Notes!

- Supervised mode Vs. User mode.
- Critical Section of Code.
- Kernel Components (Scheduler, Objects, Services).
- Priority-based kernels (Non-preemptive kernel, preemptive kernel).

Notes!

- **Supervised mode Vs. User mode:** *User Mode* runs application code with limited access, while *Supervisor (Kernel) Mode* runs OS code with full control over hardware and system resources.
- **Critical Section of Code:** A part of a program that accesses shared resources and must be executed by only one task at a time to prevent race conditions.
- **Kernel Components (Scheduler, Objects, Services): (Scheduler, Objects, Services):** The **Scheduler** manages task execution order, **Objects** are kernel-managed entities like tasks or semaphores, and **Services** are system functions provided to applications.
- **Priority-based kernels (Non-preemptive kernel, preemptive kernel):** *Non-preemptive kernels* let a task run until it finishes, while *Preemptive kernels* allow higher-priority tasks to interrupt lower-priority ones for real-time responsiveness.

Timing parameters of a job Jj



Evaluating A Schedule

For a job J_j:

- Lateness L_j: delay of job completion with respect to its deadline.

$$L_j = f_j - d_j$$

- Tardiness E_j: the time that a job stays active after its deadline.

$$E_j = \max\{0, L_j\}$$

- Laxity (or Slack Time)(X_j): The maximum time that a job can be delayed and still meet its deadline.

$$X_j = d_j - a_j - C_j$$

Metrics of Scheduling Algorithms (for Jobs)

Given a set $\mathbb{J}$ of n jobs, common metrics are to minimize

- Average response time:

$$\sum_{J_j \in \mathbb{J}} \frac{f_j - a_j}{|\mathbb{J}|}$$

- Makespan (total completion time):

$$\max_{J_j \in \mathbb{J}} f_j - \min_{J_j \in \mathbb{J}} a_j$$

- Total weighted response time:

$$\sum w_j (f_j - a_j)$$

- Maximum latency:

$$L_{\max} = \max_{J_j \in \mathbb{J}} (f_j - d_j)$$

- Number of late jobs:

$$N_{late} = \sum_{J_j \in \mathbb{J}} miss(J_j),$$

where $miss(J_j) = 0$ if $f_j \leq d_j$, and $miss(J_j) = 1$ otherwise.

Example

Consider the following four-job, three-machine job- shop scheduling problem

	Processing time/machine number				
Job	Op.1	Op.2	Op.3	Release Date	Due date
1	4/1	3/2	2/3	0	16
2	1/2	4/1	4/3	0	14
3	3/3	2/2	3/1	0	10
4	3/2	3/3	1/1	0	8

Assume the following sequences:

- ❑ **2-1-4-3 on M1**
- ❑ **2-4-3-1 on M2**
- ❑ **3-4-2-1 on M3**

Example

Compute Makespan?

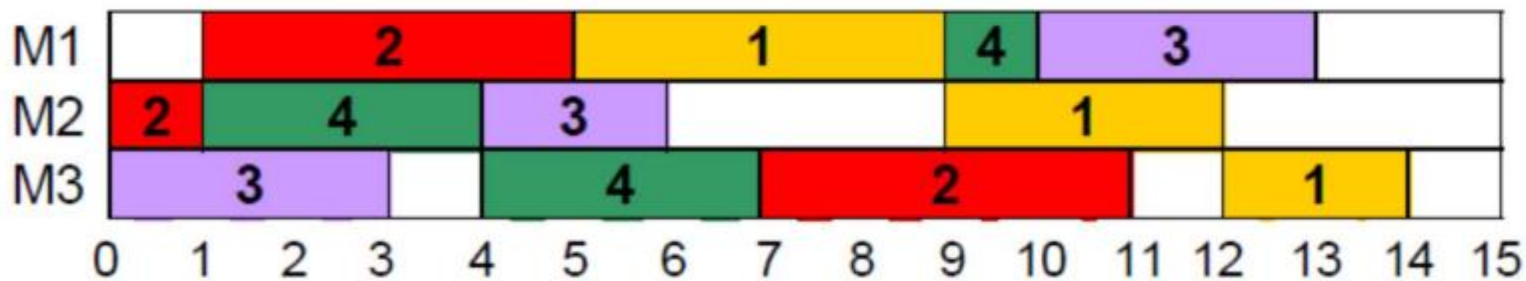
- Makespan (total completion time):

$$\max_{J_j \in \mathbb{J}} f_j - \min_{J_j \in \mathbb{J}} a_j$$

We have 4 jobs

- ❑ **max f_1=14, max f_2=11, max f_3=13, max f_4=10**
- ❑ **min a_1=0, min a_2=0, min a_3=0, min a_4=0**
- ❑ **max of all f_j=14, min of all a_j=0**
- ❑ **Makespan = 14**

Example



Compute Maximum latency?

