Unit #2

Process and Threads Management

Marwadi University

Department of Computer Engineering

Operating System
Sem 4
3140702
5 Credits

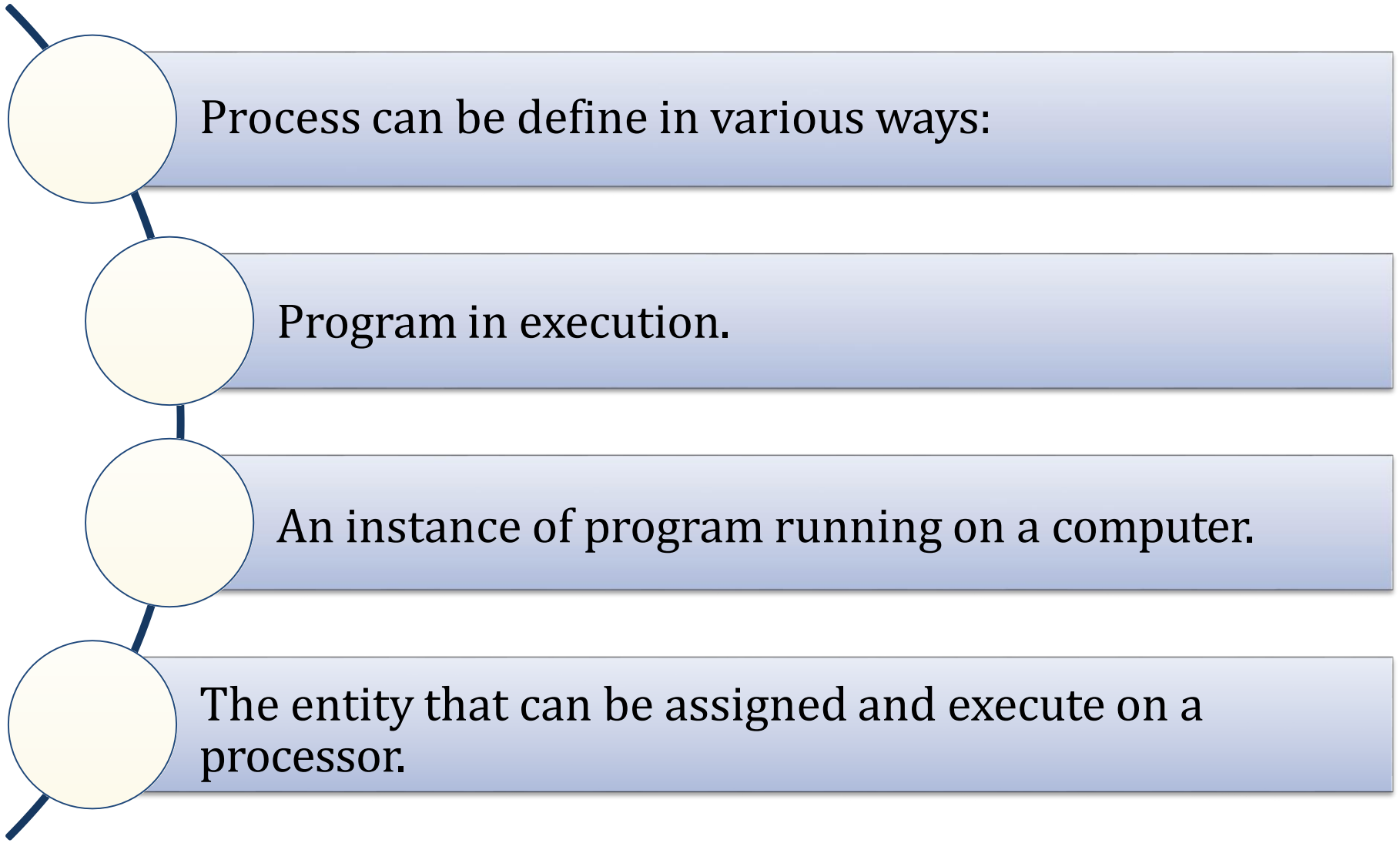
Prof. Chetan Chudasama

Topics to be covered

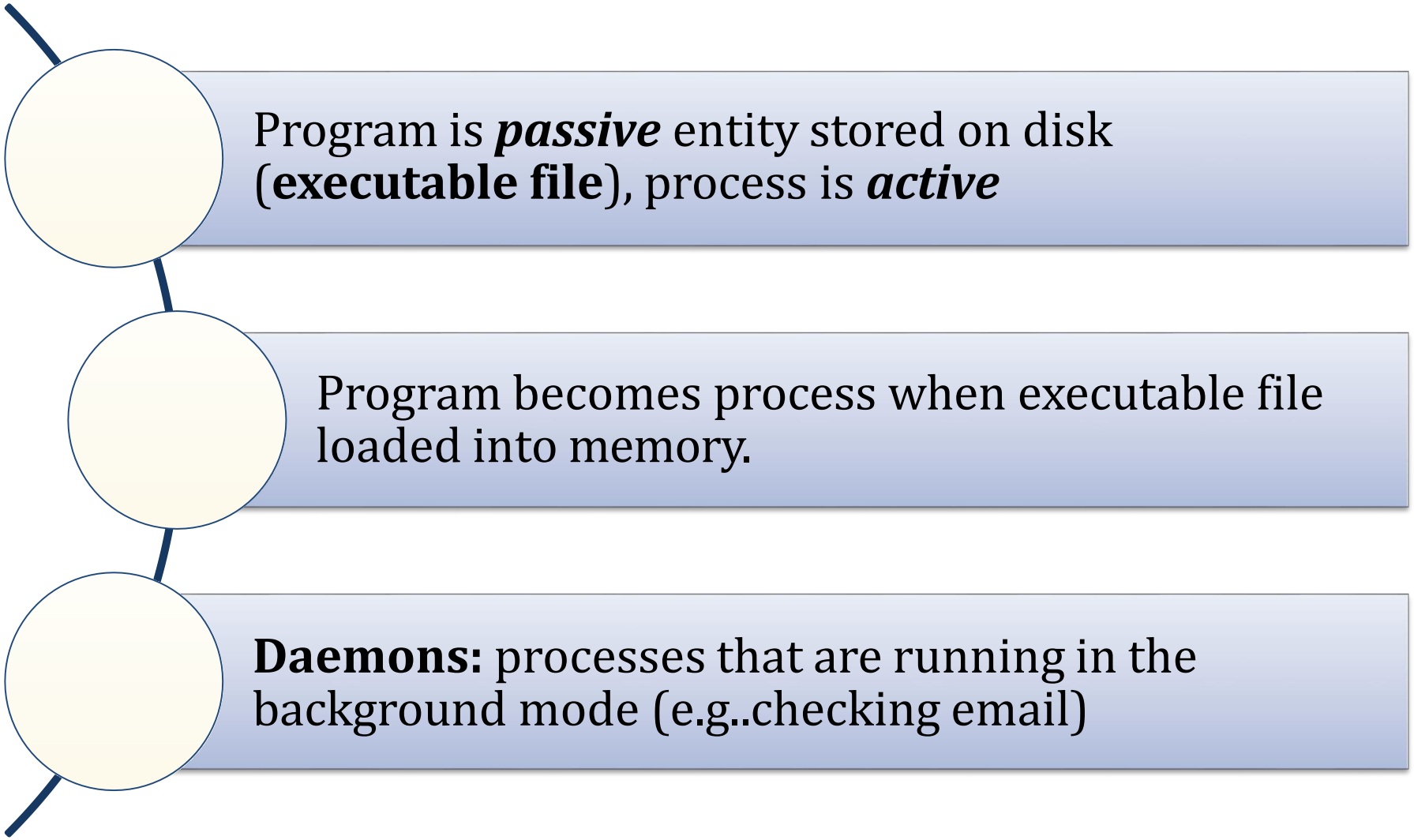
Marwadi University

- Concept of Process
- Process Relationship, Process states
- Process State transitions
- Process Control Block
- Context switching
- Concept of Threads, Thread Scheduling
- Types of Schedulers, Scheduling criteria
- CPU Scheduling algorithms : FCFS, SJF, RR and Priority
- Types of Scheduling: Preemptive and Non- preemptive

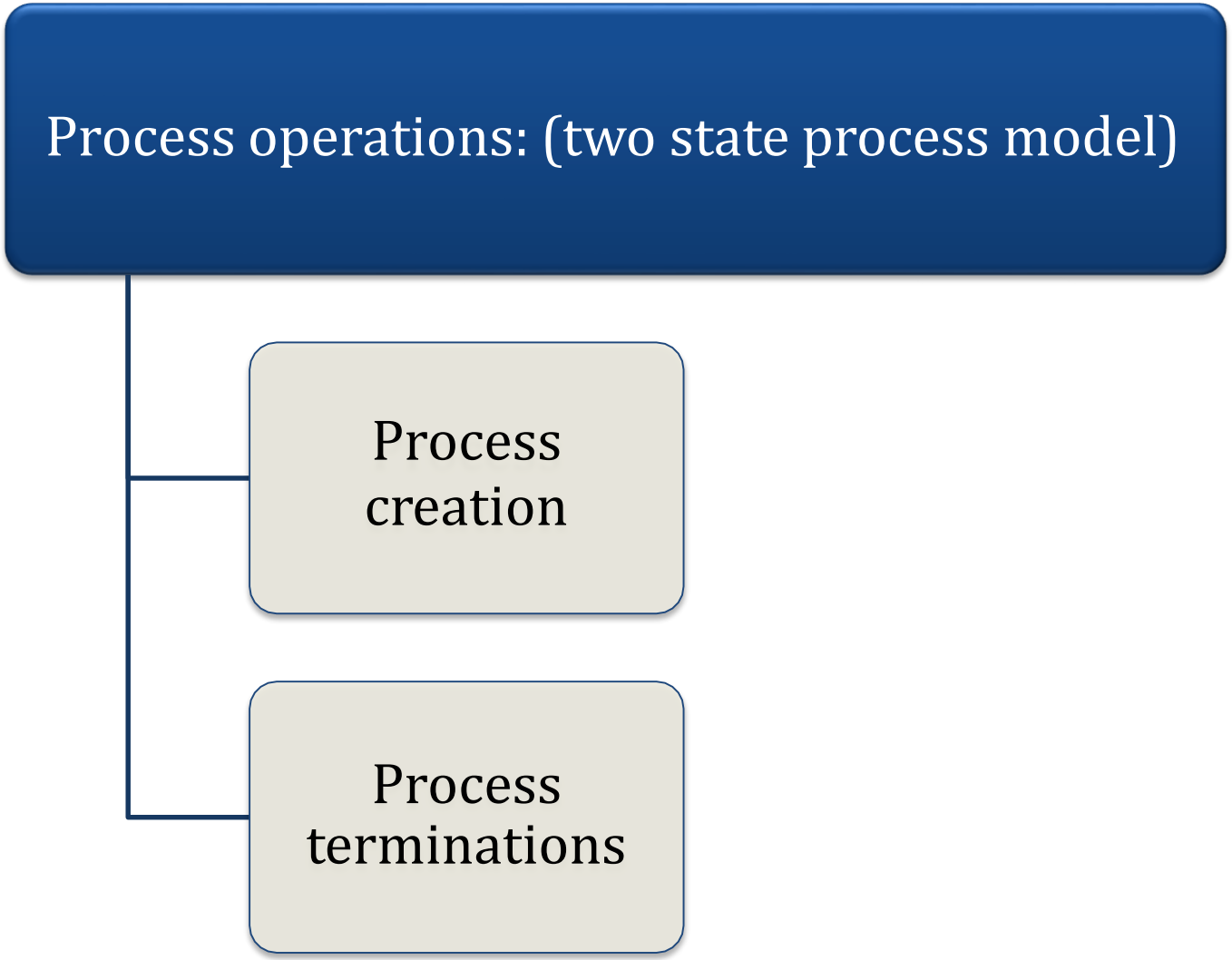
Concept of Process



Concept of Process



Two State Process Model



Process operations- process creation

Marwadi University

System initialization:

- When OS is booted, several system process are created. They provide system services. They do not need user interaction it just execute in background.

