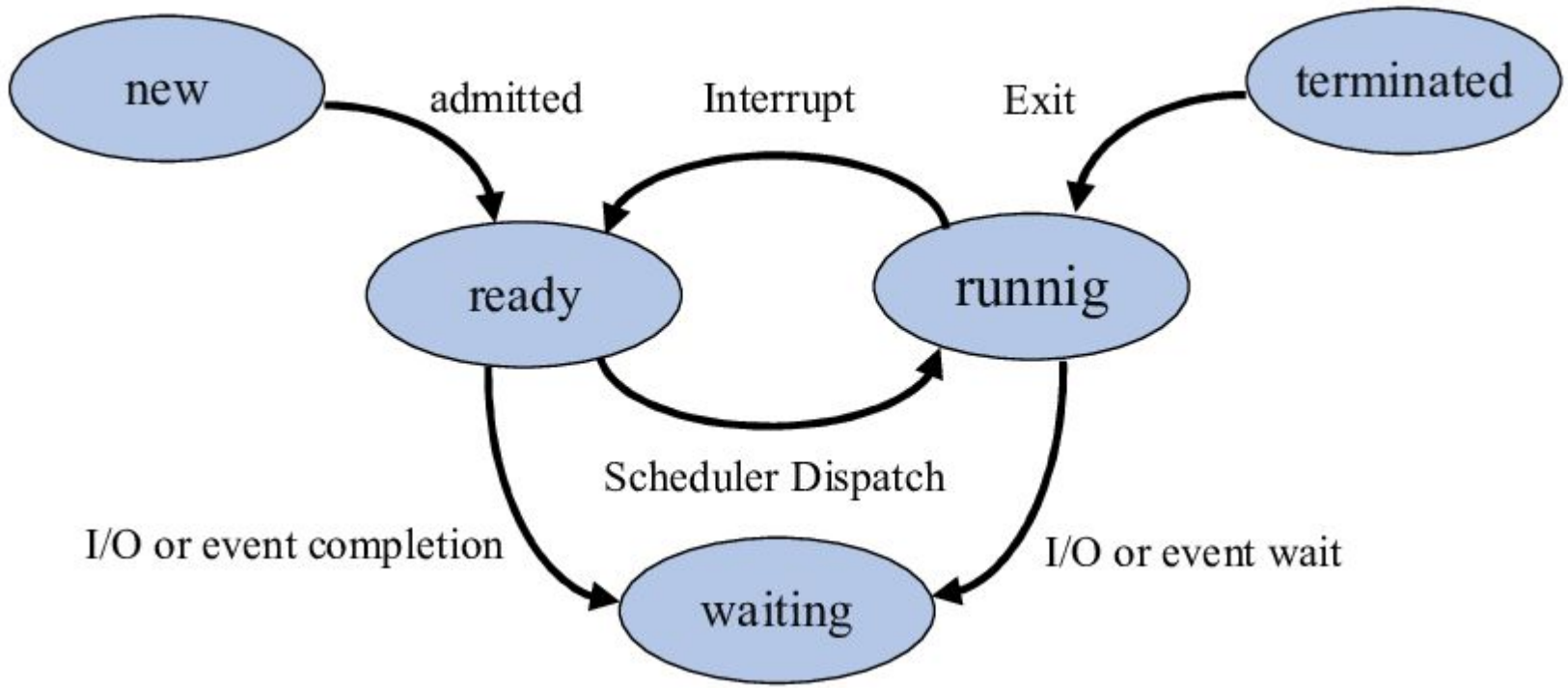
Process Management

What is process?

- A process is an active program.
- Process means Program in execution.

