COMP-SCI-431
Intro Operating Systems

Lecture 3 – Scheduling

Adu Baffour, PhD

University of Missouri-Kansas City
Division of Computing, Analytics, and Mathematics
School of Science and Engineering
aabnxq@umkc.edu

Lecture Objectives

- To understand the fundamental principles of scheduling in operating systems.
- To explore the intricacies involved in scheduling batch processes efficiently.
- To examine the challenges and considerations in scheduling interactive processes.
- To gain insights into the complexities of scheduling real-time processes and the critical factors involved.
- To learn about combined approaches to scheduling, including integrating various scheduling techniques and methodologies.

Outline

3.1 Principles of scheduling

3.2 Scheduling of batch processes

3.3 Scheduling of interactive processes

3.4 Scheduling of real-time processes

3.5 Combined approaches

Outline

3.1 Principles of scheduling

3.2 Scheduling of batch processes

3.3 Scheduling of interactive processes

3.4 Scheduling of real-time processes

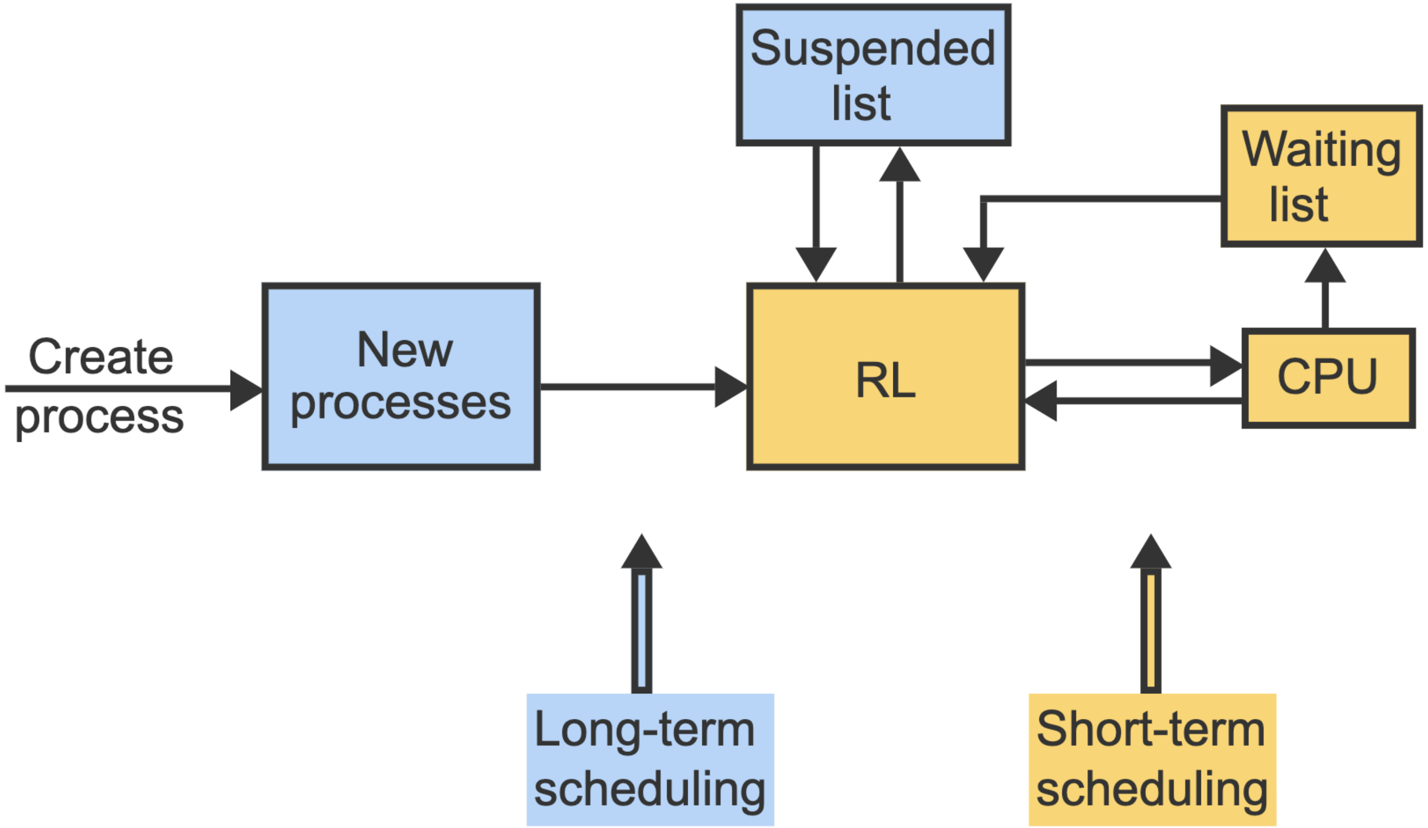
3.5 Combined approaches

3.1 Principles of scheduling

Long-term vs short-term scheduling

Scheduling decisions are made at two different levels.

- ***Long-term scheduling*** decides when a process should enter the ready state and start competing for the CPU.
- ***Short-term scheduling*** decides which ready processes should run next on the CPU.