- Maximum latency:

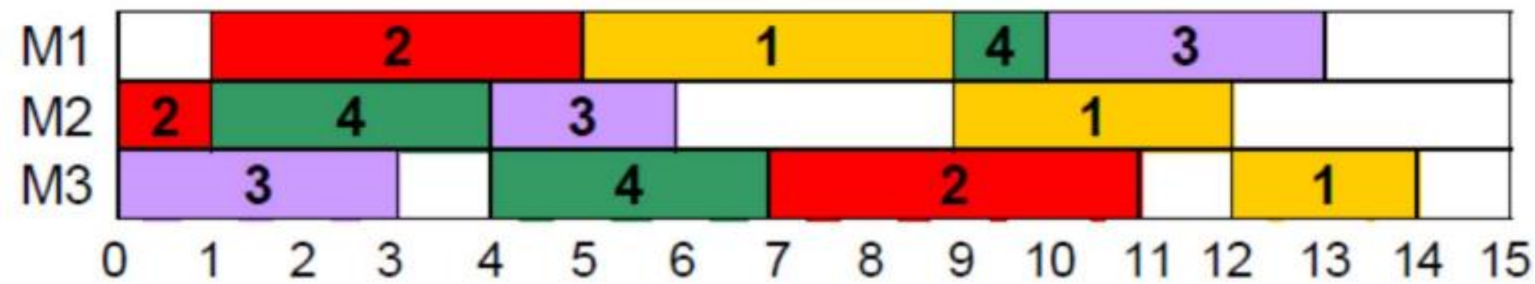
$$L_{\max} = \max_{J_j \in \mathbb{J}} (f_j - d_j)$$

We have 4 jobs

- ❑ **max f_1=14, max f_2=11, max f_3=13, max f_4=10**
- ❑ **d_1=16, d_2=14, d_3=10, d_4=8**
- ❑ **L_1=14-16=-2, L_2=11-14=-3, L_1=13-10=3, L_1=10-8=2**
- ❑ **Maximum latency = 3**

Job	Due date
1	16
2	14
3	10
4	8

Example



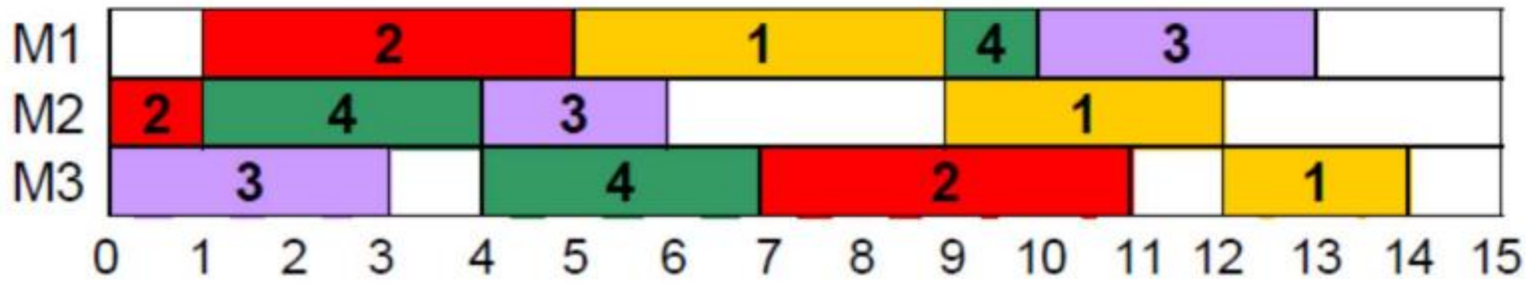
$$E_j = \max\{0, L_j\}$$

Compute Maximum Tardiness?