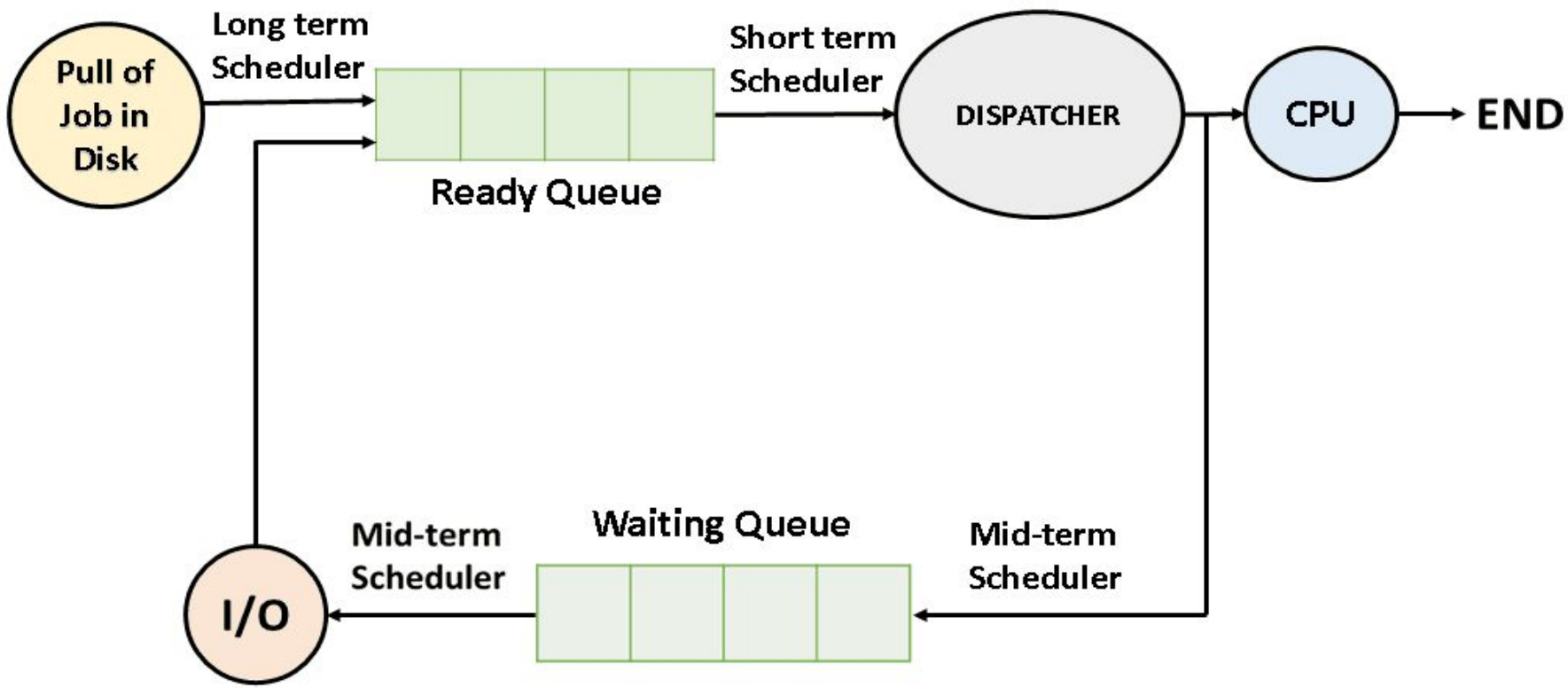
Process State diagram



Process State diagram

- **New**- The process is in new state when it has just been created.
- **Ready** - The process is waiting to be assigned the processor by the short term scheduler.
- **Running** - The process instructions are being executed by the processor.
- **Waiting** - The process is waiting for some event such as I/O to occur.
- **Terminated** - The process has completed its execution.

Process Scheduling



Created by NotesJam

- **Long Term Scheduler**

The job scheduler or long term scheduler selects processes from the storage pool and loads them into memory for execution. The job scheduler must select a careful mixture of I/O bound and CPU bound processes to yield optimum system throughput. If it selects too many CPU bound processes then the I/O devices are idle and if it selects too many I/O bound processes then the processor has nothing to do.

- **Short Term Scheduler**

The short term scheduler selects one of the processes from the ready queue and schedules them for execution. The short term scheduler executes much more frequently than the long term scheduler as a process may execute only for a few milliseconds.

- **Medium Term Scheduler**

The medium term scheduler swaps out a process from main memory. It can again swap in the process later from the point it stopped executing. This is helpful in reducing the degree of multiprogramming. Swapping is also useful to improve the mix of I/O bound and CPU bound processes in the memory.

Context Switching

Removing a process from a CPU and scheduling another process requires saving the state of the old process and loading the state of the new process. This is known as context switching. The context of a process is stored in the Process Control Block (PCB) and contains the process register information, process state and memory information.