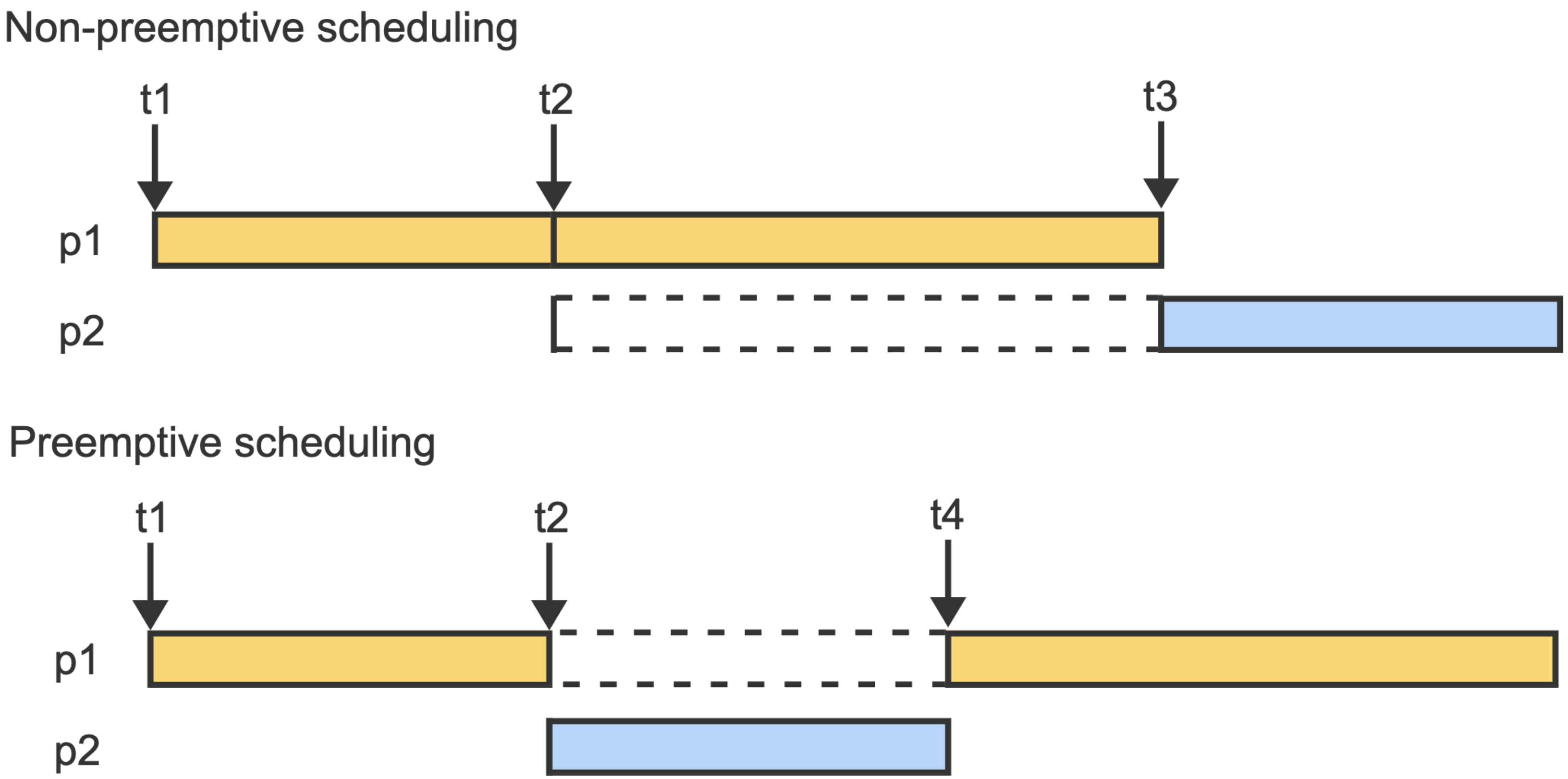
3.1 Principles of scheduling

Preemptive vs non-preemptive scheduling

- A ***non-preemptive*** scheduling algorithm allows a running process to continue until the process terminates or blocks a resource.
- A ***preemptive*** scheduling algorithm may stop the currently running process and choose another process to run. The decision is made whenever:
  - A new process enters the ready list.
  - A previously blocked or suspended process re-enters the RL.
  - The OS periodically interrupts the current process to allow other processes to run.

3.1 Principles of scheduling

Preemptive vs non-preemptive scheduling



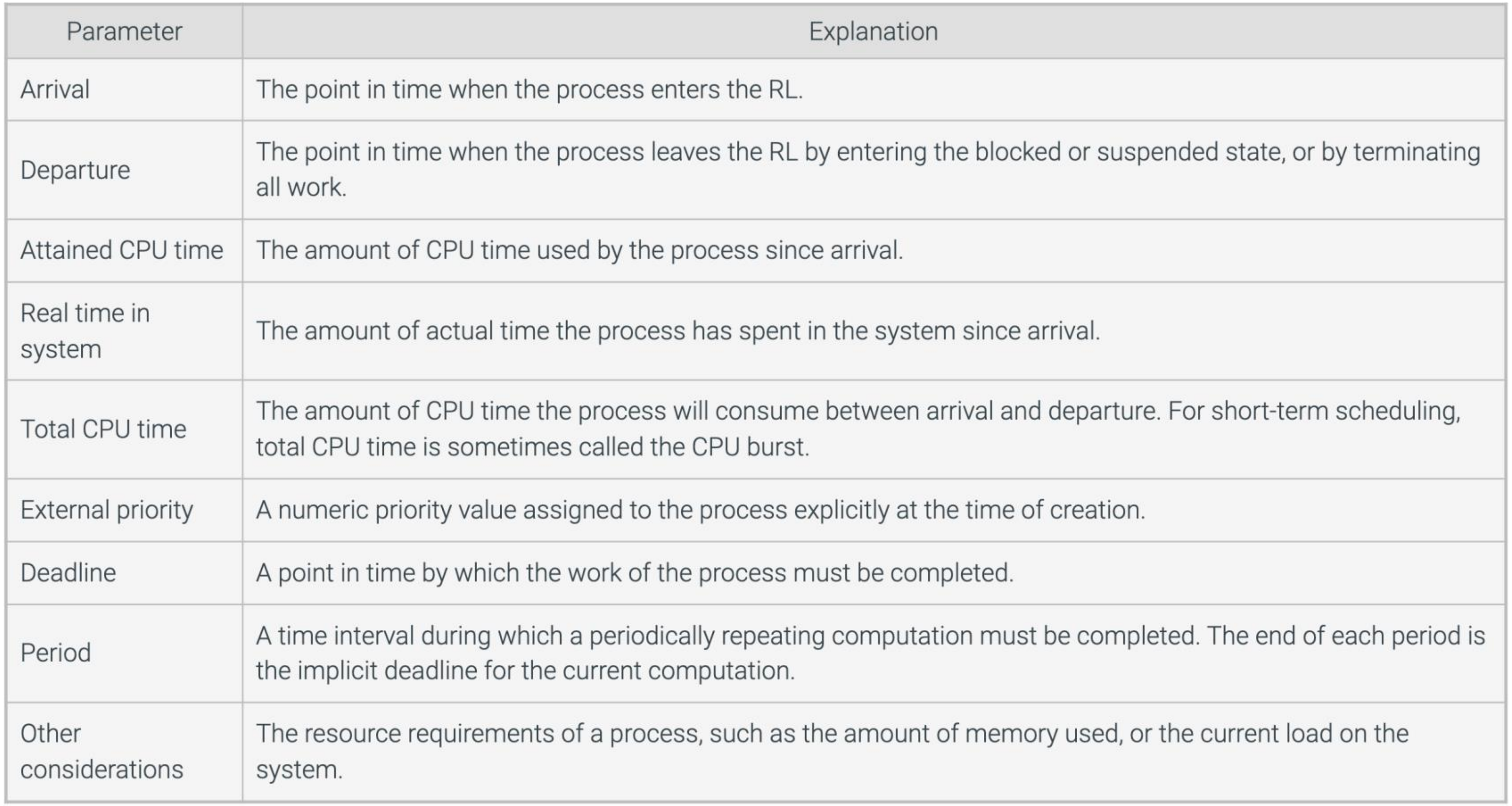
3.1 Principles of scheduling

Priority for short-term scheduling

- The ***priority of a process*** (or thread) is a numerical value that indicates the importance of the process relative to other processes.
- The priority can be a constant value assigned when the process is created or can change dynamically based on some combination of parameters.
- The ***arbitration rule*** decides which process should proceed if two or more processes have the same priority.