We have 4 jobs

- ❑ **max f_1=14, max f_2=11, max f_3=13, max f_4=10**
- ❑ **d_1=16, d_2=14, d_3=10, d_4=8**
- ❑ **L_1=14-16=-2, L_2=11-14=-3, L_1=13-10=3, L_1=10-8=2**
- ❑ **E_1=0, E_2=0, $E=_3$, $E4=_2$**
- ❑**Maximum Tardiness is 3**

Example



- Number of late jobs:

$$N_{late} = \sum_{J_j \in \mathbb{J}} miss(J_j),$$

where $miss(J_j) = 0$ if $f_j \leq d_j$, and $miss(J_j) = 1$ otherwise.

Compute the number of late (tardy) jobs?

We have 4 jobs

- **max f_1=14, max f_2=11, max f_3=13, max f_4=10**
- **d_1=16, d_2=14, d_3=10, d_4=8**
- **miss(J_1)=0, miss(J_2)=0, miss(J_3)=1, miss(J_4)=1**
- **N_{late} = 2**

Q

- The tardiness for any job in hard real time systems must be --------------
- The number of tardy jobs should be maximized in any real-time systems (T|F).
- The maximum lateness in any real-time systems should be minimized (T|F).

Recurrent Task Models

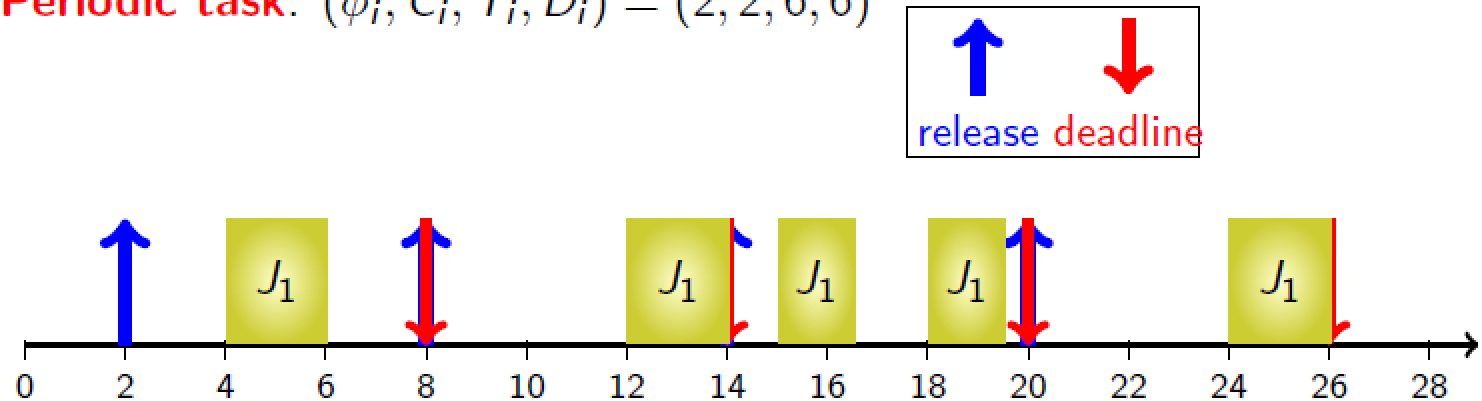
- When jobs (usually with the same computation requirement) are released recurrently, these jobs can be modeled by a recurrent task
- Periodic Task τ_i:
  - A job is released exactly and periodically by a period T_i
  - A phase ϕ_i indicates when the first job is released
  - A relative deadline D_i for each job from task τ_i
  - (ϕ_i, C_i, T_i, D_i) is the specification of periodic task τ_i, where C_i is the worst-case execution time.

Recurrent Task Models

- Sporadic Task τ_i:
  - T_i is the minimal time between any two consecutive job releases
  - A relative deadline D_i for each job from task τ_i
  - (C_i, T_i, D_i) is the specification of sporadic task τ_i, where C_i is the worst-case execution time.
- Aperiodic Task: Identical jobs released arbitrarily

Examples of Recurrent Task Models

Periodic task: $(\phi_i, C_i, T_i, D_i) = (2, 2, 6, 6)$



Sporadic task: $(C_i, T_i, D_i) = (2, 6, 6)$