Operating Systems

Course #8

Process scheduling

Răzvan Daniel ZOTA
Faculty of Cybernetics, Statistics and Economic Informatics
zota@ase.ro
https://zota.ase.ro/os

Process scheduling – what is all about?

- Basically, it represents the way a process is "attached" to the processor
- It's centered around efficient algorithms
- A scheduler design must assure that all users have fair access to resources.

CPU scheduling

Main concepts (recap)

Multiprogramming Multiple programs can be in memory in the same time. The CPU and I/O can be overloaded.

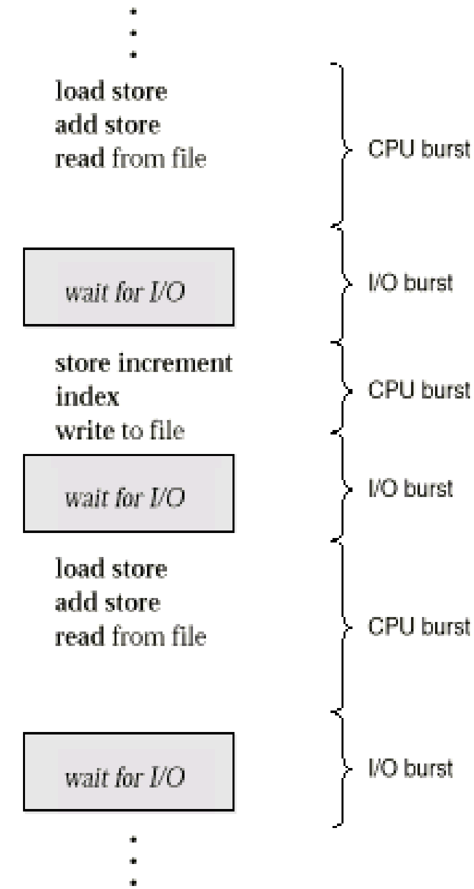
Jobs (Batch) programs that are running without the user intervention.

User programs (Time sharing) programs that may need user intervention

Process Can be both

CPU - I/O burst cycle There are representing the execution of the processes alternating between CPU activity and I/O. The CPU times are a lot smaller comparative with I/O operations.

Preemptive scheduling An interrupt is stopping the current running process and replacing it with another process.



CPU scheduling

Performance evaluation criteria

Operation degree	The time fraction when a device is used (operation_time/total_time)
Throughput	The number of jobs terminated in a period of time (jobs/second)
Service time	Time used by a device to solve a request (in seconds)
Queue waiting time	Time spent in the waiting queue (in seconds)

CPU scheduling

Performance evaluation criteria

Residence time Time spent by a request at a device.
Residence time= Service time+ Queue waiting time