Execution of a process creation system call:

- A running process can issue system call to create new process. i.e. in Unix system call name “fork” is used to create new process.

Process operations- process creation

A user request to create a new process

- User can start new process by typing command or double click on some application icon.

Initiation of a batch job:

- This applies only to the batch system. Here user submit batch jobs to the system. When there are enough resources are free to execute a next job a new job is selected from batch queue.

Process operations- process termination

Marwadi University

Normal exit (voluntary)

- Terminate when done their job. i.e. when all the instructions from program get executed.
- Event user can select option to terminate the process.

Error exit (voluntary):

- Process even terminated when some error occurred. Example divide by zero error.

Process operations- process termination

Marwadi University

Fatal error (involuntary):

- Fatal error are generated due to user mistake in executing program. Such as file name not found error.

Killed by another process (involuntary):

- If some other process request for OS to kill the process. Then UNIX it can be done by “kill” system call while in windows “Terminate Process”

Process Relationship

Marwadi University

One to one:

- when *single execution of sequential program* is in progress.
- Program consist of main program and set of functions, during execution control flows between main program and set of functions.
- OS is not aware of the existence of function. So program consist of single process.

Many to one:

- Many simultaneous executions of program.
- Program informs OS about its parts that are to be executed concurrently thus OS consider each part as process.
- So one program any many process.
- This process are know as concurrent process.

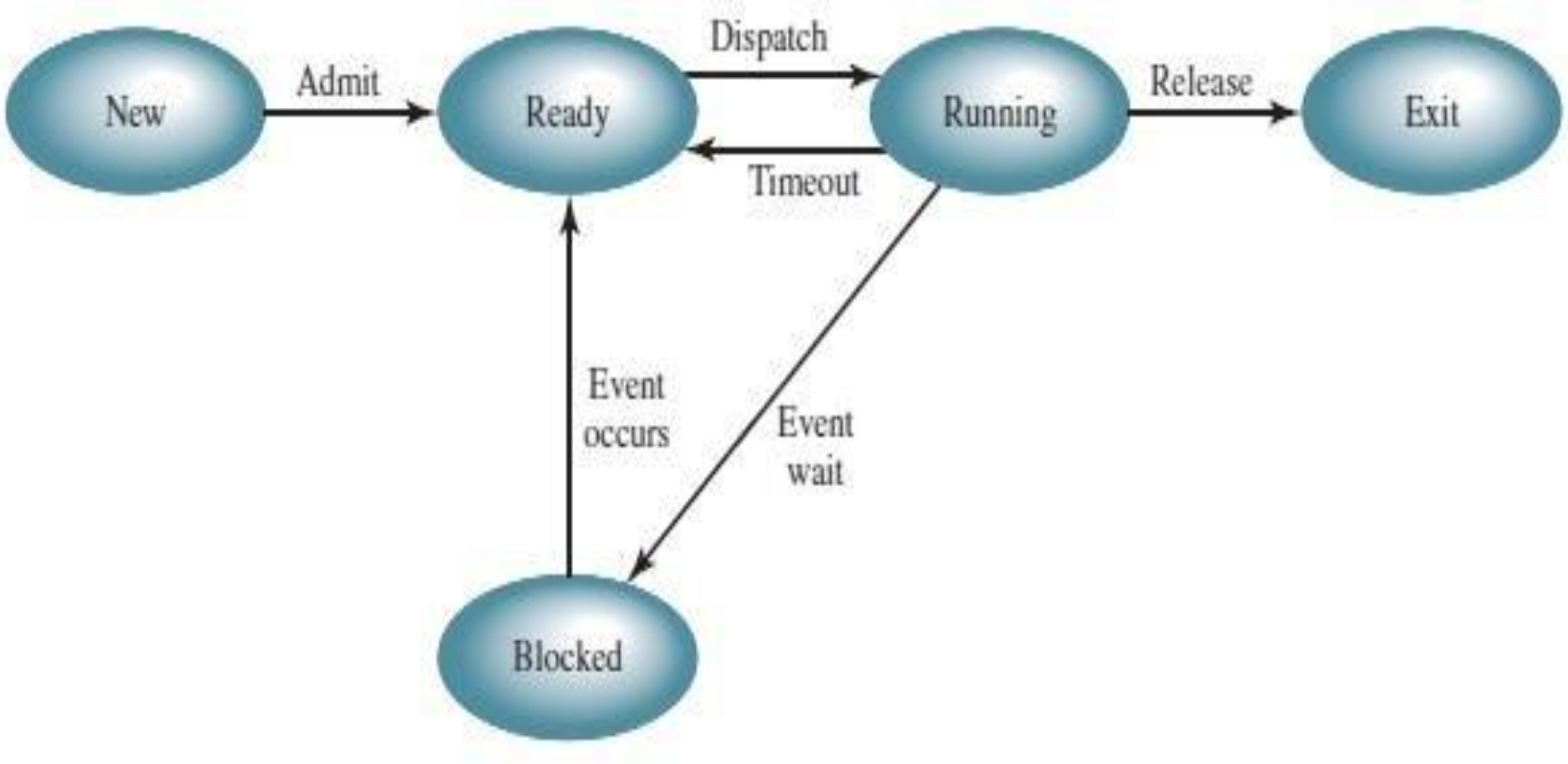
Process States

- Process state is an indicator of the nature of current activity in process.

State	Meaning
New	The process is being created
Running	CPU currently executing instructions of process.
Waiting/ blocked	Process has to wait until requested resource is granted or process is waiting for some event to occur (i.e. input from user)
Ready	The process is waiting to be assigned to a processor (CPU)
Terminated	The process has finished execution or OS has aborted it.

Process State Transitions

- A state transitions for process p is change in its state.
- **Five state process model:**

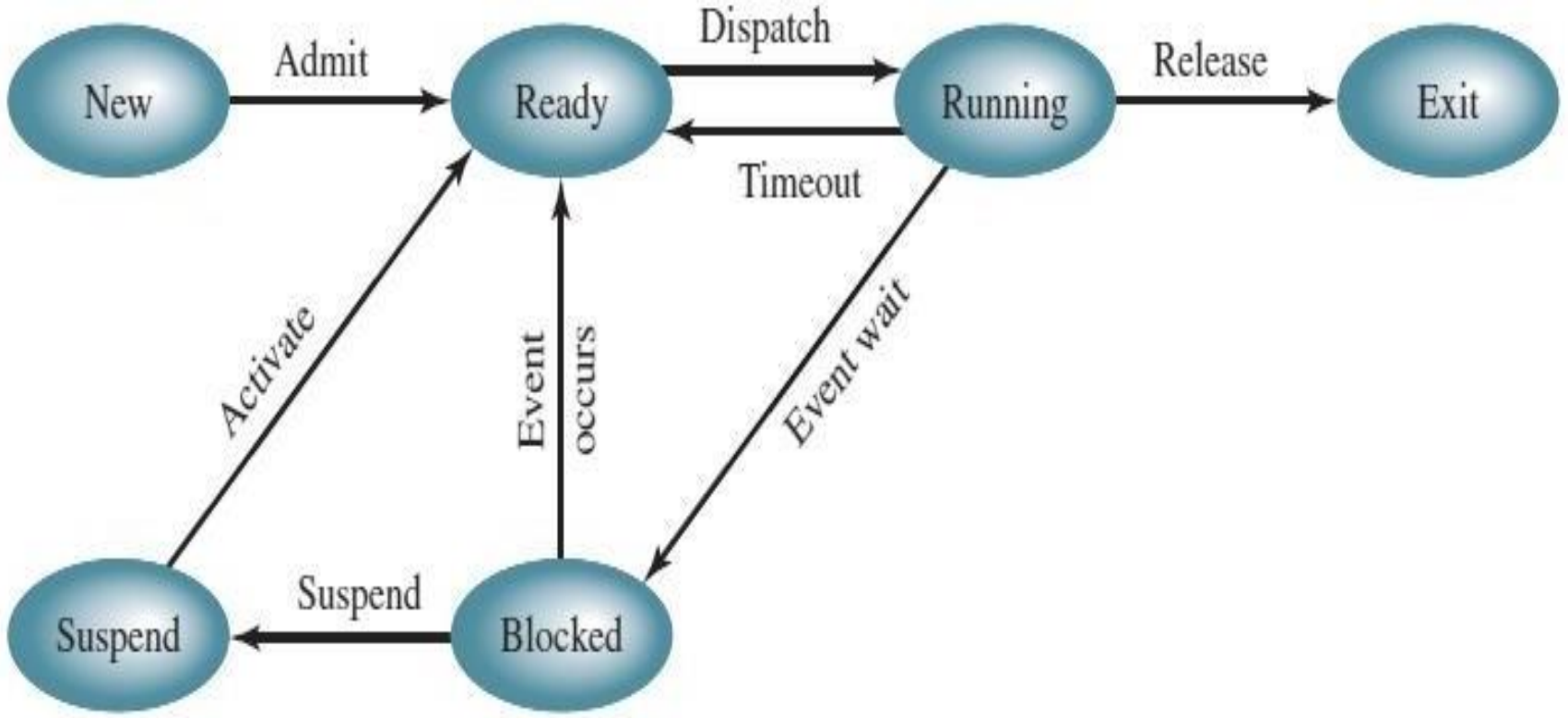


Process State Transitions

Marwadi University

State Transition	Description
Ready → Running	CPU start or resumes its execution
Blocked → Ready	Request made by process is satisfied or event for which process was waiting occurs.
Running →Blocked	Process request for I/O Request for memory or some other resources. Waits for message from other process.

Process State Transitions



Process State transitions

Marwadi University

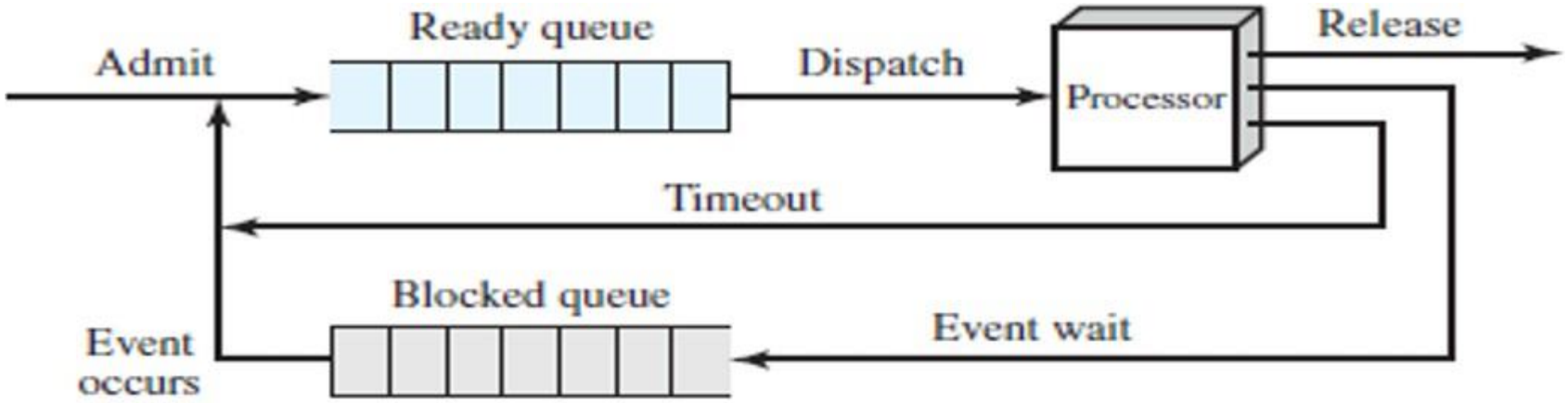
Reason for suspension of process:

- **Swapping:** moving process from memory to disk.
- **Other OS reasons:** OS may suspend process that is suspected of causing problem.
- **Interactive user request:** a user may wish to suspend execution of program for debugging.
- **Timing:** time out.
- **Parent process request:** parent process may suspend execution of child process.