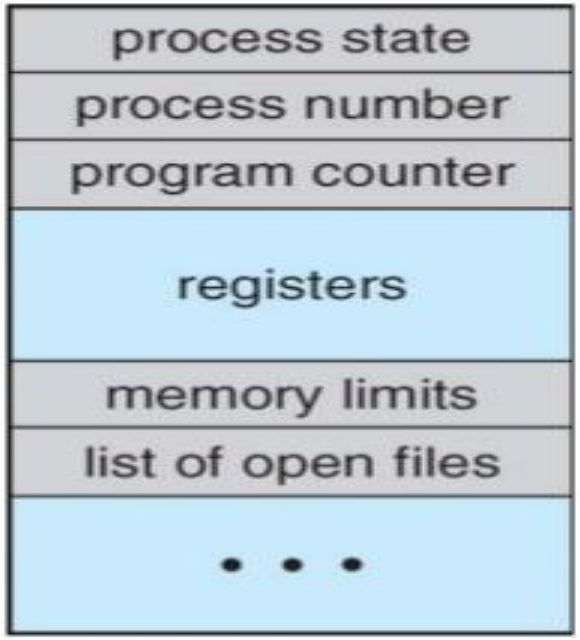
Process Control Block

Process Control Block (PCB)

Information associated with each process

(also called **task control block**)

- Process state – running, waiting, etc
- Program counter – location of instruction to next execute
- CPU registers – contents of all process-centric registers
- CPU scheduling information- priorities, scheduling queue pointers
- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files



CPU Scheduling Criteria

- **CPU utilization –**
  The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically, CPU utilization can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the load upon the system.
- **Throughput –**
  A measure of the work done by CPU is the number of processes being executed and completed per unit time. This is called throughput. The throughput may vary depending upon the length or duration of processes.

- **Turnaround Time (TAT):**

1. It is the time interval from the time of submission of a process to the time of the completion of the process.
2. Difference b/w Completion Time and Arrival Time is called Turnaround Time.

TAT = CT - AT

- **Completion Time (CT):** This is the time when the process completes it's execution.
- **Arrival Time (AT):** This is the time when the process has arrived in the ready state.

$$TAT = CT - AT$$

- **Waiting Time (WT):**

1. The time spent by a process waiting in the ready queue for getting the CPU.
2. The time difference b/w Turnaround Time and Burst Time is called Waiting Time.

Burst Time (BT): This is the time required by the process for it's execution.

Preemptive Scheduling	Non-Preemptive Scheduling
Resources are allocated according to the cycles for a limited time.	Resources are used and then held by the process until it gets terminated.
The process can be interrupted, even before the completion.	The process is not interrupted until its life cycle is complete
starvation may be caused, due to the insertion of priority process in the queue.	Starvation can occur when a process with large burst time occupies the system.
Maintaining queue and remaining time needs storage overhead.	No such overheads are required
SRTF, RR, Priority	FCFS, SJF, Priority

FCFS(First come first serve)

J1(Job)	15(Execution time)
J2	8
J3	10
J4	3

J1(wj1=0)	J2(wj2=15)	J3(wj3=23)	J4(wj4=33	

Average waiting time= Wj1 +Wj2 +Wj3+Wj4= 0+15+23+33=71
71/4=17.75

SJF(Shortest job first)

Job	CPU time
J4(Wj4=0)	3
J2(Wj2=3)	8
J3(Wj3)=11	10
J1(Wj1)=21	15

J4		J2		J3		J1	
0	3	3	11	11	21	21	36

Average waiting time= 0 +3+11+21=35/4=8.75

CPU Scheduling Criteria

- **Turnaround time**
- **Waiting time**
- **Response time :**In an interactive system, turn-around time is not the best criteria. A process may produce some output fairly early and continue computing new results while previous results are being output to the user. Thus another criteria is the time taken from submission of the process of request until the first response is produced. This measure is called response time.

Shortest Remaining Time First (SRTF) algorithm

- Shortest Remaining Time First (SRTF) is the preemptive version of Shortest Job Next (SJF) algorithm, where the processor is allocated to the job closest to completion.