3.1 Principles of scheduling

Common parameters used to compute short-term priority



Parameter	Explanation
Arrival	The point in time when the process enters the RL.
Departure	The point in time when the process leaves the RL by entering the blocked or suspended state, or by terminating all work.
Attained CPU time	The amount of CPU time used by the process since arrival.
Real time in system	The amount of actual time the process has spent in the system since arrival.
Total CPU time	The amount of CPU time the process will consume between arrival and departure. For short-term scheduling, total CPU time is sometimes called the CPU burst.
External priority	A numeric priority value assigned to the process explicitly at the time of creation.
Deadline	A point in time by which the work of the process must be completed.
Period	A time interval during which a periodically repeating computation must be completed. The end of each period is the implicit deadline for the current computation.
Other considerations	The resource requirements of a process, such as the amount of memory used, or the current load on the system.

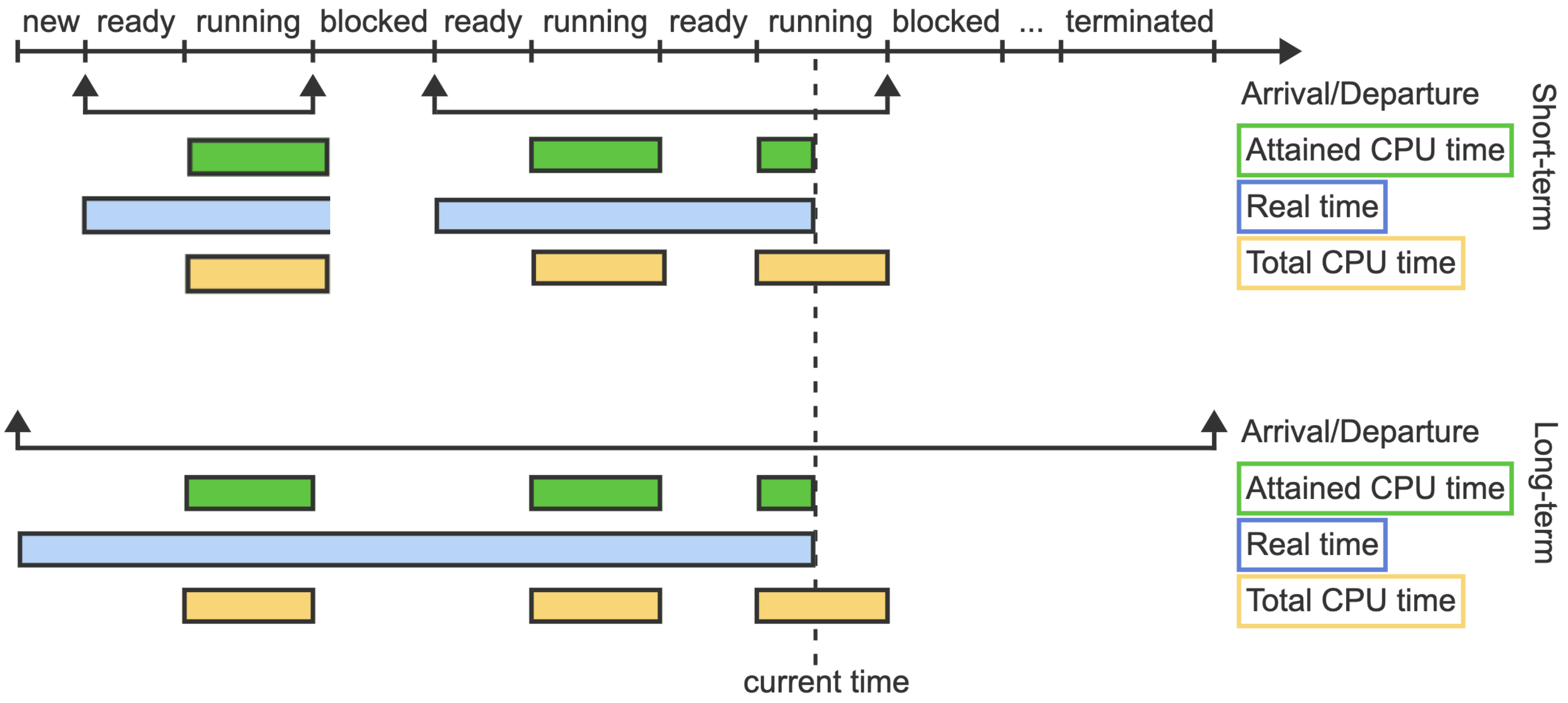
3.1 Principles of scheduling

Priority for long-term scheduling

- Long-term priority can be based on the same parameters as short-term priority: arrival, departure, attained CPU time, real-time in the system, total CPU time, and external priority.
- Long-term scheduling occurs much less frequently than short-term scheduling, and thus, the decisions are made at a higher granularity of time.
- Arrival is the time of process creation.
- Departure is the time of process destruction.
- Consequently, the attained CPU time and the total CPU time have different meanings in short-term and long-term scheduling.