Time	*Event*	*Remarks*	*New state*	
			P_1	P_2
0		P_1 is scheduled	*running*	*ready*
10	P_1 is preempted	P_2 is scheduled	*ready*	*running*
20	P_2 is preempted	P_1 is scheduled	*running*	*ready*
25	P_1 starts I/O	P_2 is scheduled	*blocked*	*running*
35	P_2 is preempted		*blocked*	*ready*
		P_2 is scheduled	*blocked*	*running*
45	P_2 starts I/O		*blocked*	*blocked*
125	I/O interrupt	P_1 becomes *ready*	*ready*	*blocked*
		P_1 is scheduled	*running*	*blocked*

Queue Diagram

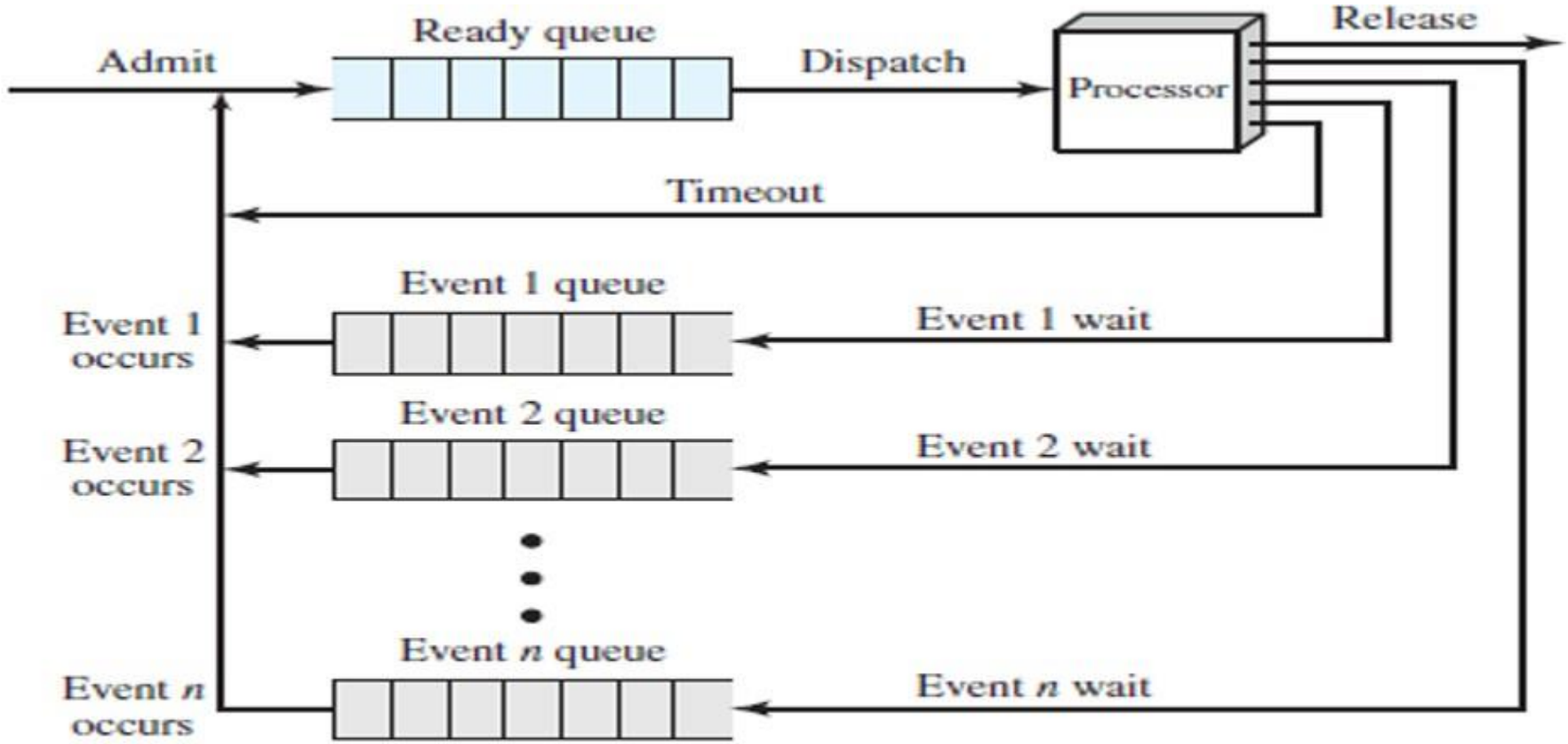
- For single Waiting Queue



(a) Single blocked queue

Queue Diagram

- For multiple waiting Queue



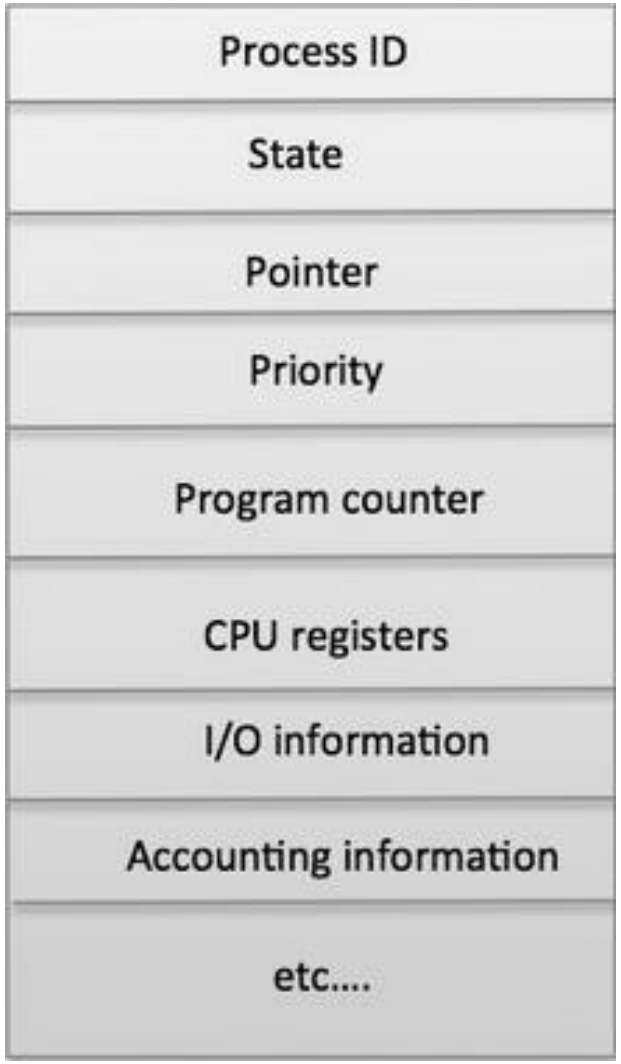
(b) Multiple blocked queues

Process Control Block (PCB)

- To implement the process model, the operating system maintains a table called the process table.
- Process table is *array* of structure
- Another name for process table is process control block (PCB)
- PCB is the key to multiprogramming.
- PCB must be saved when a process is switched from *running* to *ready* or *blocked* states.

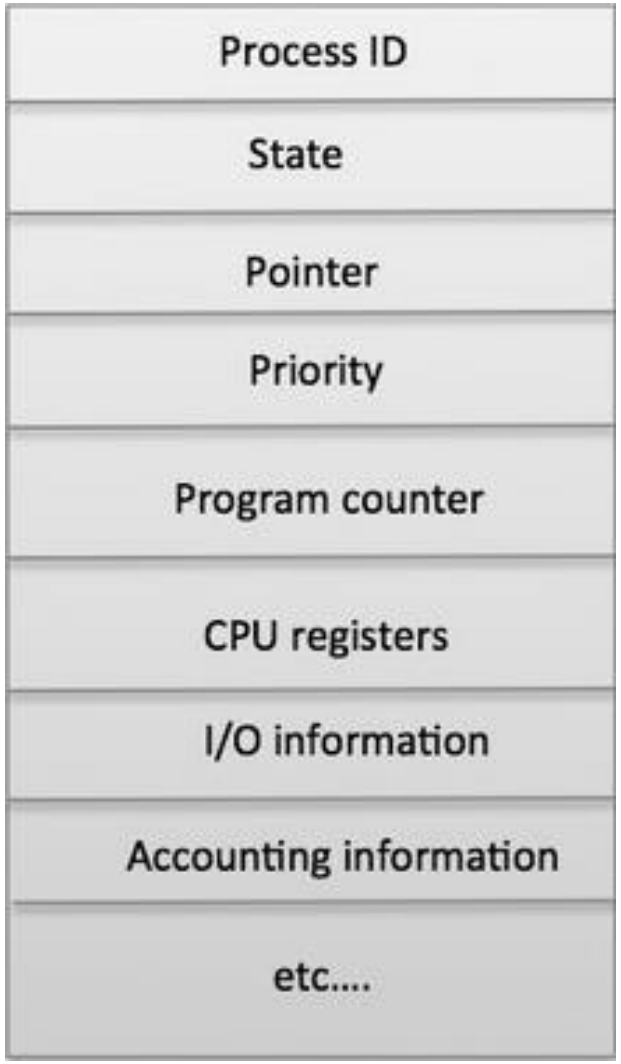
PCB Fields

- **Process ID** - **Unique identification** for each of the process in the operating system.
- **Process State** - The **current state** of the process i.e., whether it is ready, running, waiting.
- **Pointer** - A **pointer to parent process**.
- **Priority** - **Priority** of a process, its typically numeric value. A process is assigned a priority at its creation.
- **Program Counter** - Program Counter is a **pointer** to the **address** of the **next instruction** to be **executed** for this process.



