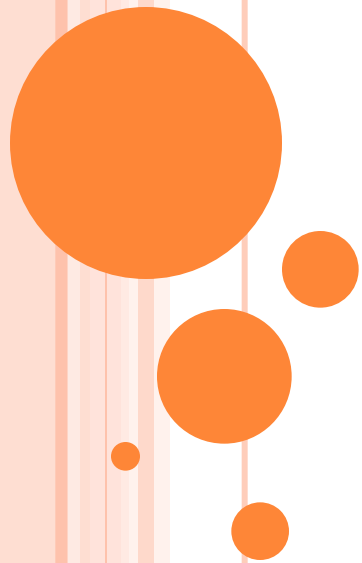


# OPERATING SYSTEM



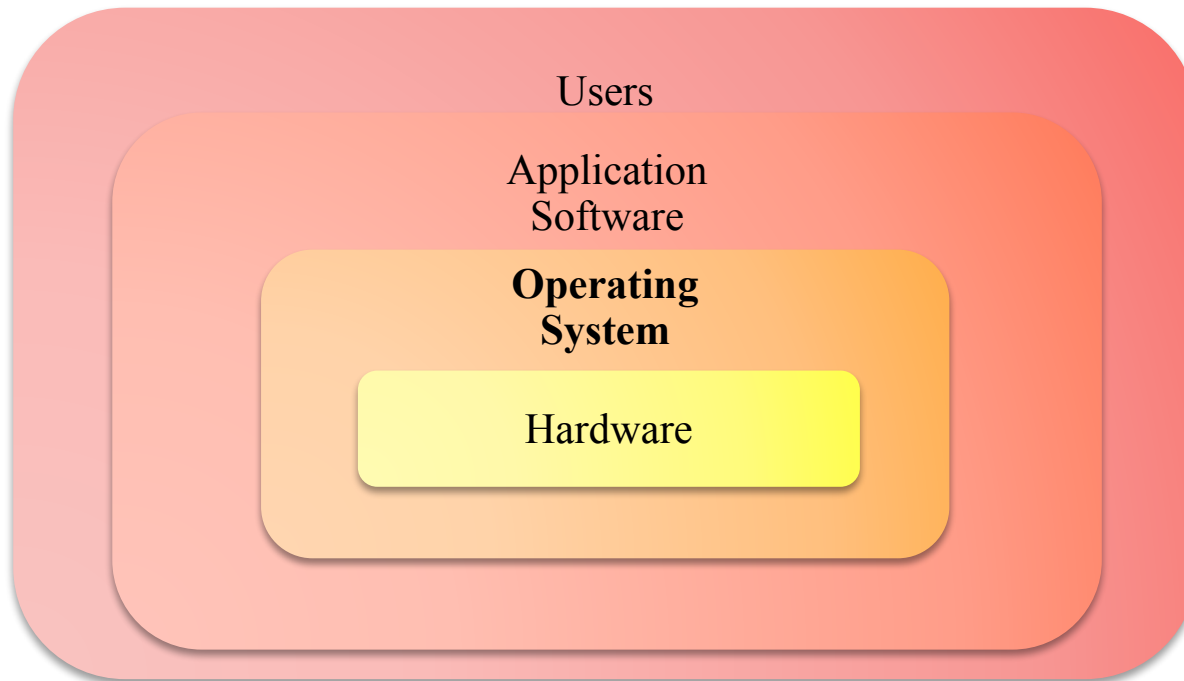
# COMPONENTS OF COMPUTER SYSTEM?