3.1 Principles of scheduling

Parameters for short-term and long-term priority



Outline

3.1 Principles of scheduling

3.2 Scheduling of batch processes

3.3 Scheduling of interactive processes

3.4 Scheduling of real-time processes

3.5 Combined approaches

3.2 Scheduling of batch processes

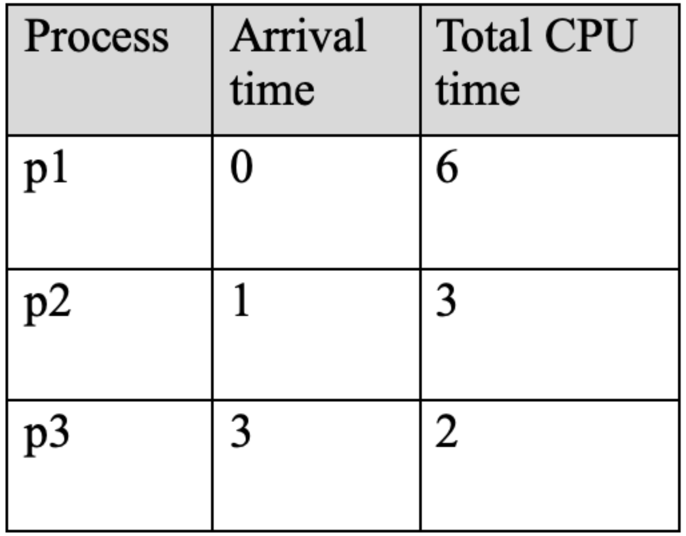
- A ***batch process*** performs a long-running and generally repetitive task that does not require any intervention from the user.
- Ex: Payroll, insurance claims processing, weather prediction, scientific calculations.

FIFO scheduling algorithm

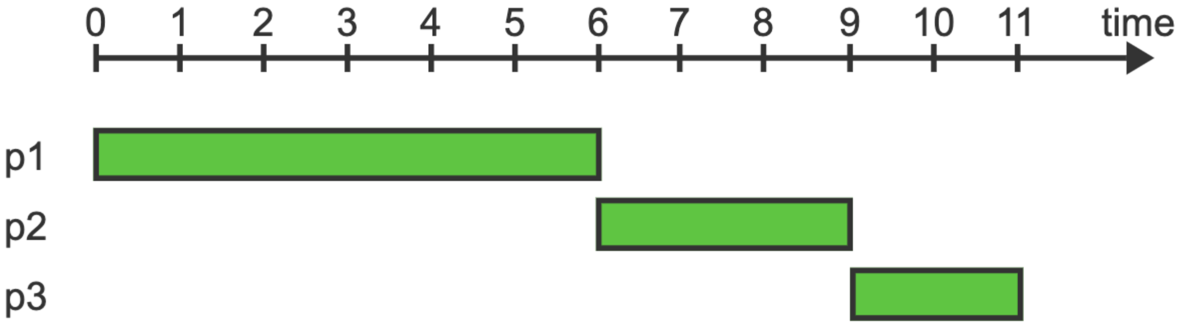
- The ***FIFO (First-In-First-Out) algorithm***, also known as FCFS (First-Come-First-Served), schedules processes strictly according to the process arrival time.
- Theoretically, multiple processes could have the same arrival time, so the arbitration rule can randomly pick a process.
- FIFO is non-preemptive.

3.2 Scheduling of batch processes

FIFO scheduling algorithm



Process	Arrival time	Total CPU time
p1	0	6
p2	1	3
p3	3	2



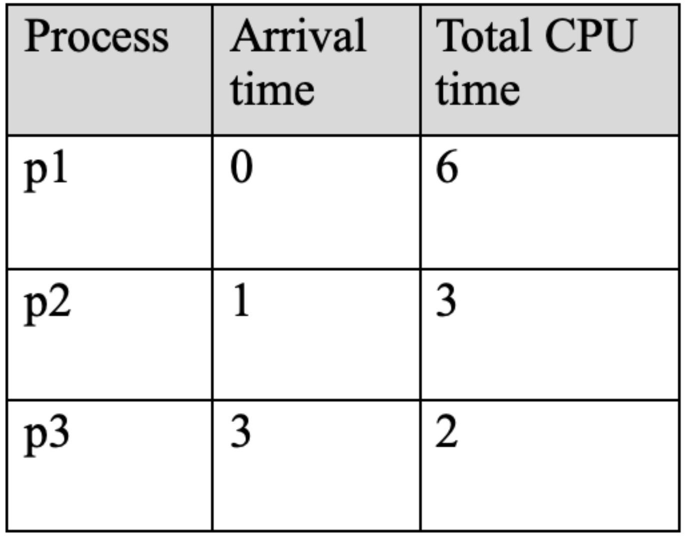
3.2 Scheduling of batch processes

SJF scheduling algorithm

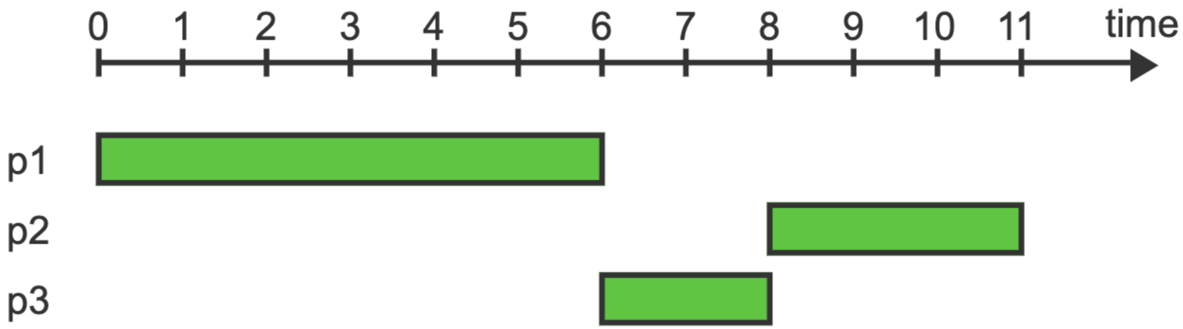
- The ***SJF (Shortest Job First) algorithm***, also known as SJN (Shortest Job Next), schedules processes according to the total CPU time requirements.
- The shorter the required CPU time, the higher the priority.
- If multiple processes have the exact CPU time requirement, then the arbitration rule can select a process based on the arrival times.
- SJF is non-preemptive.