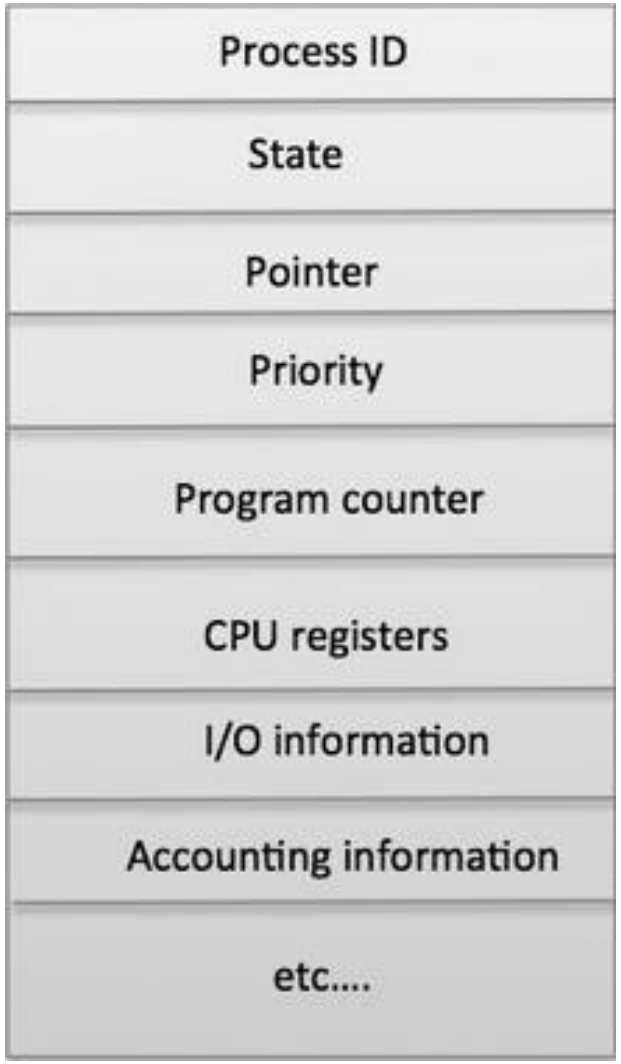
PCB Fields

- **CPU registers** – contains the content of the register when the **CPU was last released** by the process.
- **IO status information** - This includes a list of **I/O devices allocated** to the process.
- **CPU scheduling information** - it includes **process priority**, pointer to **various scheduling queue**, **information about events on which process** is waiting and other parameters.



PCB Fields

- **Program Status Word (PSW)** - **this is snapshot** i.e. the image of the PSW when the **CPU was last voluntarily leave by the process**.
- **Memory management information -** it includes values of **base and limit registers,** information **about page table or segment table.**
- **Event Information** - for a process in the **blocked state**, this field contains information about event for which the process is waiting, **when an event occurs** kernel uses this information.

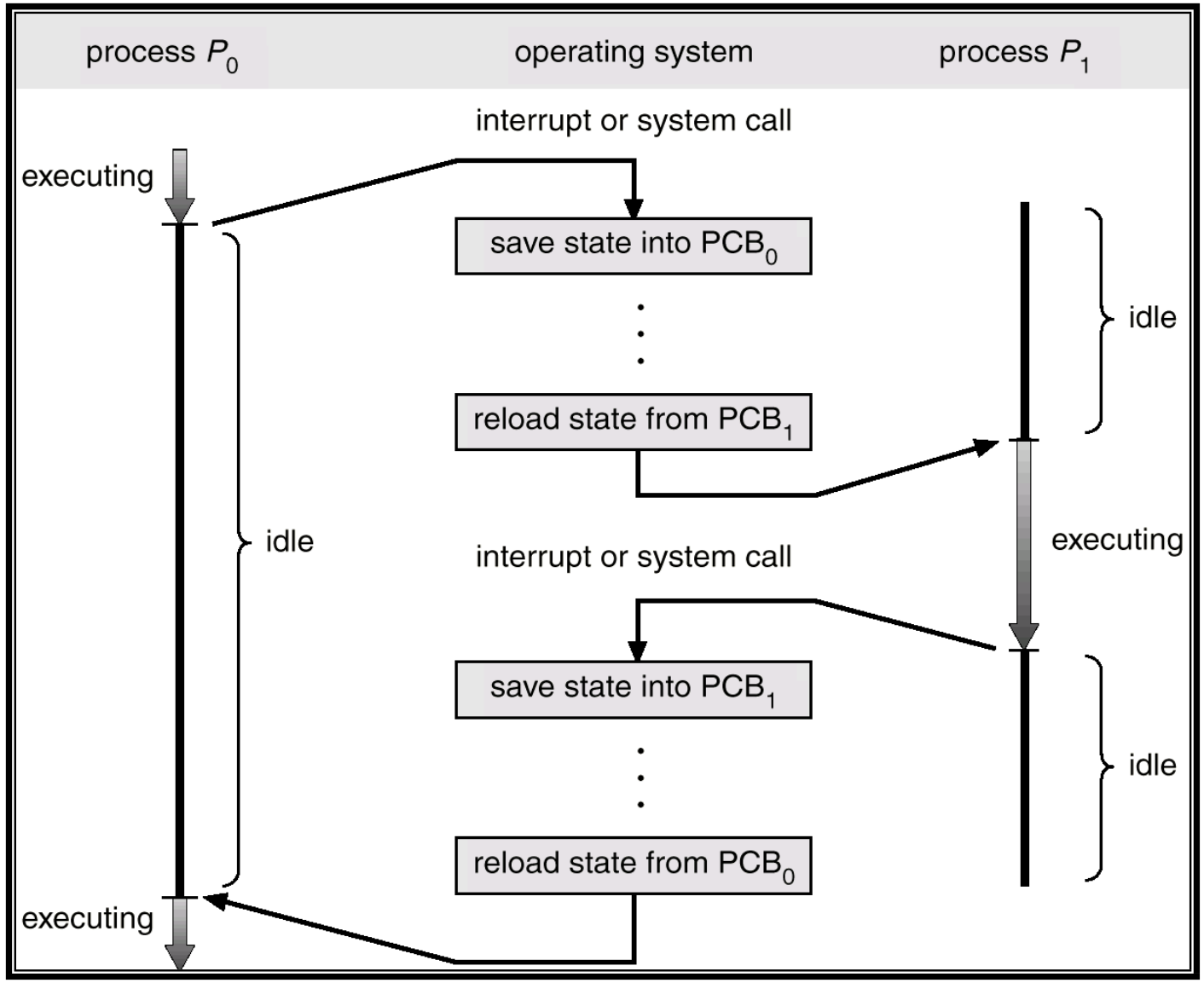


Context Switching

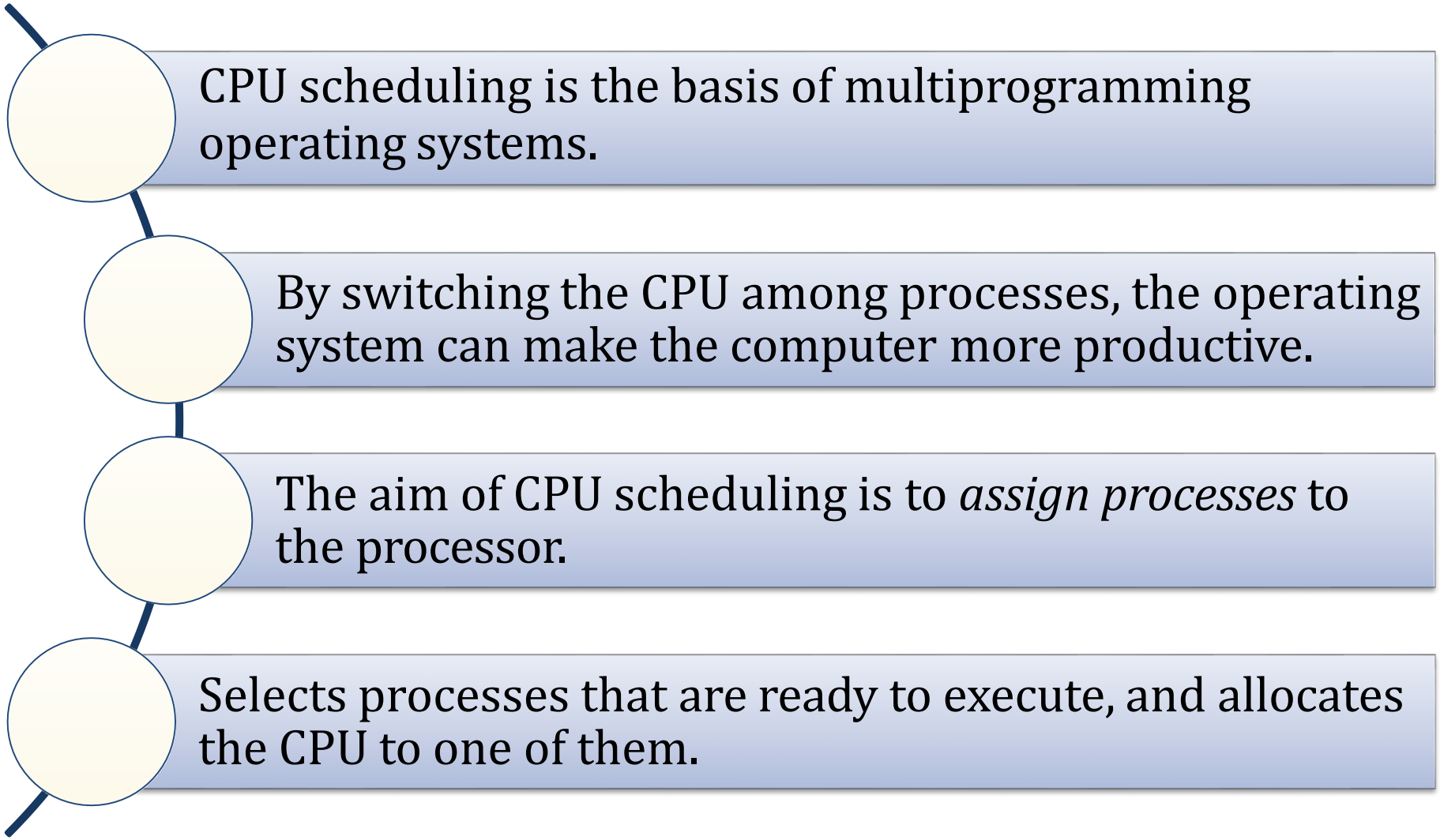
Marwadi University

- This needed when multiple processes to share a single CPU.
- context is information stored in PCB.(e.g. program counter and other registers).
- a **context switch** is the process of storing and restoring the information of a process so that execution can be resumed from the same point at a later time.

Context Switching



Process Scheduling



Process Scheduling

Marwadi University

Need

- Switches from running to waiting state
- Switches from running to ready state
- Switches from waiting to ready
- Terminates