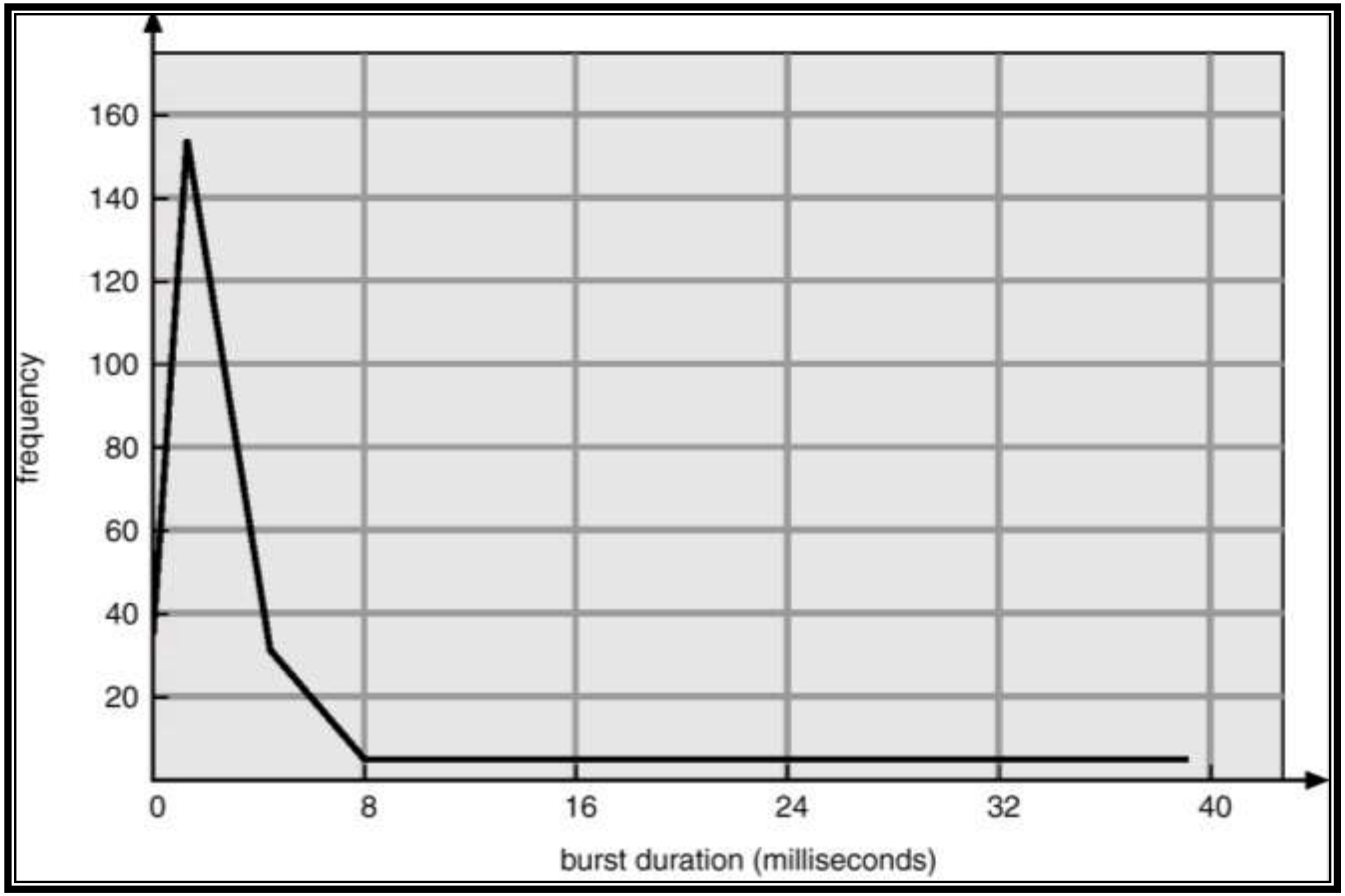
Response time Time used by the system to respond to a user job (in seconds).

"Thinking" time Time used by the user of an interactive system in order to make up the next request (in seconds).

The main purpose of the scheduler is to optimize these times.

CPU scheduling

Most of the processes are not using efficiently their allocated time. The processors' utilization is made in burst cycles like the one from the figure below:



Scheduling algorithms - examples

First in, first out (aka: *First come, first served*):

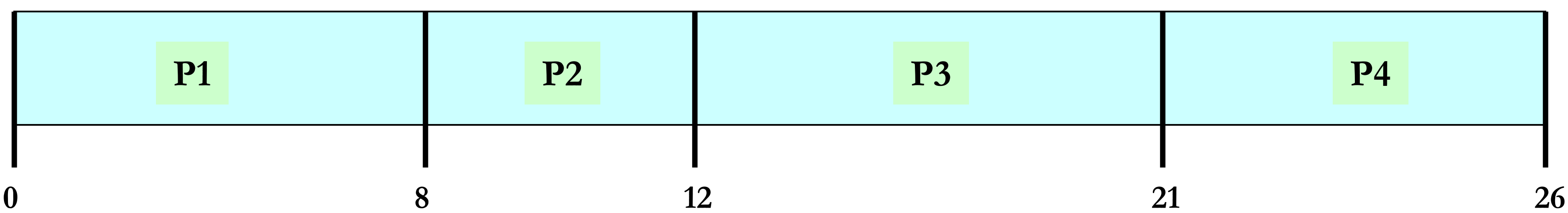
- FIFO
- Simple and fair but with weak performances. The medium waiting time in queue can be pretty high.