Shortest Remaining Time First (SRTF) algorithm(Example-1)

Process	Arrival time(AT)	Burst Time(BT)	CT(Completion)	TAT(CT-AT)	WT(TAT-BT)
P1	0	8	17	17	9
P2	1	4	5	4	0
P3	2	9	26	24	15
P4	3	5	10	7	2
				52	26

P1		P2		P4		P1		P3	
0	1	1	5	5	10	10	17	17	26

Shortest Remaining Time First (SRTF) algorithm (Example-1 Continue…)

Average TAT= 52/4=13

Average WT= 26 /4=6.5

Shortest Remaining Time First (SRTF) algorithm(Example-2)

Consider set of process with arrival time (milliseconds), CPU burst time, service . As shown below none of the processes has I/O burst time.

Process.No	A.T	B.T	C.T	T.A.T(CT-AT)	W.T (TAT- BT)
P1	0	7	19	19	12
P2	1	5	13	12	7
P3	2	3	6	4	1
P4	3	1	4	1	0
P5	4	2	9	5	3
P6	5	1	7	2	1
				43	24

P1		P2		P3		P4		P3		P3		P6		P5		P2		P1	
0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	9	9	13	13	19

Calculate average waiting time and turnaround time using SRTF scheduling algorithm

Average TAT= 43/6= 7.16

Average WT= 24/6= 4

Priority Scheduling

Non Preemptive Priority Scheduling:

In the Non Preemptive Priority scheduling, The Processes are scheduled according to the priority number assigned to them. Once the process gets scheduled, it will run till the completion.

Non Preemptive Priority Scheduling(Example-1)

Process ID	Priority	Arrival Time	Burst Time
P1	2	0	3
P2	6	2	5
P3	3	1	4
P4	5	4	2
P5	7	6	9
P6	4	5	4
P7	10	7	10

P1	P3	P6	P4	P2	P5	P7
0 3	3 7	7 11	11 13	13 18	18 27	27 37

Non Preemptive Priority Scheduling(Example-1 continue…..)

Process Id	Priority	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time	Response Time
P1	2	0	3	3	3	0	0
P2	6	2	5	18	16	11	13
P3	3	1	4	7	6	2	3
P4	5	4	2	13	9	7	11
P5	7	6	9	27	21	12	18
P6	4	5	4	11	6	2	7
P7	10	7	10	37	30	18	27

P1	P3	P6	P4	P2	P5	P7
0 3	3 7	7 11	11 13	13 18	18 27	27 37

Preemptive Priority Scheduling

- In Preemptive Priority Scheduling, at the time of arrival of a process in the ready queue, its Priority is compared with the priority of the other processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time. The One with the highest priority among all the available processes will be given the CPU next.

- The difference between preemptive priority scheduling and non preemptive priority scheduling is that, in the preemptive priority scheduling, the job which is being executed can be stopped at the arrival of a higher priority job.

- Once all the jobs get available in the ready queue, the algorithm will behave as non-preemptive priority scheduling, which means the job scheduled will run till the completion and no preemption will be done.

Preemptive Priority Scheduling(Example-1)

P.ID	Priority	A.T	B.T				
P1	2	0	4				
P2	4	1	2				
P3	6	2	3				
P4	10	3	5				
P5	8	4	1				
P6	12	5	4				
P7	9	6	6				

Preemptive Priority Scheduling(Example-2)

Pid	A.T	B.T	Priority	C.T	T.A.T	W.T
P1	1	4	4			
P2	2	2	5			
P3	2	3	7			
P4	3	5	8			
P5	3	1	5			
P6	4	2	6			

Problem in Priority Scheduling:

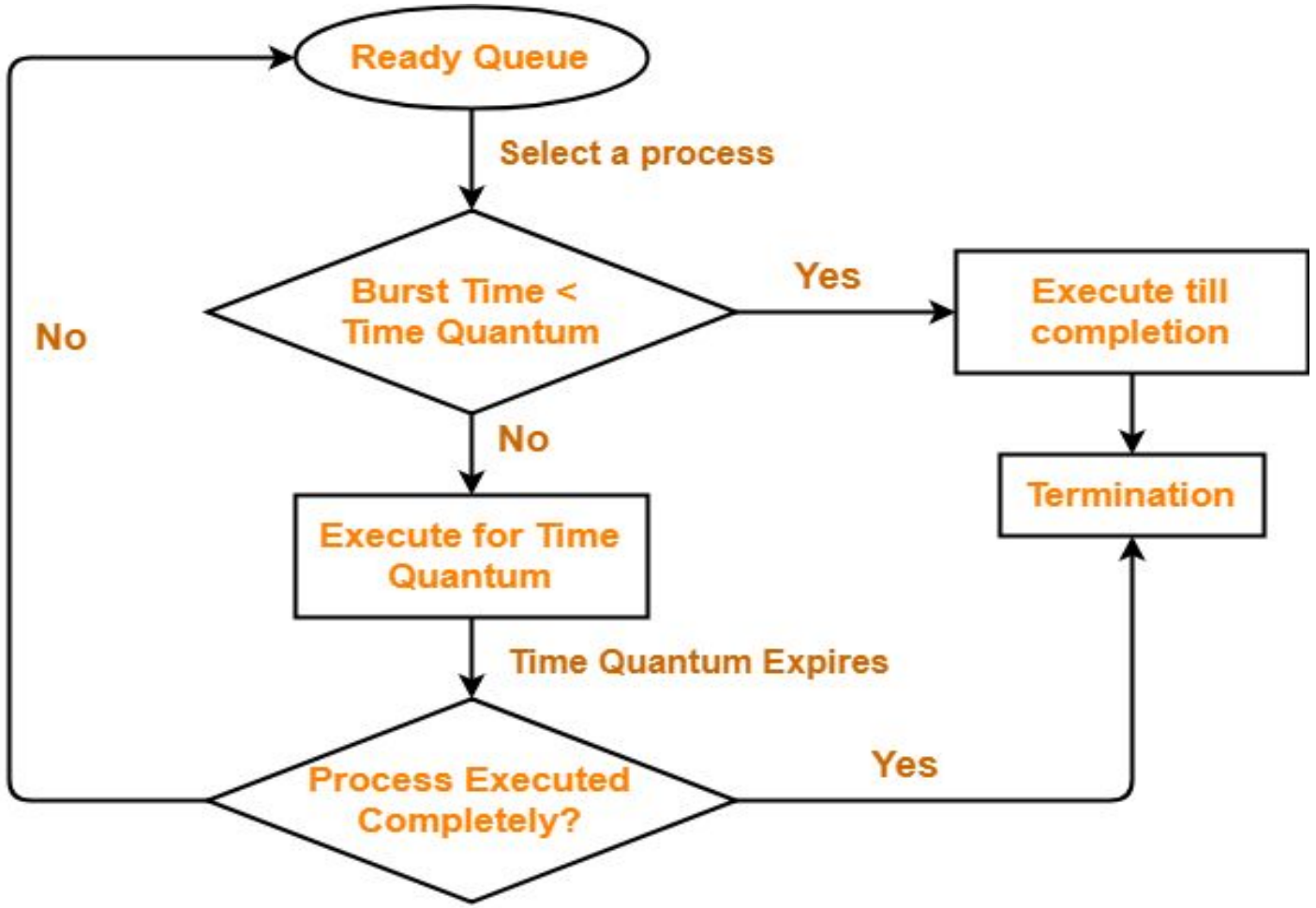
Problem in Priority Scheduling:

- low priority processes may never execute. This is called starvation.

Solution:

- The solution to this starvation problem is **aging**. In aging, as time progresses, increase the priority of the process so that the lowest priority process gets converted to the highest priority gradually.

Round Robin scheduling



Round Robin Scheduling

Round Robin scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

Round Robin scheduling (Example-1)

- Time Quantum=5

Pid	B.T	C.T	TAT	WT
P1	21			
P2	3			
P3	6			
P4	2			

AVG Waiting time=

Round Robin scheduling (Example-2)