- Computer system can be divided into four components:
  - **Hardware** – provides basic computing resources
    - CPU, memory, I/O devices
  - **Operating system**
    - Controls and coordinates use of hardware among various applications and users
  - **Application programs** – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - **Users**
    - People, machines, other computers



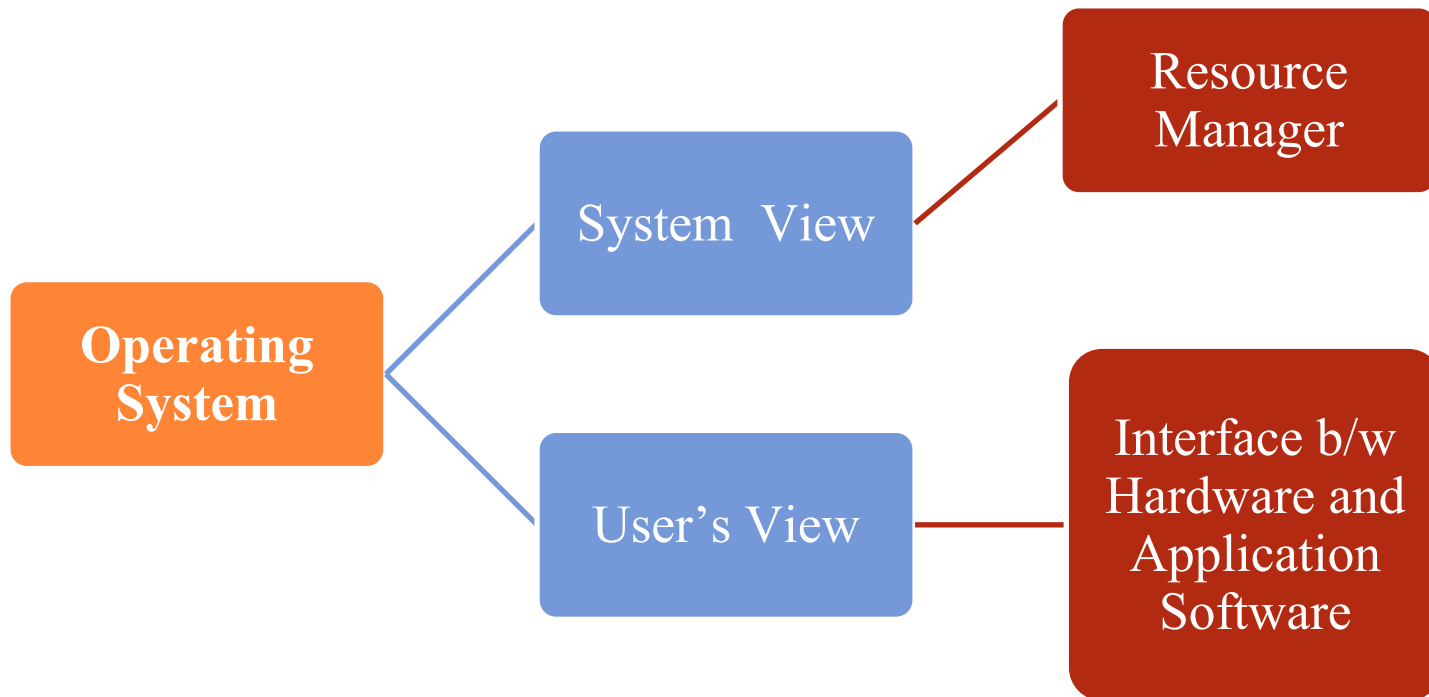
# COMPONENTS OF COMPUTER SYSTEM?

- Computer system can be divided into four components:



# WHAT IS OPERATING SYSTEM?

- We can define an Operating System from two point of view.



# WHAT IS OPERATING SYSTEM?

## *System View of O.S.*

- **Resource Manager:** It manages all the resources of the computer.eg. Memory Management, File Management, Process Management, I/O device management, CPU Scheduling Management etc.

Every Resource of Computer is managed by Operating System.



# WHAT IS OPERATING SYSTEM?

## *User's View of O.S.*

- It acts as an Interface between the Hardware and Application Software (or User).

So, Operating System provides a platform using which user (or Application Software) can make use of services of hardware.