Scheduling algorithms – FIFO example

Example:

Process	Arrival time	Service time
1	0	8
2	1	4
3	2	9
4	3	5

FIFO



Average residence time = ((8-0) + (12-1) + (21-2) + (26-3))/4 = 61/4 = 15.25

CPU residence time

Scheduling algorithms – SJF example

Shortest Job First (SJF):

- The algorithm is optimal for minimizing the queue waiting time but impossible to implement in practice. Next process to come must be foreseen by the historical basis.
- Time prediction that the process will use for the next scheduling:

$$t(n+1) = w * t(n) + (1 - w) * T(n)$$

where:

where:	t(n+1)	next burst time
	t(n)	actual burst time
	T(n)	previous burst average
	w	weighted factor reflecting current or previous bursts

Scheduling algorithms – preemptive approach

Features for *preemptive algorithms*:

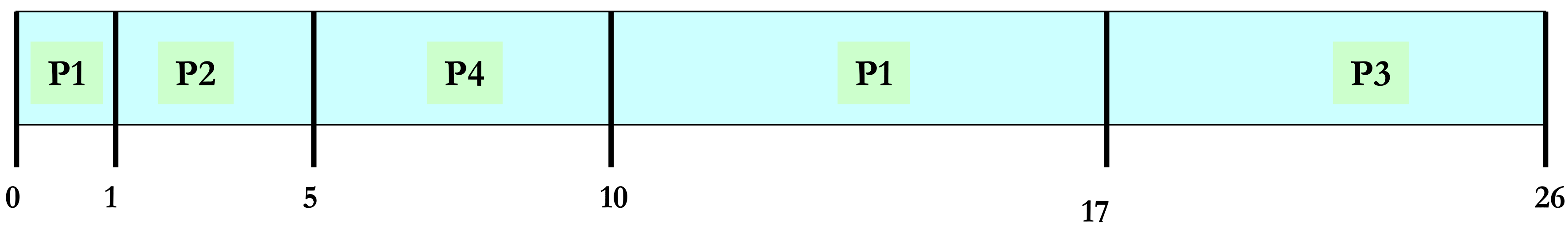
- In this case, the process is stopped from execution when another process with a higher priority is ready for execution
- It can be applied in case of SJF and in case of priority based scheduling
- **It avoids CPU monopolization by a single process**
- In case of *time sharing* it is necessary to implement this technique because the CPU must be protected by the processes with small priorities
- If shorter jobs have higher priority, the response time is better

Scheduling algorithms

Example:

Process	Arrival time	Service time
1	0	8
2	1	4
3	2	9
4	3	5

Preemptive SJF



Average residence time = ((5-1) + (10-3) + (17-0) + (26-2))/4 = 52/4 = 13.0

Priority based scheduling

Priority based scheduling:

- Each process is assigned a priority. The scheduler selects the first process with the highest priority. All the processes with the same priority are in a FIFO list.
- The priority can be set by the user or by a default mechanism. The system can determine the priority function of memory needs, time limits or other constraints.
- **Starvation** – it's the phenomenon that appears when a process with a lower priority never executes. Implementation solution: a variable that will store the "age".
- It must maintain a balance between "saying yes" to interactive jobs without having "starvation" for batch jobs.

Round Robin scheduling

ROUND ROBIN

- It uses a timer to generate an interrupt after a specified period of time. The preemptive multitasking is assured if a task exceeds the allocated quantum.
- In slide 15 we are testing the previous example for a quantum = 4 sec.
- Definitions:
  - **Context switching –** Modifying the running state from one process to another (memory change).
  - **Processor sharing–** Using a quantum such that every process will run for a maximum amount of time specified by that quantum.
  - **Rescheduling latency –** represents the waiting time from the moment when a process makes a running request till the moment it is running.

Round Robin scheduling (cont.)

ROUND ROBIN

A quantum is selected:

- If it's **too short** – too much time will be lost by context switching
- If it's **too long** – rescheduling latency will be too big. If many processes want to run, then a lot of time is lost as in the FIFO case.
- It is adjusted such that most of the processes won't use their running time.