Types of Scheduler

Marwadi University

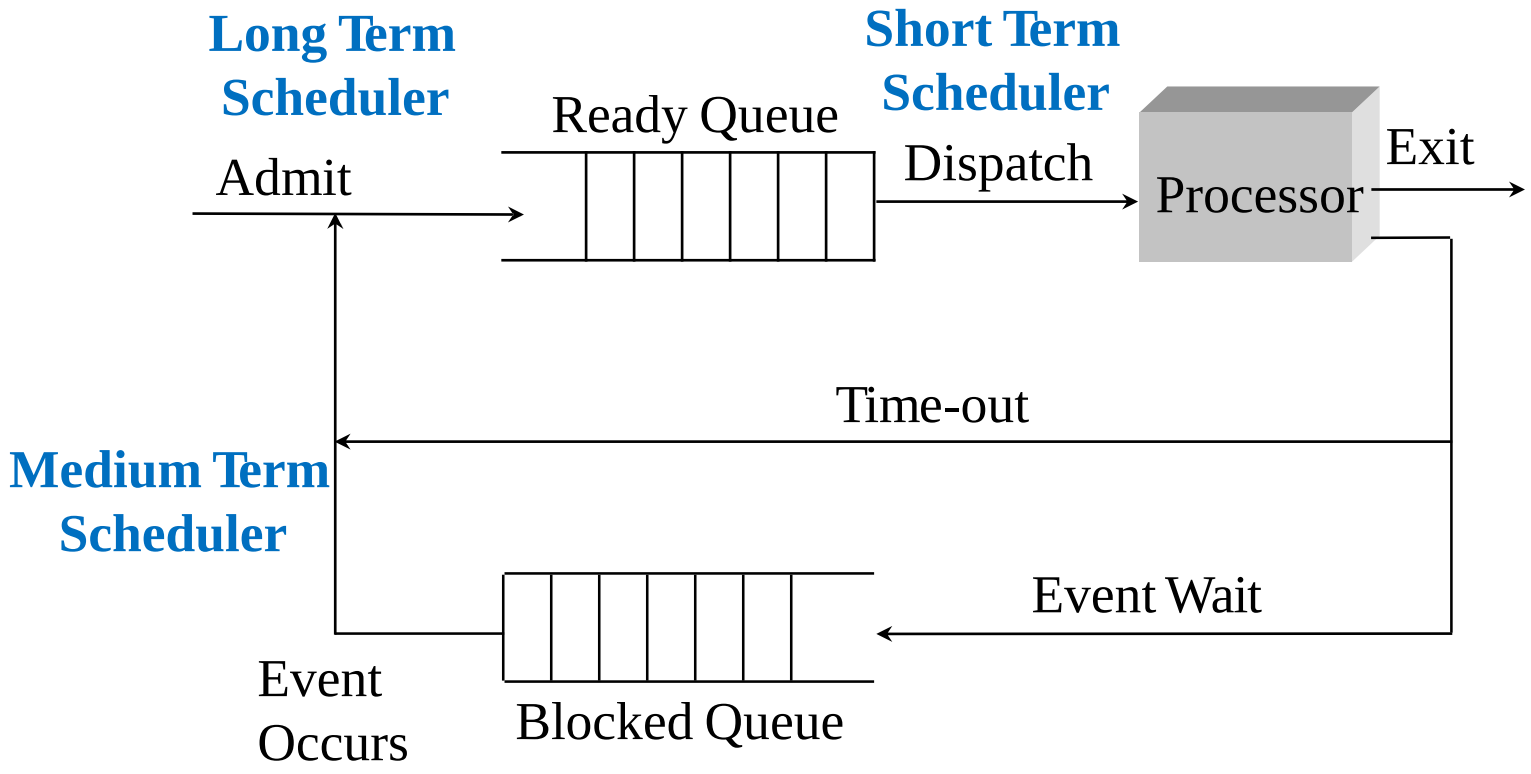
Long term scheduler

- When process is created.
- selects process and loads it into ready queue (memory) for execution.
- This is a decision whether to add a new process to the set of processes that are currently active.

Medium term scheduler

- Memory manager.
- Swap in, Swap out from main memory non-active processes.

Types of schedulers



Types of Scheduler

Marwadi University

Short term scheduler

- Deals with processes among ready processes.
- Very fast, similar to action at every clock interrupt
- if a process requires a resource (or input) that it does not have, it is removed from the ready list (and enters the WAITING state).
- Short-term scheduling is the actual decision of which ready process to execute next

CPU Scheduling Terminology

- **Burst Time/Execution Time/Running Time:** the time process requires for running on CPU.
- **Waiting Time:** time spend by process in ready state waiting for CPU.
- **Arrival Time:** when a process enters ready state.
- **Exit Time :** when process completes execution and exit from the system.
- **Turn around Time:** the total time spend by the process in the system.
- **Response Time:** the time between a process enters ready state and get scheduled on the CPU for the first time.

CPU Scheduling Criteria

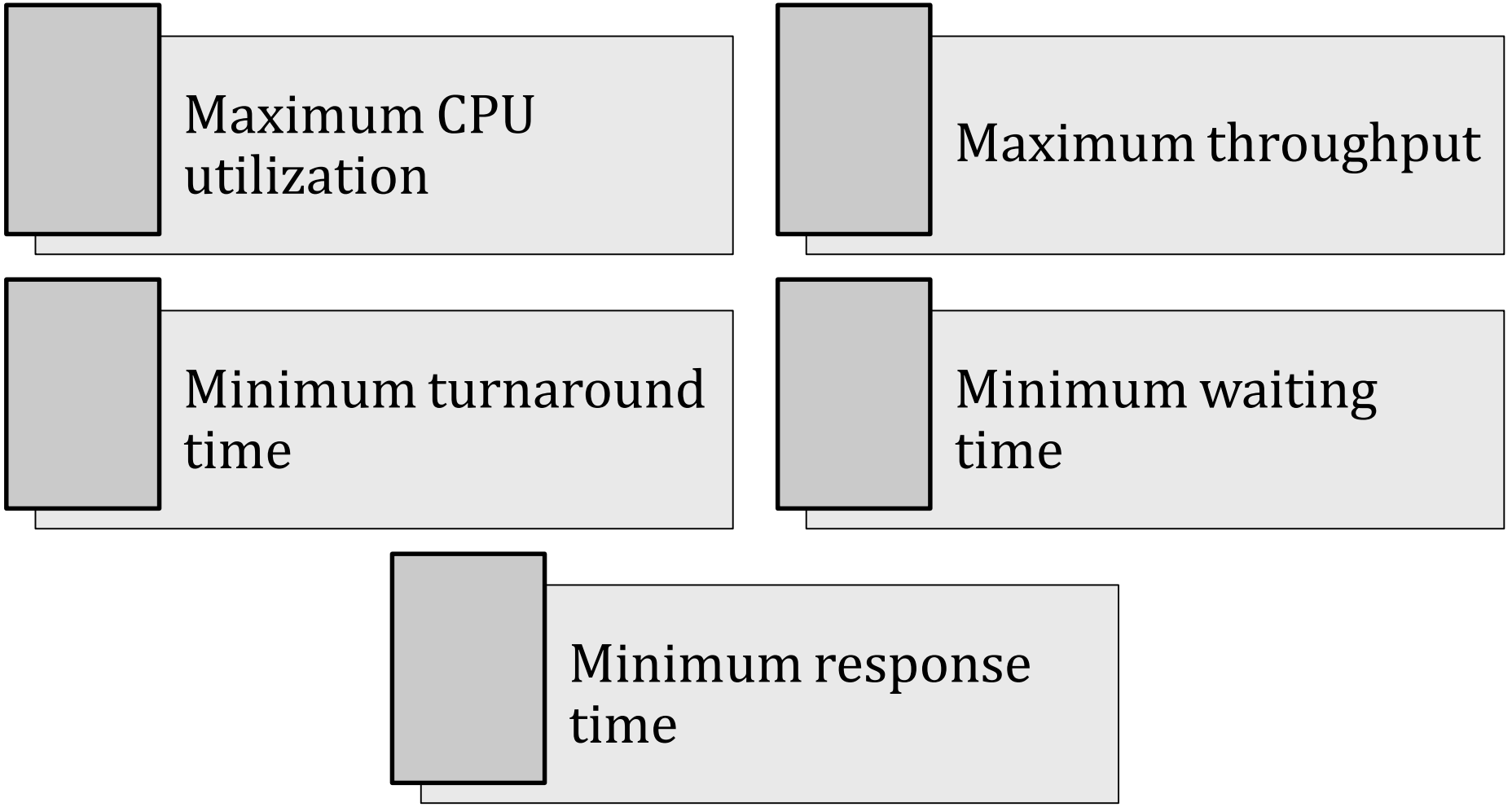
CPU utilization	• keep the CPU as busy as possible
Throughput	• number of processes that complete their execution per time unit
Turnaround time	• amount of time to execute a particular process, that is the interval from the time of submission of a process to the time of completion is the turnaround time. • TAT= Exit Time – Arrival Time OR • TAT= Burst time+ Waiting Time

CPU Scheduling Criteria

Marwadi University

Waiting time	• amount of time a process has been waiting in the ready queue. • WT= TAT- Burst Time
Response time	• time from the submission of a request until the first response is produced or amount of time it takes from when a request was submitted until the first response is produced.

Scheduling Algorithm Objectives



CPU Scheduling - Non Pre-emptive

Marwadi University

Non Pre-emptive:

- Under non preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.

CPU Scheduling - Non Pre-emptive

Marwadi University

Pre-emptive:

- Once a process has been given the CPU can taken away due to higher priority process arrived or time out.

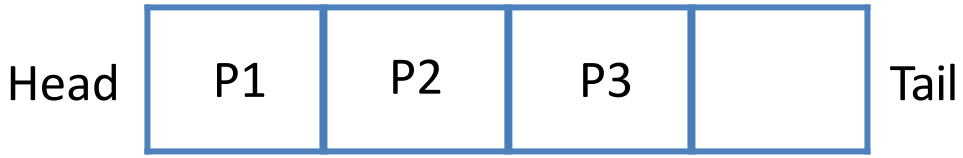
Scheduling algorithms

1. First Come First Served (FCFS)
2. Shortest Job First (SJF)
3. Shortest Remaining Time Next (SRTN)
4. Round Robin (RR)
5. Priority

First Come First Served (FCFS)

Marwadi University

- Selection criteria
  - The **process** that **request first is served first**.
  - It means that **processes are served in the exact order of their arrival**.



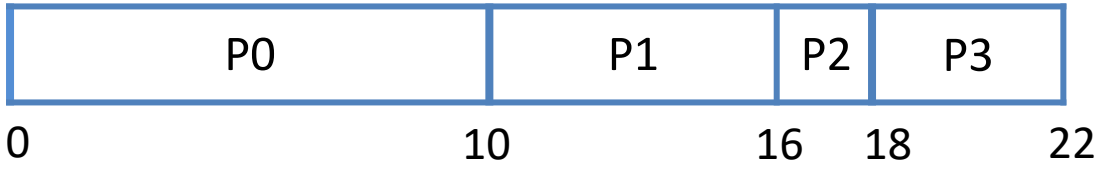
- Implementation:
  - **Non preemptive**: Once a process is selected, it runs until it is blocked for an I/O or some other event or it is terminated.
  - Always **Non-Pre-emptive** in nature.
  - This strategy can be **easily implemented** by using FIFO (First In First Out) queue.
  - When CPU becomes free, a process from the first position in a queue is selected to run.

First Come First Served (FCFS)

- Example

Process	Arrival Time (T0)	Time required for completion (ΔT) (CPU Burst Time)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

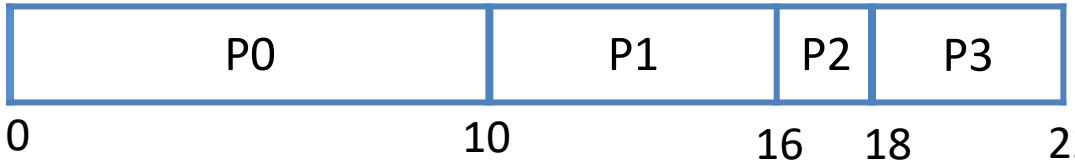
- Gantt Chart



First Come First Served (FCFS)

Process	Arrival Time (T0)	Burst Time (ΔT)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

- Gantt Chart



- Average Turnaround Time: (10+15+15+17)/4 = 14.25 ms.
- Average Waiting Time: (0+9+13+13)/4 = 8.75 ms.

First Come First Served (FCFS)

Marwadi University

- Advantages
  - **Simple** and **fair**.
  - **Easy to understand** and **implement**.
  - Every process will get a chance to run, so **starvation doesn't occur**.
- Disadvantages
  - **Not efficient** because average waiting time is too high.
  - **Convoy effect is possible**. All small I/O bound processes wait for one big CPU bound process to acquire CPU.
  - **CPU utilization may be less efficient** especially when a CPU bound process is running with many I/O bound processes.

FCFS Numerical

Marwadi University

Question - Find the average TAT and WT for the given processes by using FCFS scheduling algorithm.