3.2 Scheduling of batch processes

SJF scheduling algorithm



Process	Arrival time	Total CPU time
p1	0	6
p2	1	3
p3	3	2



3.2 Scheduling of batch processes

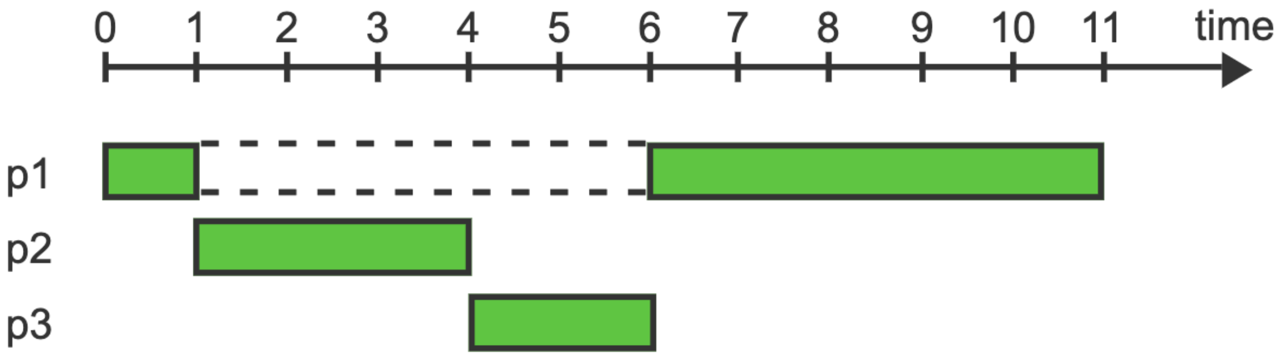
SRT scheduling algorithm

- The ***SRT (Shortest Remaining Time) algorithm*** schedules processes according to the remaining CPU time needed to complete the work.
- The shorter the remaining CPU time, the higher the priority.
- If multiple processes have the same remaining time requirement, then the arbitration rule can select a process based on the arrival times.
- SRT is the preemptive version of SJF.

3.2 Scheduling of batch processes

SRT scheduling algorithm

Process	Arrival time	Total CPU time
p1	0	6
p2	1	3
p3	3	2



3.2 Scheduling of batch processes

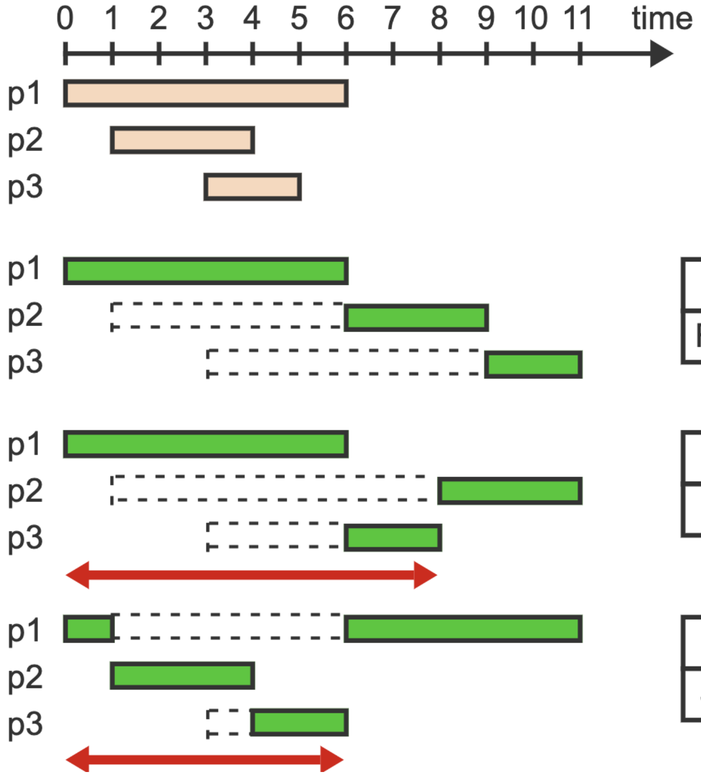
Performance of the algorithms

- The ***turnaround time*** of a process is the time between arrival and departure and is the sum of the total CPU time and the waiting time.
- Turnaround time (TAT) = Waiting time + CPU Burst
- Turnaround time (TAT) = Completion time – Arrival Time
- The ***average turnaround time (ATT)*** for a set of n processes is the mean of the n individual turnaround times.
- ***Starvation*** is the indefinite postponement of a process while other processes can proceed.
- Both SJF and SRT can lead to starvation.

3.2 Scheduling of batch processes

ATT for different scheduling algorithms

Process	Arrival time	Total CPU time
p1	0	6
p2	1	3
p3	3	2



	p1	p2	p3	ATT
FIFO	0+6	5+3	6+2	22/3=7.33

	p1	p2	p3	ATT
SJF	0+6	7+3	3+2	21/3=7

	p1	p2	p3	ATT
SRT	5+6	0+3	1+2	17/3=5.66

Outline

3.1 Principles of scheduling

3.2 Scheduling of batch processes

3.3 Scheduling of interactive processes

3.4 Scheduling of real-time processes

3.5 Combined approaches

3.3 Scheduling of interactive processes

- An ***interactive process*** communicates with the user through a dialog by receiving commands and responding by generating output on the user's terminal or another output device.
- The primary goal in scheduling interactive processes is to respond promptly to each input.
- Consequently, interactive processes must time-share the CPU using preemptive scheduling to allow each process to progress on time.

3.3 Scheduling of interactive processes

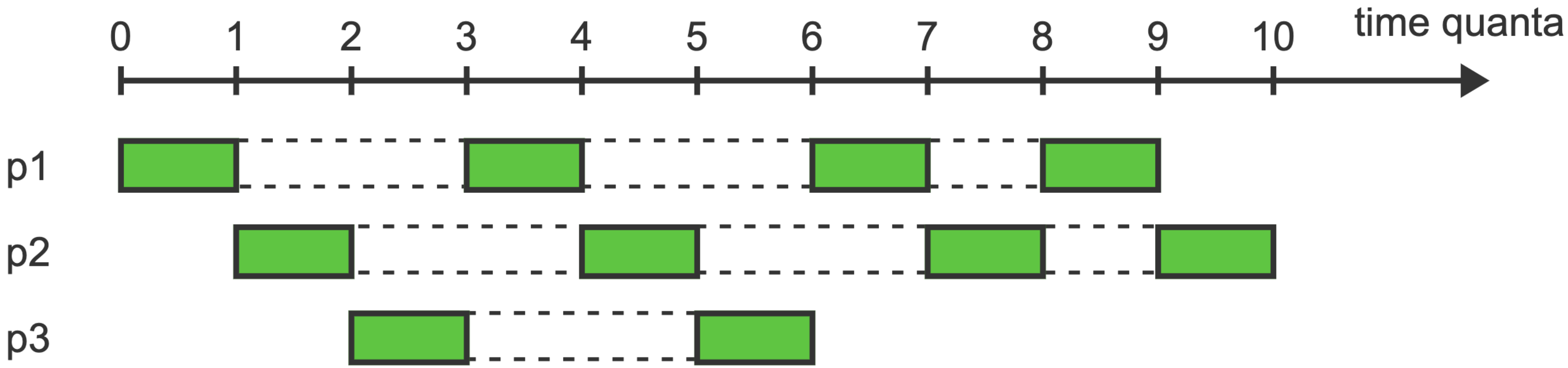
RR scheduling algorithm

- A ***time quantum***, Q, is a small amount of time (typically 10 to 100 milliseconds) during which a process is allowed to use the CPU.
- The ***round-robin (RR) algorithm*** uses a single queue of processes.
- The priority is determined solely by a process's position within the queue.
- The process at the head of the queue has the highest priority and is allowed to run for Q time units.
- When Q ends, the process is moved to the tail of the queue, and the next process, now at the head of the queue, is allowed to run for Q time units.