# How OPERATING SYSTEM IS LOADED?

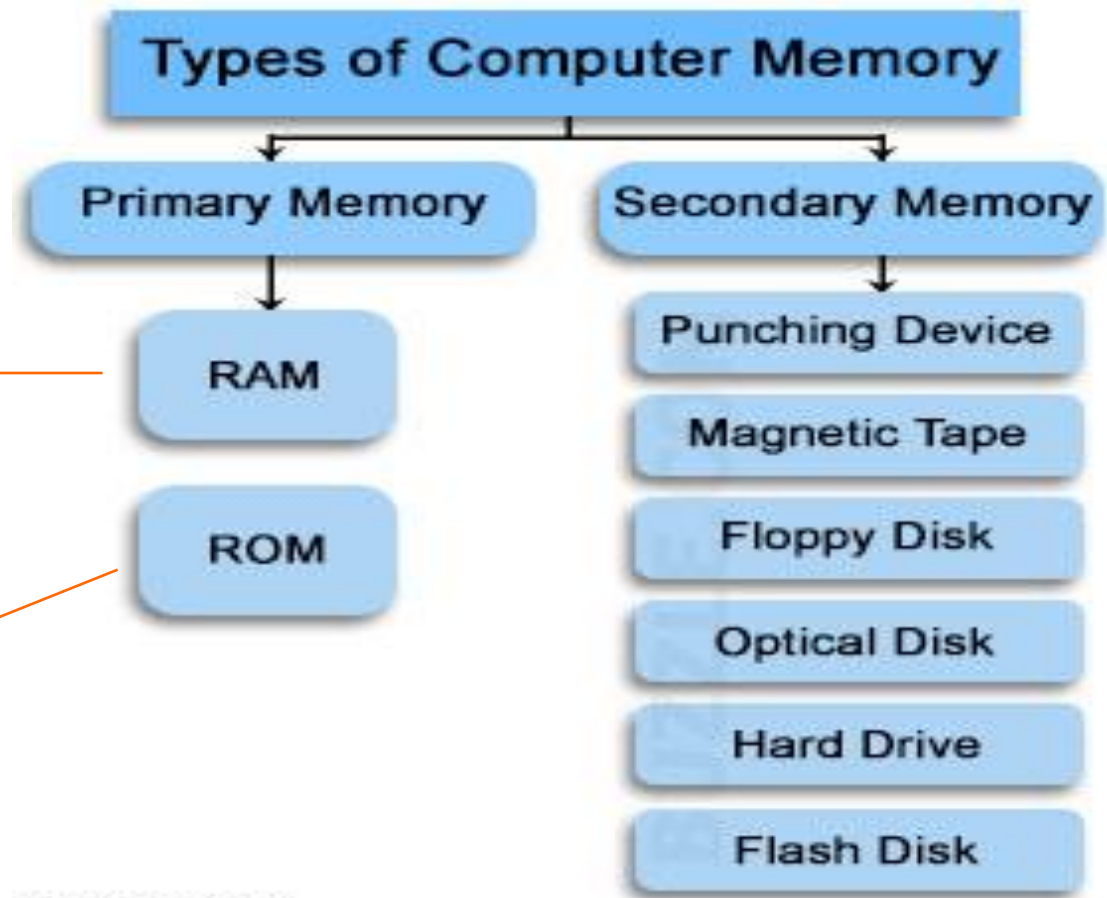
- When power is turned on, POST (Power-On Self-Test) runs to determine if the computer keyboard, random access memory, disk drives, and other hardware are working correctly.
- Now, the BIOS (Basic Input Output system) software in ROM loads the O.S. in to the RAM. This is known as **Booting**.  
RAM=Random Access Memory  
ROM=Read Only Memory
- O.S. now takes charge of the computer.