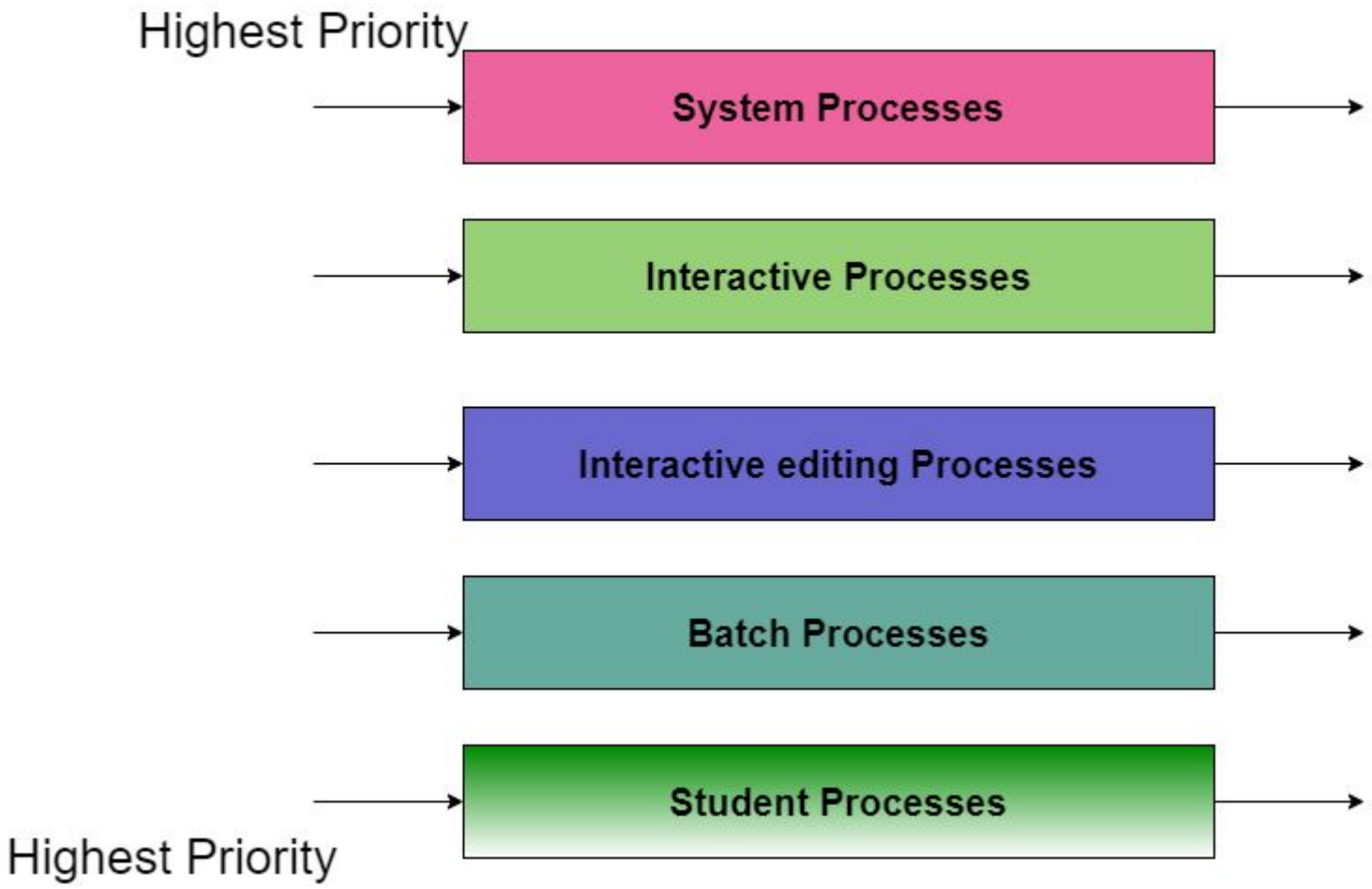
- Time quantum =2

Pid	A.T	B.T	C.T	TAT	WT
P1	0	4			
P2	1	5			
P3	2	2			
P4	3	1			
P5	4	6			
P6	5	3			

Multilevel Queue (MLQ) CPU Scheduling

- It may happen that processes in the ready queue can be divided into different classes where each class has its own scheduling needs. For example, a common division is a **foreground (interactive)** process and **background (batch)** processes. These two classes have different scheduling needs. For this kind of situation Multilevel Queue Scheduling is used.

Multilevel Queue (MLQ) CPU Scheduling



Advantages of Multilevel Queue Scheduling

With the help of this scheduling we can apply various kind of scheduling for different kind of processes:

- **For System Processes**: First Come First Serve(FCFS) Scheduling.
- **For Interactive Processes**: Shortest Job First (SJF) Scheduling.
- **For Batch Processes**: Round Robin(RR) Scheduling
- **For Student Processes**: Priority Scheduling

Disadvantages of Multilevel Queue Scheduling

The main disadvantage of Multilevel Queue Scheduling is the problem of starvation for lower-level processes.