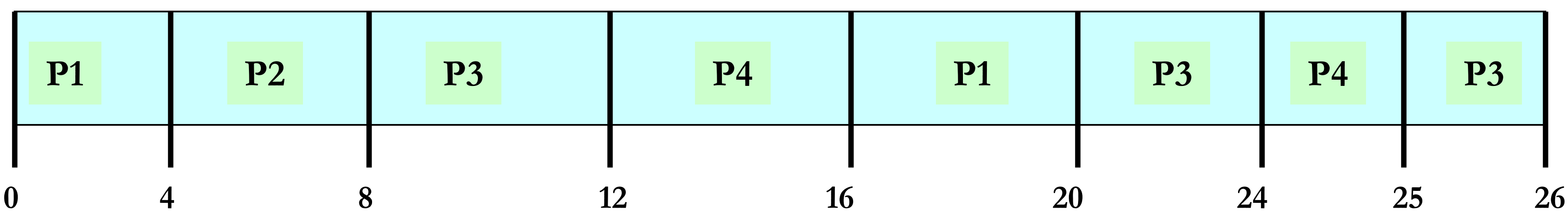
More info about scheduling and quantum in Windows:
https://www.microsoftpressstore.com/articles/article.aspx?p=2233328&seqNum=7

Scheduling algorithms - examples

Example:

Process	Arrival time	Service time
1	0	8
2	1	4
3	2	9
4	3	5

Round Robin, quantum = 4, no priorities

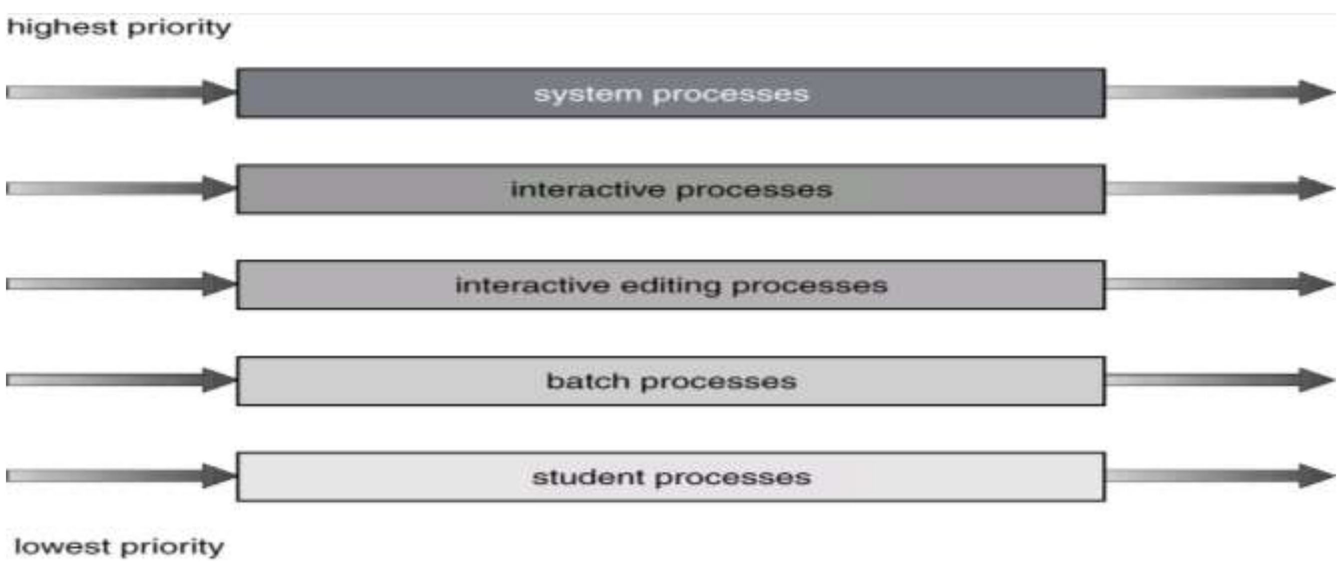


Average waiting time = ((20-0) + (8-1) + (26-2) + (25-3))/4 = 74/4 = 18.5

Scheduling algorithms – multi level queues

Multi level queues

- Each queue has its own scheduling algorithm
- Another algorithm (based on priorities) ensure the arbitration between queues
- Complex method, but flexible
- In this way different processes are separated: system processes, interactive, batch, favored or not-favored



Process priorities example

Windows priorities:

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

Process priorities

About creating, monitor and kill processes in Linux :

https://developer.ibm.com/tutorials/l-lpic1-103-5/

Process execution priorities in Linux:

https://developer.ibm.com/technologies/linux/tutorials/l-lpic1-103-6/

CPU scheduling – more info

Wikipedia about CPU scheduling:

- http://en.wikipedia.org/wiki/Scheduling_(computing)

Processes, Threads, Jobs, Quantum in Windows:

- https://www.microsoftpressstore.com/articles/article.aspx?p=2233328&seqNum=7