# MEMORY TYPES...

Volatile  
Memory  
(Read and  
Write)

Non-Volatile  
Memory  
(Read  
Only)





# OBJECTIVE OF OPERATING SYSTEM?

- ❑ Users want convenience, **ease of use** and **good performance** , **BUT** Don't care about **resource utilization**
- ❑ **Main Objective of Operating System is**

*“Efficient (Maximum) Utilization of Resources”*



# FUNCTIONS/SERVICES OF OPERATING SYSTEM?

- ❑ **Control Program:** Controls Execution of the Programs
  - ❑ **CPU Scheduling**
- ❑ **Resource Allocation and Management**
  - ❑ **Memory Management**
  - ❑ **I/O Device Management**
  - ❑ **Process Management**
- ❑ **Security and Protection**
- ❑ **Efficient Utilization Of Resources**
- ❑ **User Convenience**
- ❑ **Process Synchronization**



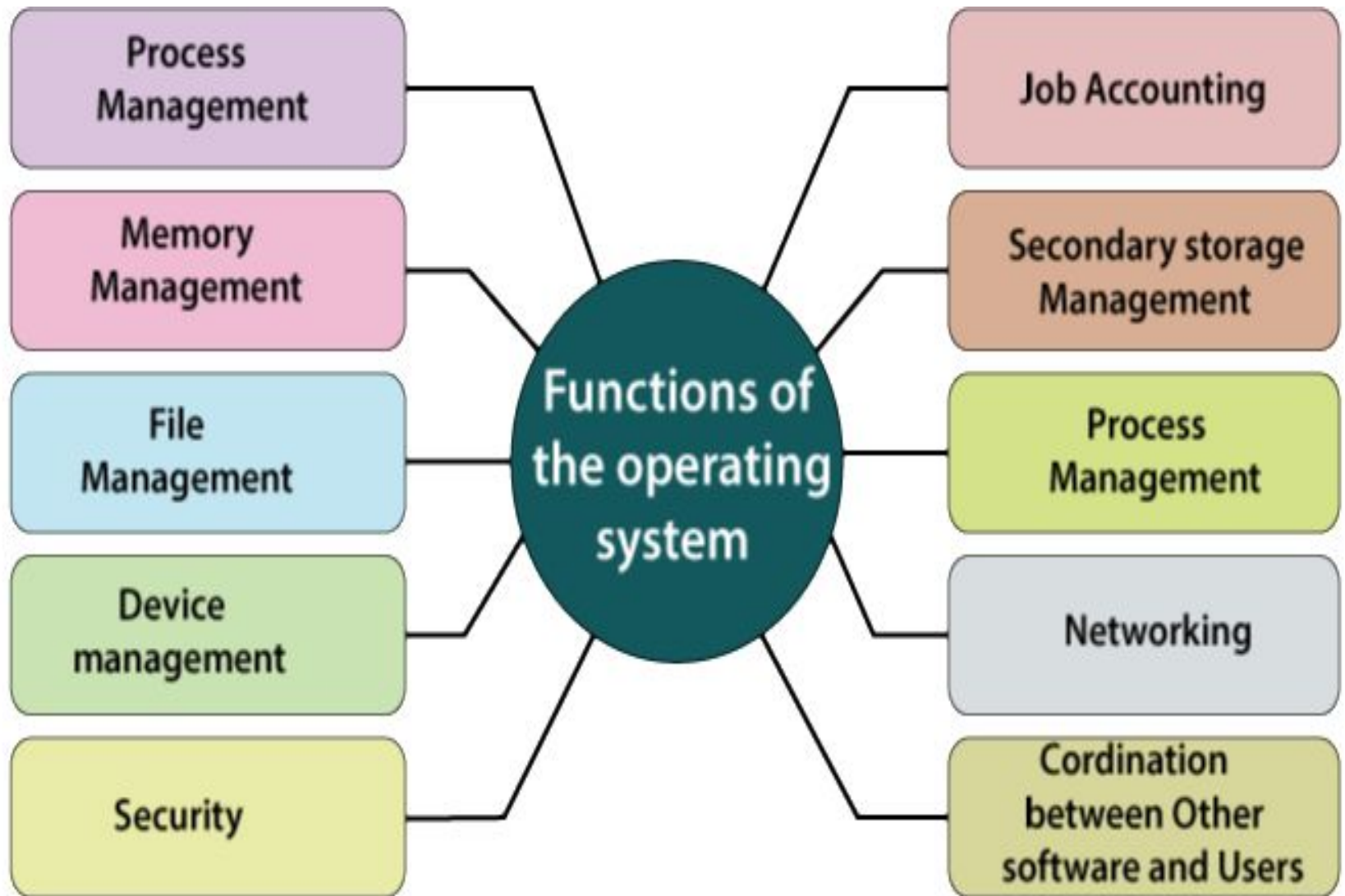
**Allocates  
Resources**

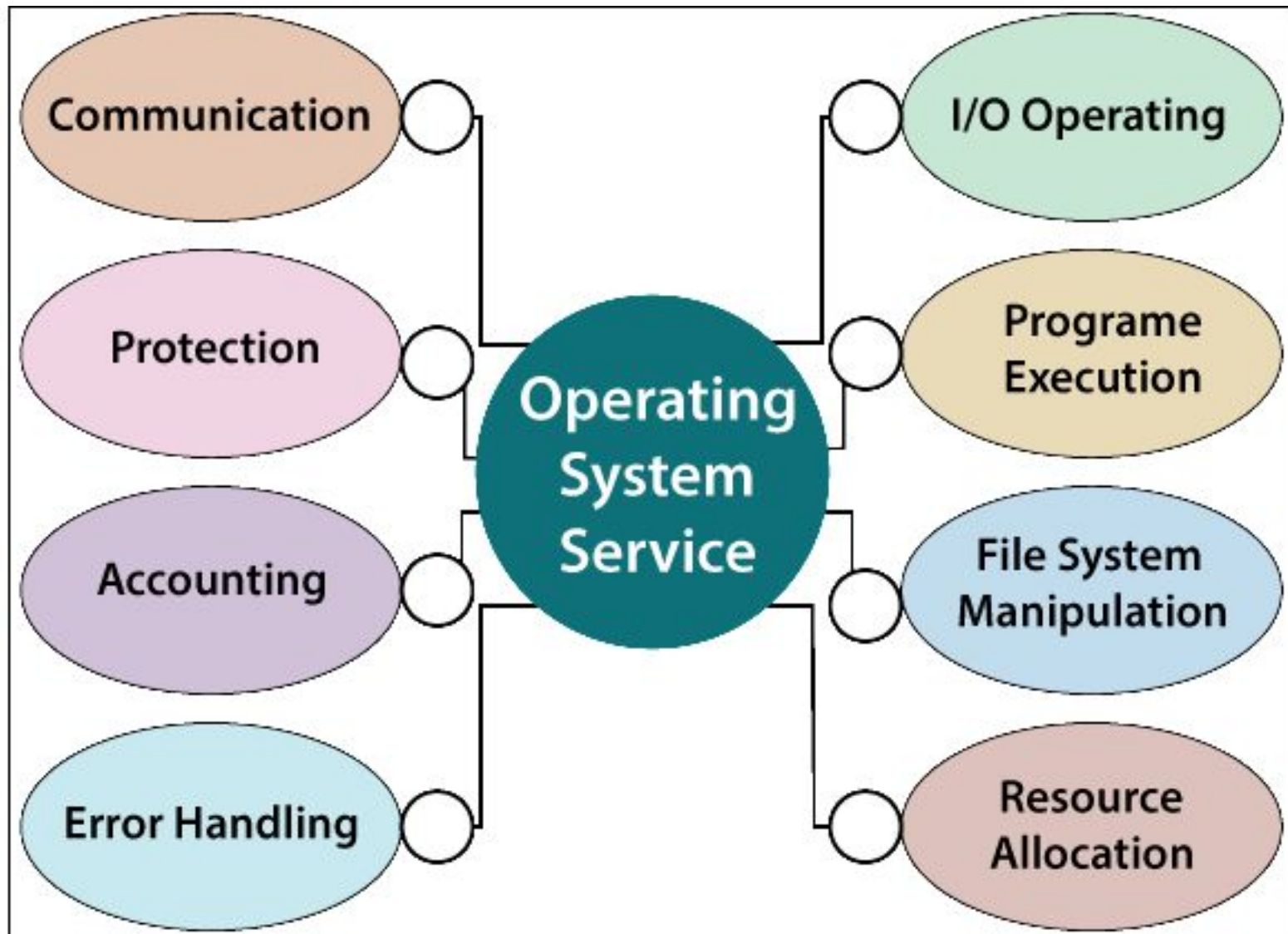
**Operating  
System**

**Monitors  
Activities**

**Manages  
Disks & Files**







# What are the different types?

**Mac OS** is a series of graphical user interface-based operating systems developed by Apple Inc. for their Macintosh



**Linux** is a Unix-like computer operating system assembled under the model of free and open source software development and distribution.



**Microsoft Windows** is a series of graphical interface operating systems developed, marketed, and sold by Microsoft.



**iOS** (previously **iPhone OS**) is a mobile operating system developed and distributed by Apple Inc. Originally unveiled in 2007 for the iPhone, it has been extended to support other Apple devices such as the iPod Touch

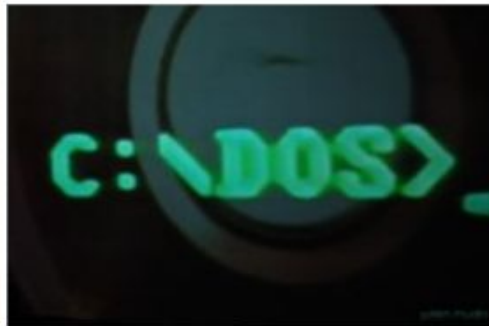


**Android** is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers. Initially developed by Android, Inc.



**BSD/OS** had a reputation for reliability in server roles; the renowned Unix programmer and author W. Richard Stevens used it for his own personal web server for this reason.