Let us understand **what is starvation?**

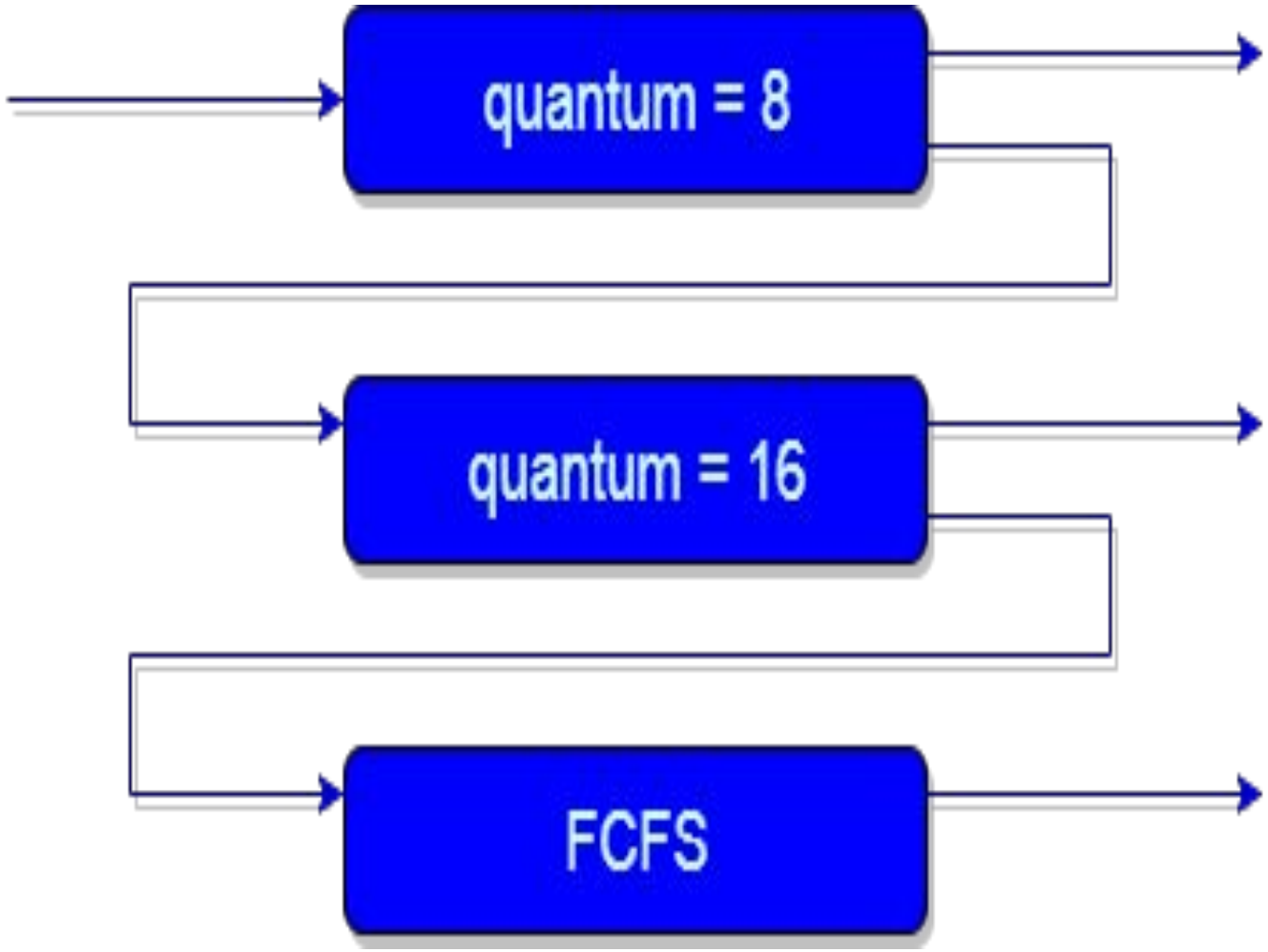
- **Starvation:** Due to starvation lower-level processes either never execute or have to wait for a long amount of time because of lower priority or higher priority process taking a large amount of time.

Multilevel Feedback Queue Scheduling

- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system. Processes do not move between queues. This setup has the advantage of low scheduling overhead, but the disadvantage of being inflexible.

- Multilevel feedback queue scheduling, however, allows a process to move between queues. The idea is to separate processes with different CPU-burst characteristics. If a process uses too much CPU time, it will be moved to a lower-priority queue. Similarly, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. This form of aging prevents starvation.