Process	Arrival time(AT)	Burst Time (BT)
P1	0	24
P2	0	3
P3	0	3

Process	Arrival time(AT)	Burst Time (BT)
P1	0	10
P2	1	6
P3	3	2
P4	5	4

Solution-1

Process	Arrival time(AT)	Burst Time (BT)	Completion Time (CT)	Turn Around Time (TAT) TAT=CT-AT	Waiting Time(WT) WT= TAT-BT
P1	0	24	24	24	0
P2	0	3	27	27	24
P3	0	3	30	30	27

Average waiting time: (0 + 24 + 27)/3 = 17
Average Turn Around time: (24 + 27 + 30)/3 = 27

First-Come First-Served (FCFS) Scheduling

Process	Arrival time(AT)	Burst Time (BT)	Completion Time (CT)	Turn Around Time (TAT) TAT=CT-AT	Waiting Time(WT) WT= TAT-BT
P1	0	1	1	1	0
P2	1	100	101	100	0
P3	2	1	102	100	99
P4	3	100	202	199	99

Average waiting time: (0 + 0 + 99 + 99)/4 = 49.5
Average Turn Around time: (1 + 100 + 100 + 199)/4 = 100

Shortest Job First (SJF)

Marwadi University

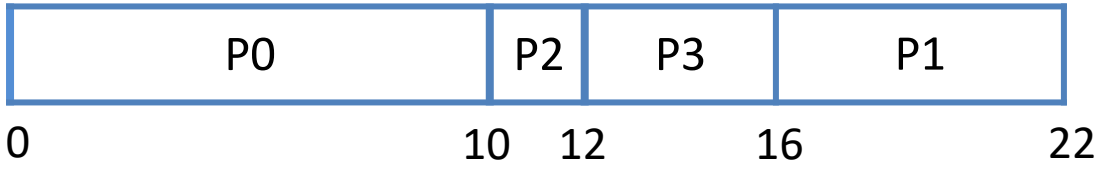
- Selection criteria
  - The process, that **requires shortest time to complete execution**, is **served first**.
- Decision Mode
  - **Non preemptive**: Once a process is selected, it runs until either it is blocked for an I/O or some other event or it is terminated.
- Implementation:
  - This strategy can be **easily implemented** by using FIFO (First In First Out) queue.
  - All processes in a queue are **sorted in ascending order** based on their required CPU bursts.
  - When CPU becomes free, a process from the first position in a queue is selected to run.

Shortest Job First (SJF)

- Example

Process	Arrival Time (T0)	Time required for completion (ΔT) (CPU Burst Time)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

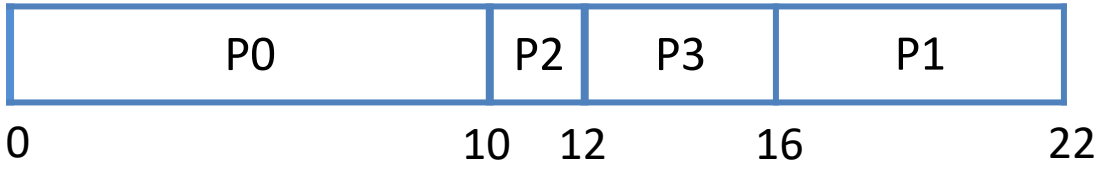
- Gantt Chart



Shortest Job First (SJF)

Process	Arrival Time (T0)	Burst Time (ΔT)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

- Gantt Chart



- Average Turnaround Time: (10+21+9+11)/4 = 12.75 ms.
- Average Waiting Time: (0+15+7+7)/4 = 7.25 ms.

Shortest Job First (SJF)

- Advantages:
  - **Less waiting time**.
  - **Good response** for **short processes**.
- Disadvantages :
  - It is **difficult to estimate time required** to complete execution.
  - **Starvation is possible for long process**. Long process may wait forever.

Shortest Remaining Time Next (SRTN)

Marwadi University

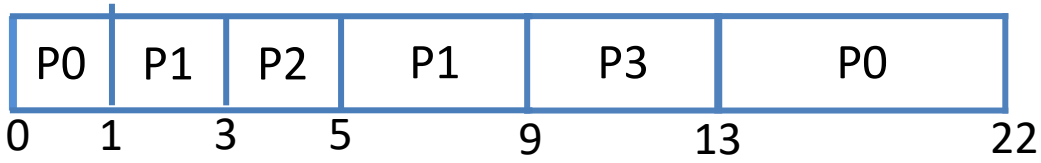
- Selection criteria :
  - The process, **whose remaining run time is shortest**, is **served first**. This is a **preemptive version of SJF scheduling**.
- Decision Mode:
  - **Preemptive**: When a new process arrives, its total time is compared to the current process remaining run time.
  - If the new process needs less time to finish than the current process, the current process is suspended and the new job is started.
- Implementation :
  - This strategy can also be implemented by using sorted FIFO queue.
  - All processes in a queue are **sorted in ascending order on their remaining run time**.
  - When CPU becomes free, a process from the first position in a queue is selected to run.

Shortest Remaining Time Next (SRTN)

- Example

Process	Arrival Time (T0)	Time required for completion (ΔT) (CPU Burst Time)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

- Gantt Chart

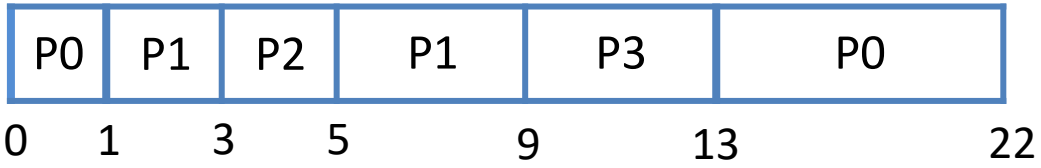


Shortest Remaining Time Next (SRTN)

Marwadi University

Process	Arrival Time (T0)	Burst Time (ΔT)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

- Gantt Chart



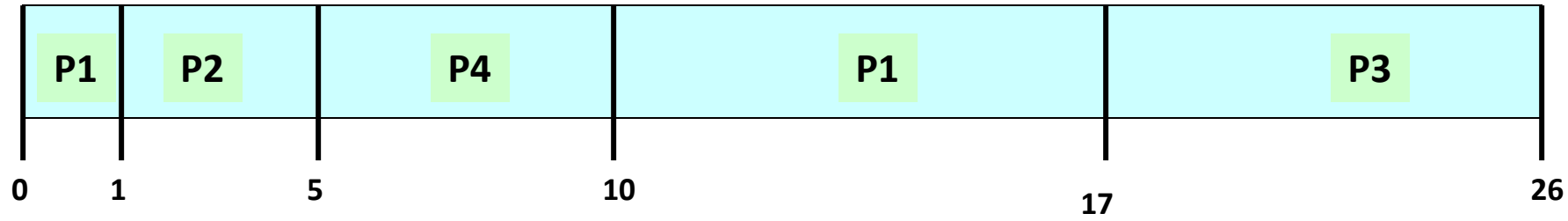
- Average Turnaround Time: (22+8+2+8) / 4 = 10 ms.
- Average Waiting Time: (12+2+0+4)/4 = 4.5 ms.

Preemptive SJF(SRTN-Shortest Remaining Time Next)

EXAMPLE DATA:

Process	Arrival Time	Service Time/ BT
1	0	8
2	1	4
3	2	9
4	3	5

Preemptive Shortest Job First



TAT=(17-0)+(5-1)+(26-2)+(10-3)=52/4=13

Average wait = ((17-8) + (4-4) + (24-9) + (7-5))/4 = 26/4 = 6.5

Example 2: Pre-emptive SJF (Shortest-remaining-time-first)

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

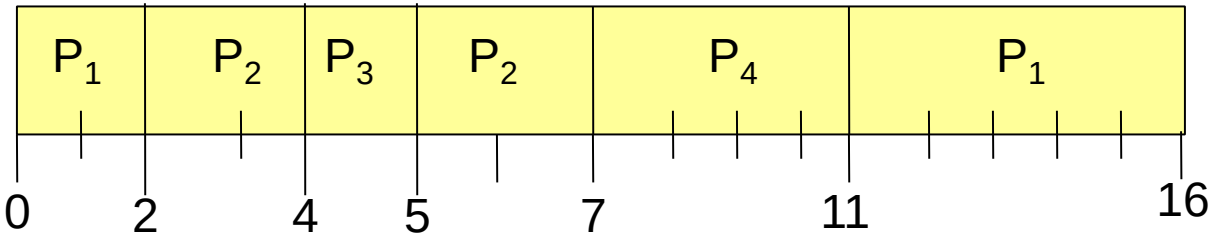
- SJF (preemptive, varied arrival times)

- Average turn-around time =
- Average waiting time=

Example 2: Pre-emptive SJF (Shortest-remaining-time-first)

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (preemptive, varied arrival times)



- Average turn-around time = ((16-0)+(7-2)+(5-4)+ (11-5))/4 = 7
- Average waiting time = ([(16– 7) + (5 - 4)] + [(1 – 1) + (6 –4)/4 = 3

Shortest Remaining Time Next (SRTN)

Marwadi University

- Advantages :
  - **Less waiting time**.
  - **Quite good response** for **short processes**.
- Disadvantages :
  - Again it is **difficult to estimate remaining time** necessary to complete execution.
  - **Starvation is possible** for **long process**. Long process may wait forever.
  - **Context switch overhead is there**.

Round Robin (RR)

Marwadi University

Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.

Performance

- *Time quantum q* is large ⇒ FIFO
- *q* small ⇒ *q* must be large with respect to context switch, otherwise overhead is too high.

Round Robin (RR)

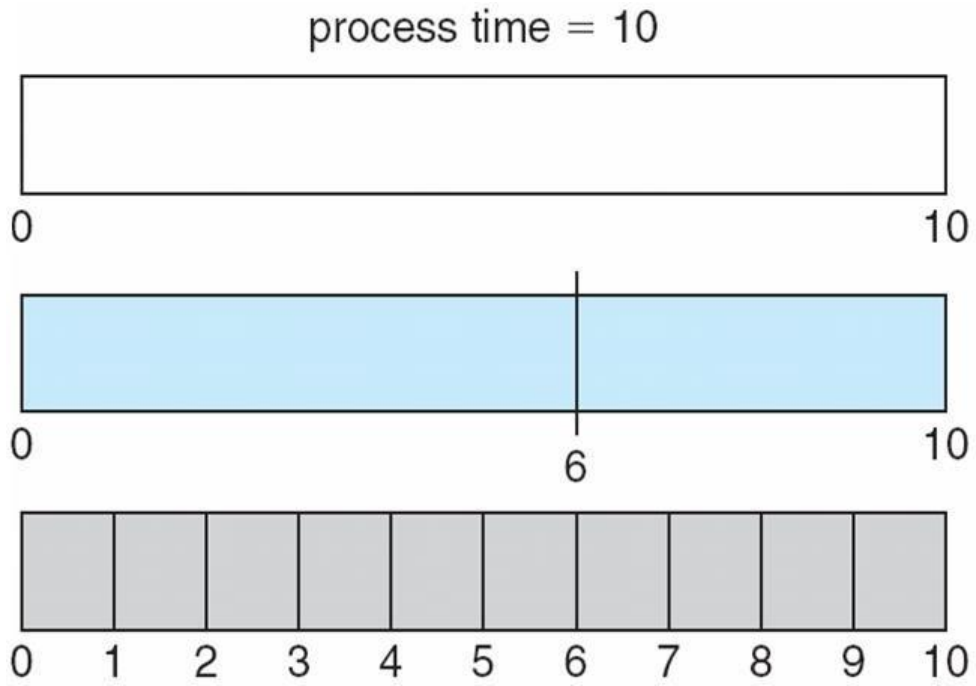
Marwadi University

- Selection Criteria
  - Each selected process is assigned a time interval, called **time quantum or time slice**.
  - Process is **allowed to run only for this time interval**.
  - Here, two things are possible:
    - First, process is either blocked or terminated before the quantum has elapsed. In this case the CPU switching is done and another process is scheduled to run.
    - Second, process needs CPU burst longer than time quantum. In this case, process is running at the end of the time quantum.
    - Now, it will be preempted and moved to the end of the queue.
    - CPU will be allocated to another process.
    - Here, length of time quantum is critical to determine.