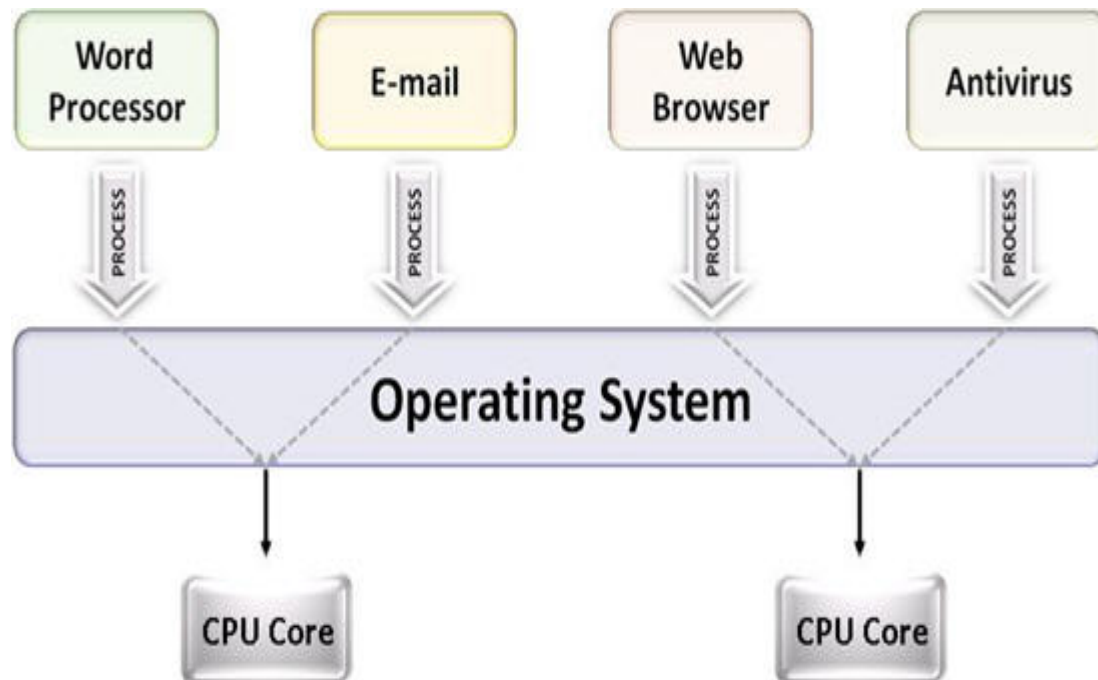




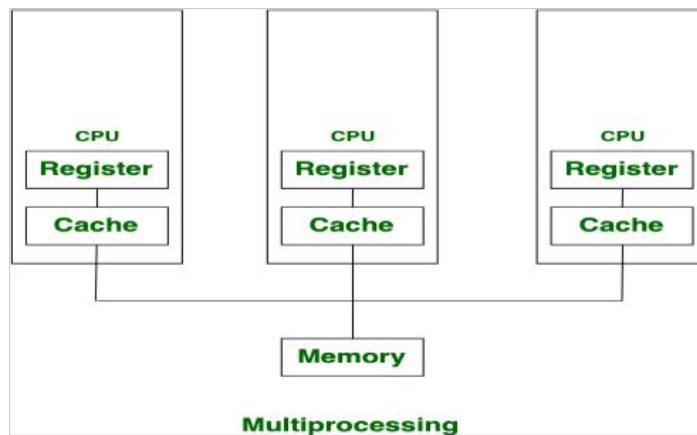
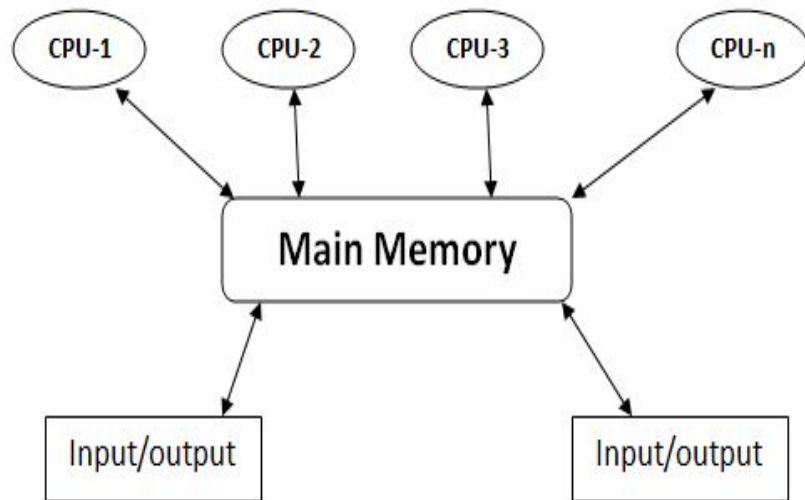
# Basic concepts of O.S.

## MULTIPROCESSING

Multiprocessing is the ability of an