Multilevel Feedback Queue Scheduling



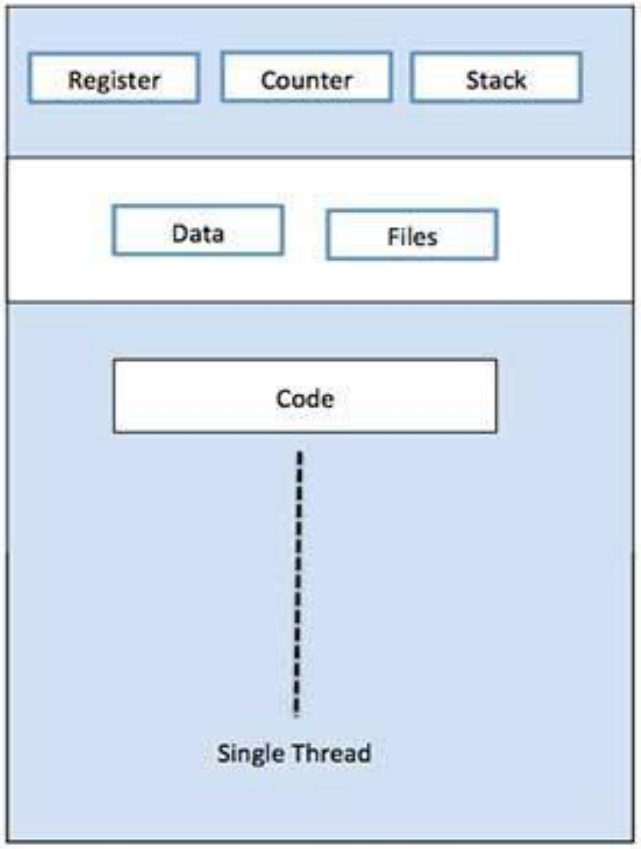
What is Thread?

- A thread is a path of execution within a process. A process can contain multiple threads.
- A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.
- A thread shares with its peer threads few information like code segment, data segment and open files. When one thread alters a code segment memory item, all other threads see that.
- A thread is also called a **lightweight process**. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

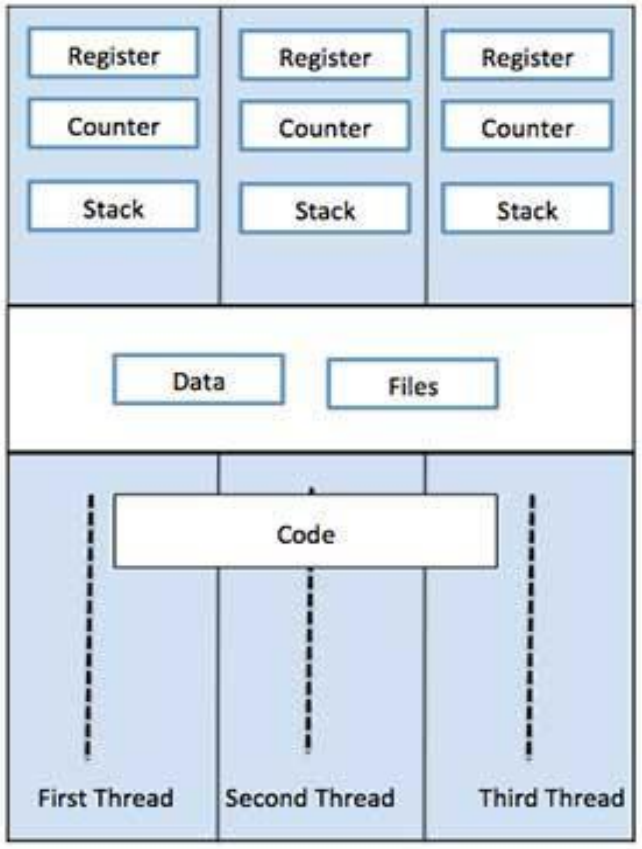
What is Thread?

- Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. The following figure shows the working of a single-threaded and a multithreaded process.

What is Thread?



Single Process P with single thread



Single Process P with three threads

Difference between Process and Thread

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Advantages of Thread

- Threads minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

Types of Thread

Threads are implemented in following two ways −

- **User Level Threads** − User managed threads.
- **Kernel Level Threads** − Operating System managed threads acting on kernel, an operating system core.