Round Robin (RR)

- Decision Mode:
  - **Preemptive**: When quantum time is over or process completes its execution (which ever is earlier), it starts new job.
  - **Selection** of new job is **as per FCFS** scheduling algorithm
- Implementation :
  - This strategy can be implemented by using circular FIFO queue.
  - If any process comes, or process releases CPU, or process is preempted. It is moved to the end of the queue.
  - When CPU becomes free, a process from the first position in a queue is selected to run.

Time Quantum and Context Switch Time

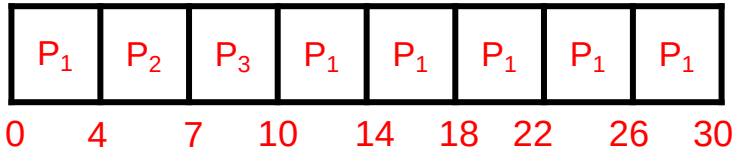


quantum	context switches
12	0
6	1
1	9

Example of RR with TQ = 4

Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



Avg. TAT=15.66

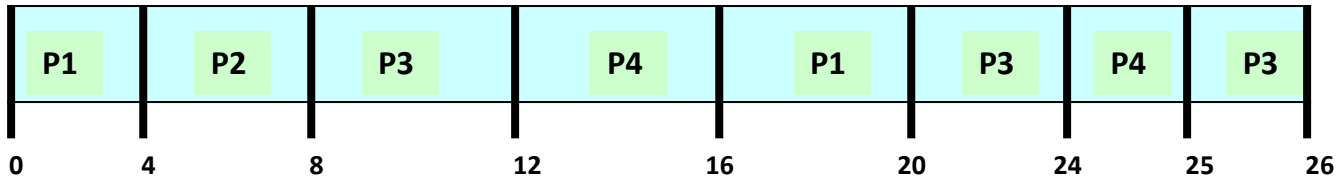
- Av WT=5.66

- Typically, higher average turnaround than SJF, but better *response*

Round Robin

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

Round Robin, quantum = 4, no priority-based preemption



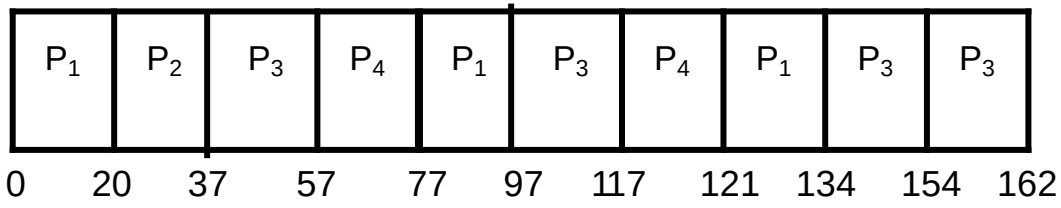
Average Turn around time=((20-0)+(8-1)+(26-2)+(25-3))/4=18.25

Average wait = ((20-8) + (7-4) + (24-9) + (22-5))/4
=(12+3+15+17)/4= 11.75

Example of RR with TQ = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart is:



  Typically, **higher** average turnaround than SJF, but **better** *response time*

- Average turn-around time =(134+37+162+121)/4=113.5
- Average waiting time=((134-53)+(37-17)+(162-68)+(121 -24))/4

  =(81+20+94+97)/4 =73

Round Robin (RR)

Marwadi University

- Advantages:
  - **Simplest, fairest** and **most widely used algorithms**.
- Disadvantages:
  - **Context switch overhead** is there.
  - **Determination of time quantum is too critical**.
    - ✓ If it is too **short**, it **causes frequent context switches** and **lowers CPU efficiency**.
    - ✓ If it is too **long**, it **causes poor response** for **short interactive process**.

Priority Scheduling

Marwadi University

A priority number (integer) is associated with each process

The CPU is allocated to the process with the highest priority (smallest integer = highest priority)

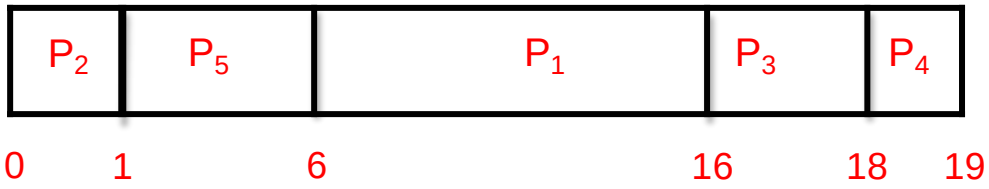
- Preemptive
- nonpreemptive

Problem =**Starvation** – low priority processes may never execute
Solution =**Aging** – as time progresses increase the priority of the process

Example-1(Non-preemptive Priority Scheduling)

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- Priority scheduling Gantt Chart
- Average turn around time=?
- Average waiting time =?



Example-1(Non preemptive Priority Scheduling)

Process	Burst-Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- Priority scheduling Gantt Chart
- Average turn around time=16+1+18+19+6=60/5= 12
- Average waiting time =(16-10)+(1-1)+(18-2)+(19-1)+(6-5)= 8.2

Example-2

- Non preemptive Priority Scheduling

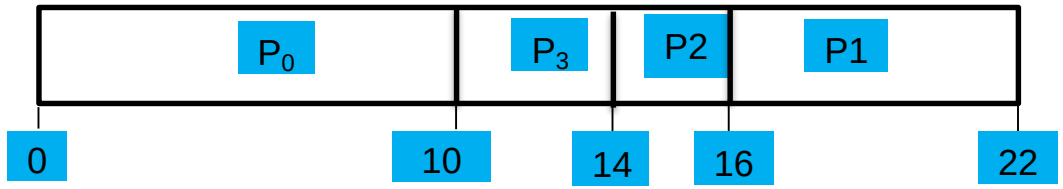
Process	Arrival time(AT)	Burst Time (BT)	Priority
P0	0	10	5
P1	1	6	4
P2	3	2	2
P3	5	4	0

Example-2 : Solution

Process	Arrival time(AT)	Burst Time (BT)	Priority	Completion Time (CT)	Turn Around Time (TAT) TAT=CT-AT	Waiting Time(WT) WT= TAT-BT
P0	0	10	5	10	10	0
P1	1	6	4	22	21	15
P2	3	2	2	16	13	11
P3	5	4	0	14	9	5

Average waiting time: (0+15+11+5)/4= 7.75

Average Turn Around time: (10+21+13+9)/4 =13.25



Non Preemptive Priority Scheduling

- Advantages:
  - Priority is considered so **critical processes can get even better response time**.
- Disadvantages:
  - **Starvation is possible for low priority processes**. It can be overcome by using technique called 'Aging'.
  - **Aging**: **gradually increases the priority of processes** that wait in the system for a long time.

Preemptive Priority Scheduling

- Now we add the concepts of varying arrival times and preemption to the analysis

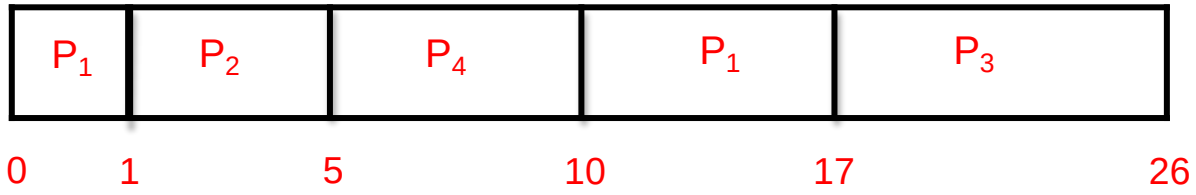
Process	*Arrival* Time	Burst Time	Priority
P_1	0	8	3
P_2	1	4	1
P_3	2	9	5
P_4	3	5	2

Example of Preemptive Priority Scheduling

- Now we add the concepts of varying arrival times and preemption to the analysis

Process	*Arrival* Time	Burst Time	Priority
P_1	0	8	3
P_2	1	4	1
P_3	2	9	5
P_4	3	5	2

- **Preemptive Gantt Chart**



Average Turn Around Time=(17-0)+(5-1)+(26-2)+(10-3)=13

Average waiting time = (17-8)+(4-4)+(24-9)+(7-5)=6.5 msec

Example-2

Marwadi University

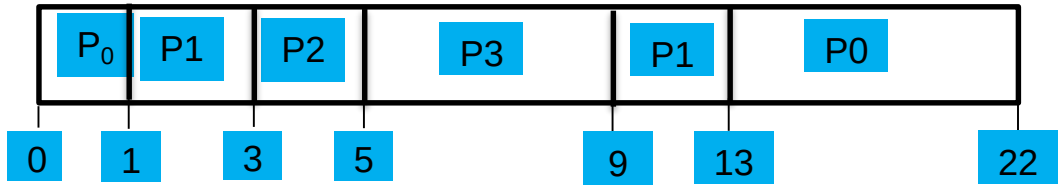
Process	Arrival time(AT)	Burst Time (BT)	Priority
P0	0	10	5
P1	1	6	4
P2	3	2	2
P3	5	4	0

Example-2 Solution

Process	Arrival time(AT)	Burst Time (BT)	Priority	Completion Time (CT)	Turn Around Time (TAT) TAT=CT-AT	Waiting Time(WT) WT= TAT-BT
P0	0	10	5	22	22	12
P1	1	6	4	13	12	6
P2	3	2	2	5	2	0
P3	5	4	0	9	4	0

Average waiting time(12+6+0+0)/4= 4.5 ms

Average Turn Around time: (22+12+2+4)/4 =10 ms



Preemptive Priority Scheduling

Marwadi University

- Selection criteria :
  - The process, that **has highest priority, is served first**.
- Decision Mode:
  - **Preemptive**: When a new process arrives, its priority is compared with current process priority.
  - If the new process has higher priority than the current, the current process is suspended and new job is started.
- Implementation :
  - This strategy can be implemented by using sorted FIFO queue.
  - All processes in a queue are **sorted based on priority with highest priority process at front end**.
  - When CPU becomes free, a process from the first position in a queue is selected to run.

Preemptive Priority Scheduling

- Advantages:
  - Priority is considered so **critical processes can get even better response time**.
- Disadvantages:
  - **Starvation is possible for low priority processes**. It can be overcome by using technique called 'Aging'.
  - **Aging**: **gradually increases the priority of processes** that wait in the system for a long time.
  - **Context switch overhead is there**.