3.3 Scheduling of interactive processes

Execution under RR scheduling

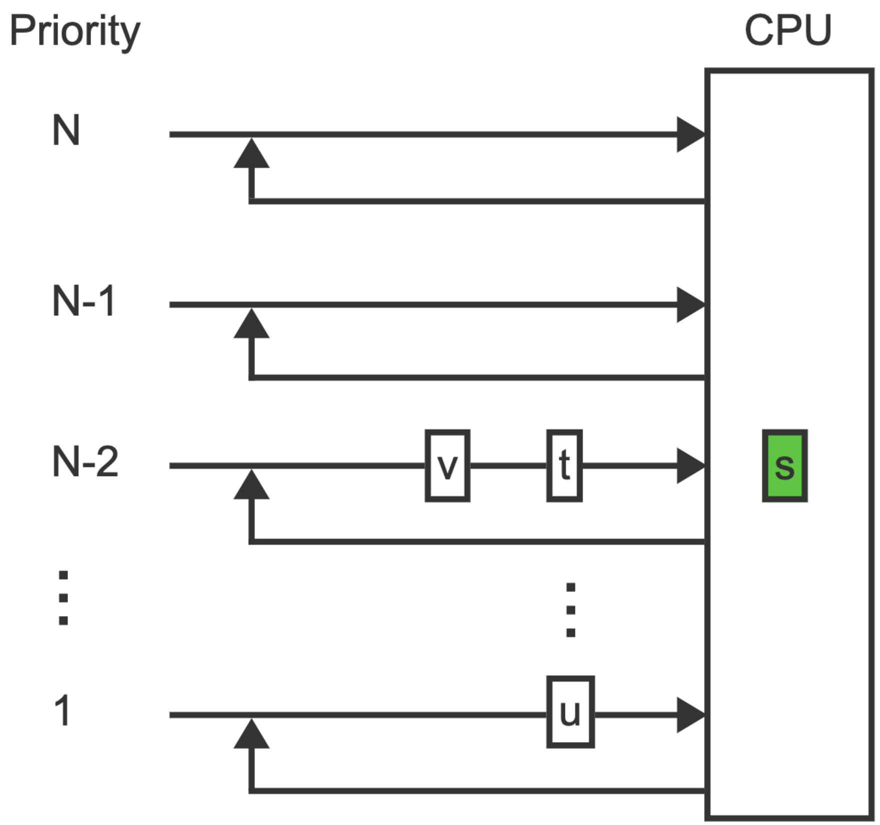
- Three processes with total CPU times of 4, 4, and 2 start executing under RR, with Q=1.



3.3 Scheduling of interactive processes

ML scheduling algorithm

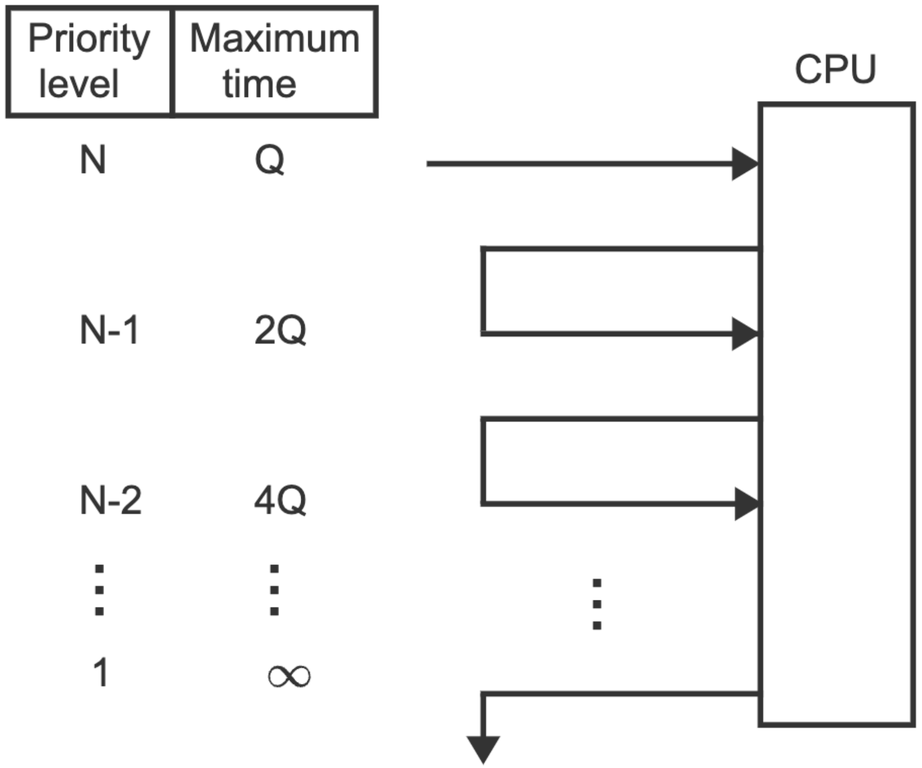
- External priorities can be used to divide processes into groups based on importance.
- ***Multilevel (ML) scheduling*** maintains a separate queue of processes at each priority level.
- Within each level, processes are scheduled using RR.



3.3 Scheduling of interactive processes

MLF scheduling algorithm

- Under the ***multilevel feedback (MLF) algorithm,*** a newly arriving process enters the highest-priority queue, N, and can run for Q time units.
- When Q is exceeded, the process is moved to the next lower priority queue, N-1, and can run for 2Q time units.
- The quantum size is doubled with each decreasing priority level.
- MLF automatically favors short-running processes, while processes with long running times gradually migrate to lower priority levels.



Priority level	Maximum time
N	Q
N-1	2Q
N-2	4Q
⋮	⋮
1	∞

3.3 Scheduling of interactive processes

Performance of interactive scheduling algorithms

- The ***response time*** of a process is the elapsed time from submitting a request (pressing the Enter key or clicking a mouse button) until the response begins to arrive.
- Guaranteeing adequate response time is the primary goal in scheduling interactive processes.