Do this example for all Algorithms

Marwadi University

Process	Arrival Time	Burst Time /Service Time	Priority
A	0	3	5
B	2	6	3
C	4	4	1
D	6	5	2
E	8	2	4

Do this example for all Algorithms

Marwadi University

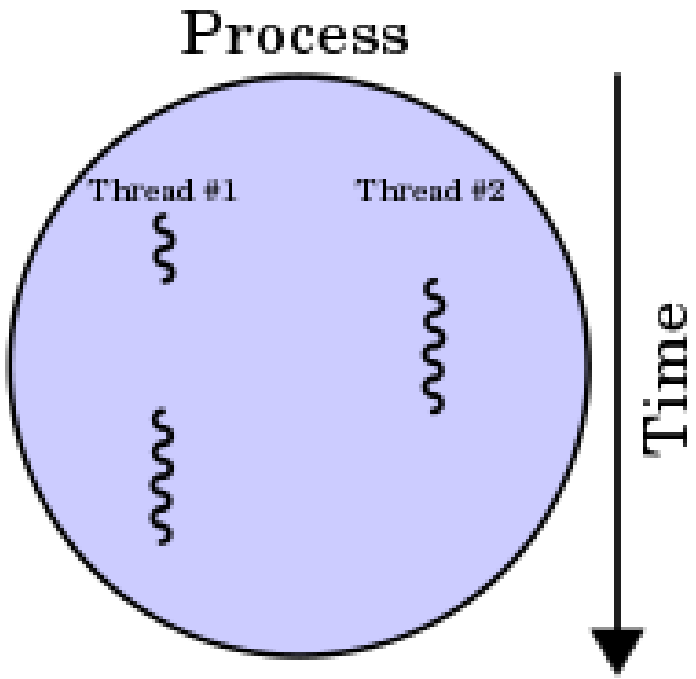
FCFS TAT	FCFS WT	SJF (non – pre-em ptive TAT	SJF WT	SRTF (SJF Preem ptive) TAT	SRTF WT	Priority Non pre-emptive TAT	Priority Non-pre-emptive WT	Priority Pre-em ptive TAT	Priori ty Pre-emptive WT	RR TAT (q=4)	RR WT
3	0	3	0	3	0	3	0	20	17	3	0
7	1	7	1	13	7	7	1	15	9	17	11
9	5	11	7	4	0	9	5	4	0	7	3
12	7	14	9	14	9	10	7	7	2	14	9
12	10	3	1	2	0	10	10	11	9	9	7
8.6	4.6	7.6	3.6	7.2	3.2	7.4	4.6	11.4	7.4	10	6

Threads

Definition :-

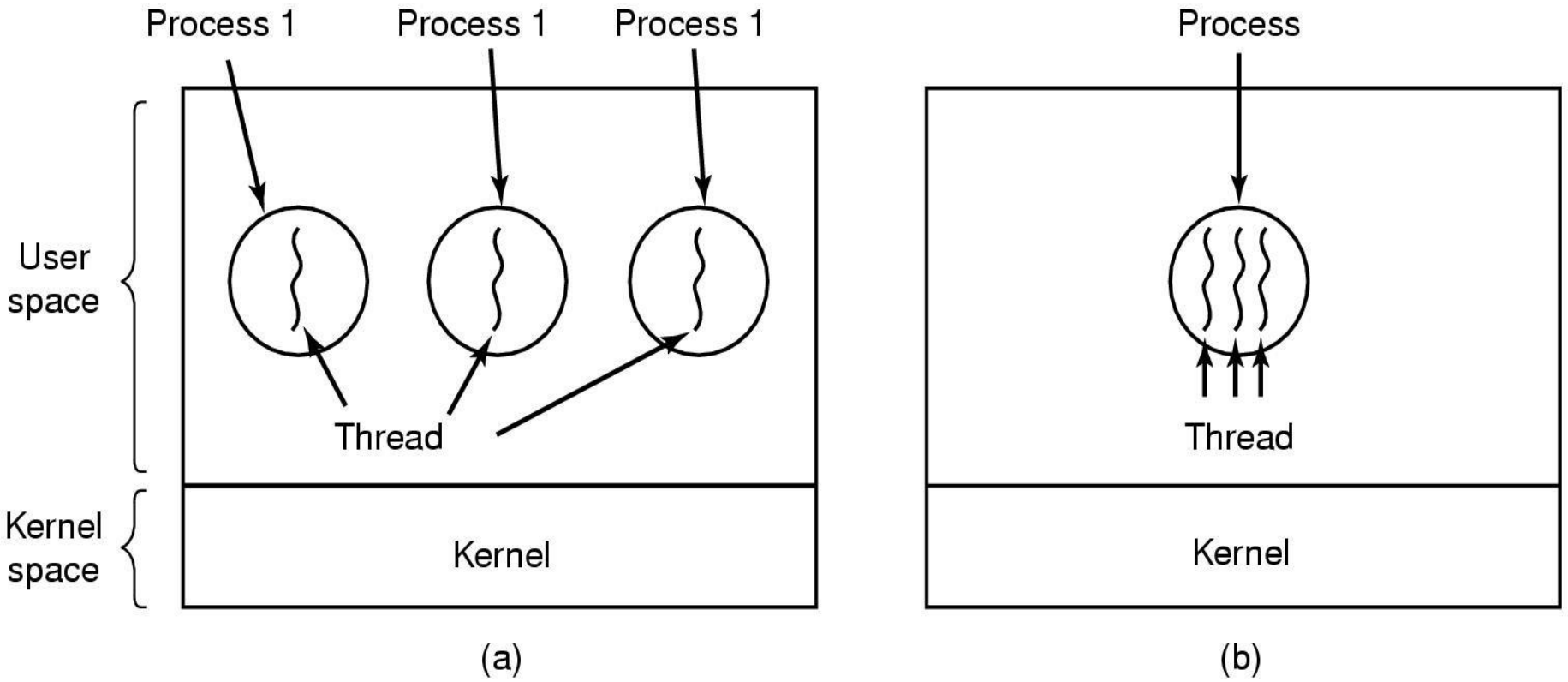
"A thread is the smallest unit of execution."

- Figure : A process with two threads of execution on a single processor machine.
- A thread is a light-weight process.



Threads

- Threads in three process (a) and one process (b)



Threads vs Processes

Marwadi University

Process	Thread
1. System calls involved in process	1. There is no system call involved
2. OS treats different processes differently	2. All user level threads treated as single task for OS
3. Different process have different copies of data, files and code	3. Different threads share same copy of code and data
4. Context Switching is slower	4. Context Switching is faster
5. Blocking of a process will not block to another process	5. Blocking a thread will block entire process
6. Independent	6. Interdependent

Types of Threads

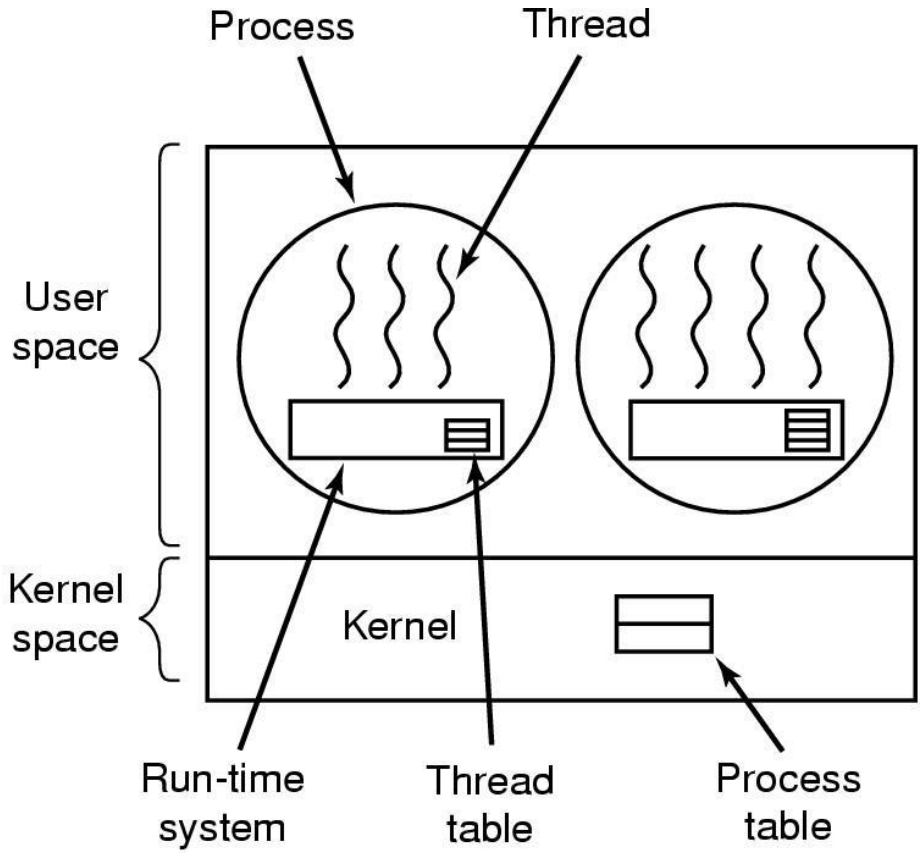
Marwadi University

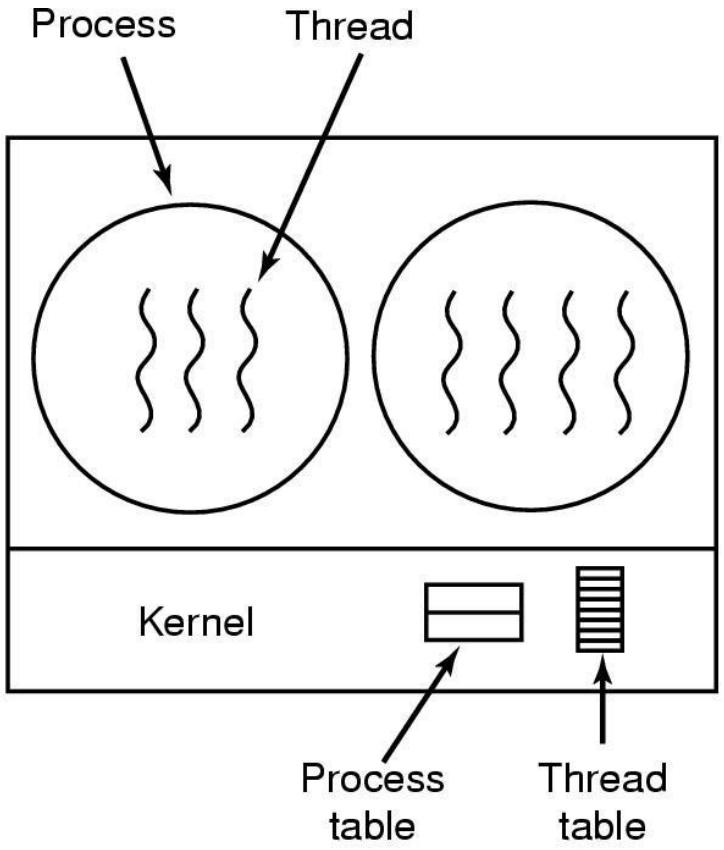
- There are two types of threads to be managed in a modern system: **User threads and kernel threads**.

- **User threads** are supported above the kernel, without kernel support. These are the threads that application programmers would put into their programs.

- **Kernel threads** are supported within the kernel of the OS itself. All modern OSes support kernel level threads, allowing the kernel to perform multiple simultaneous tasks and/or to service multiple kernel system calls simultaneously.

Types of Threads





Threads of User Space

Marwadi University

- **Advantages**
  - Fast switching among threads.
  - Thread scheduling can be application specific.

- **Disadvantages**
  - When a user level thread executes a system call, not only that thread but all of the threads within that process are blocked.
  - Advantages of multiprocessing can not achieve because kernel assigns one process to one processor. (kernel schedules processes not threads).

Threads of Kernel Space

Marwadi University

- **Advantages**

- OS is aware of the presence of threads so if one process thread is blocked it can choose next process to run.
- OS can schedule multiple threads from the process on multiple processor

- **Disadvantages**

- Switching between threads is time consuming because kernel must do the switching.

Multiprocessing vs Multithreading

- **Multiprocessing** is the use of two or more central processing units (CPUs) within a single computer system.

- **Multi-threading** is a widespread programming and execution model that allows multiple threads to exist within the context of a single process.


THANK
YOU