Outline

3.1 Principles of scheduling

3.2 Scheduling of batch processes

3.3 Scheduling of interactive processes

3.4 Scheduling of real-time processes

3.5 Combined approaches

3.4 Scheduling of real-time processes

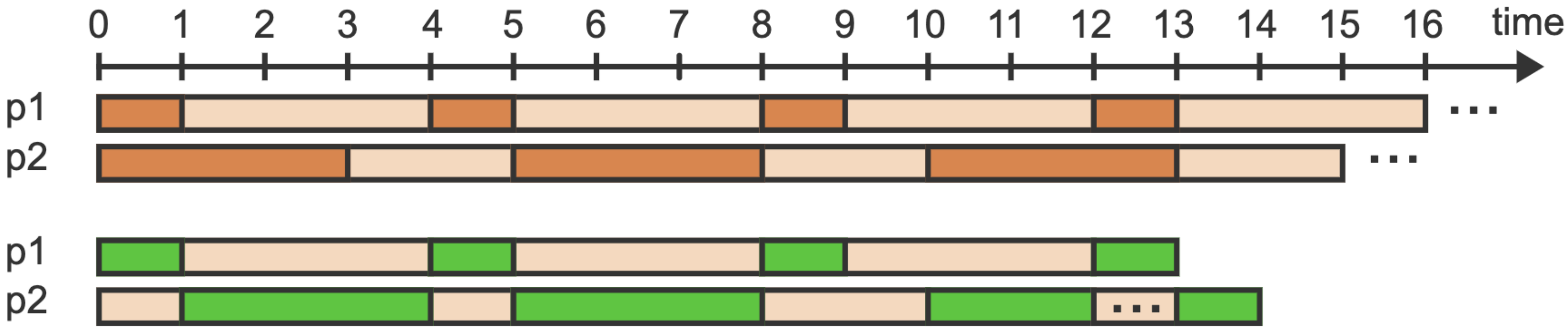
- A ***real-time process*** is characterized by continual input, which must be processed fast enough to generate nearly instantaneous output.
- Each arriving input item is subject to a deadline. Ex: Streaming audio or video.
- A ***period*** is a time interval within which each input item must be processed.

RM scheduling algorithm

- The ***rate monotonic (RM)*** algorithm schedules processes according to the period.
- The shorter the period, the higher the priority.
- RM is preemptive.

3.4 Scheduling of real-time processes

Execution under RM scheduling

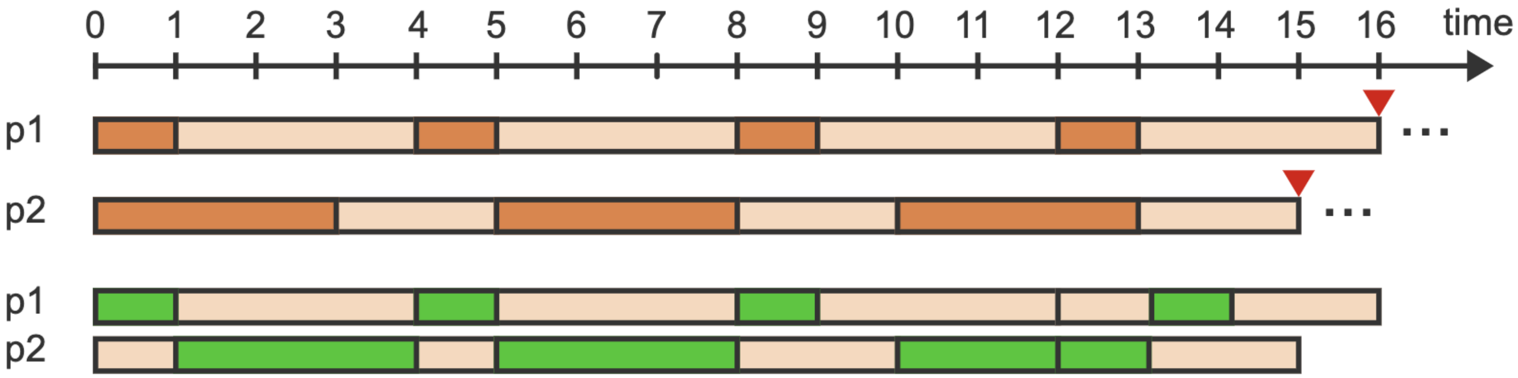


Process	Period	Tot. CPU
p1	4	1
p2	5	3

3.4 Scheduling of real-time processes

EDF Scheduling algorithm

- The ***earliest deadline first (EDF)*** algorithm schedules processes according to the shortest remaining time until the deadline.
- The shorter the remaining time, the higher the priority.
- EDF is preemptive.



Process	Period	Tot. CPU
p1	4	1
p2	5	3

3.4 Scheduling of real-time processes

Performance of the algorithms

- The most important goal is to meet all deadlines.
- A schedule is ***feasible*** if the deadlines of all processes can be met.
- The fraction of CPU time used by process i, is $\frac{T_i}{D_i}$ where T_i is the total CPU time and D_i is the period of process i.
- The ***CPU utilization (U)*** is the sum of the individual fractions of CPU times used by each process:

$$U = \sum_{i=1}^{n} \frac{T_i}{D_i}$$

- If U = 1 then the CPU is utilized 100%.
- A feasible schedule exists as long as U ≤ 1.

Outline

3.1 Principles of scheduling

3.2 Scheduling of batch processes

3.3 Scheduling of interactive processes

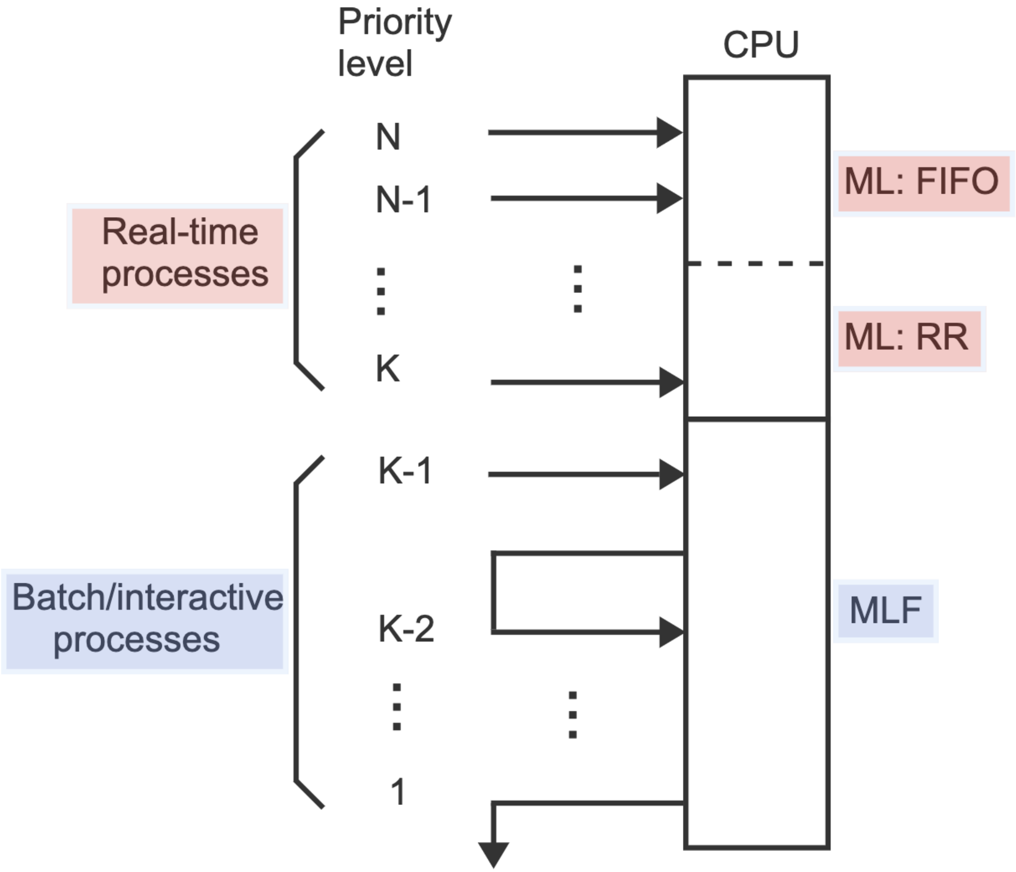
3.4 Scheduling of real-time processes

3.5 Combined approaches

3.5 Combined approaches

A 2-tier approach

- The most straightforward approach is to divide processes into two groups.
- Real-time processes run at the highest priority level but can use FIFO due to their short running times.
- Interactive and batch processes can be scheduled together using MLF.



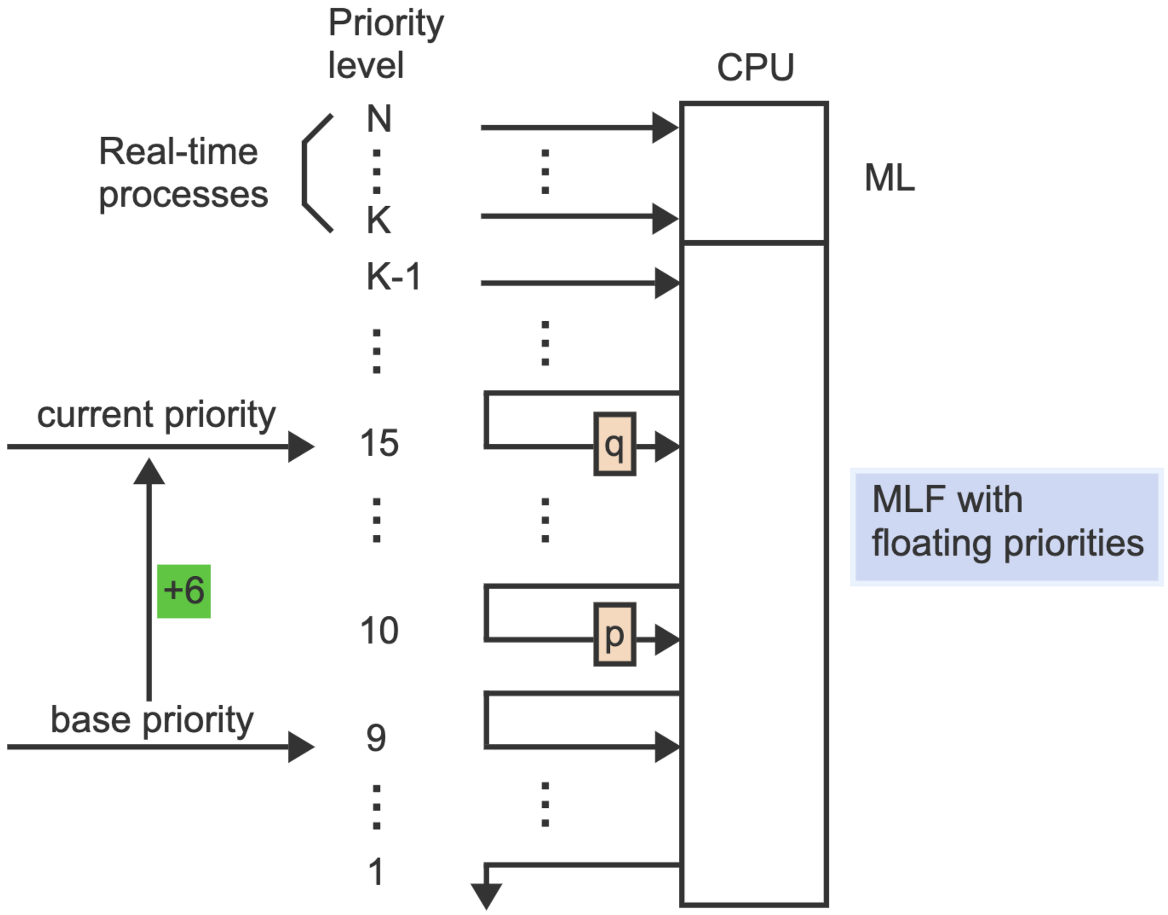
3.5 Combined approaches

A 2-tier approach with floating priorities

- Most modern general-purpose OSs (Ex, Windows, Linux) use more sophisticated scheduling strategies to provide more flexibility.
- Real-time processes use either FIFO or RR in an ML scheme.
- Interactive and batch processes use a variation of MLF.
- Every process is assigned a base priority at creation, but an increment is added based on the past action the process has taken.
- Thus, the current priority depends on the type of process and the process's behavior.

3.5 Combined approaches

A 2-tier approach with floating priorities



Further Reading

- Chapter 5 – Avi Silberschatz, Peter Baer Galvin, Greg Gagne, Operating System Concepts Essentials.

- The Linux CFS scheduler is further described in https://www.ibm.com/developerworks/library/l-completely-fair-scheduler/.

End of Lecture

Thank you

Any questions?