operating system to execute more than one process simultaneously on a multi processor machine. In this, a computer uses more than one CPU at a time.

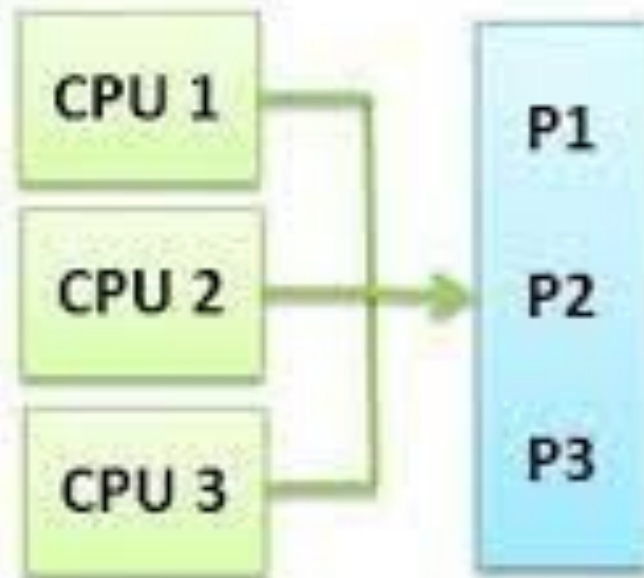






Sr. No.	Multiprocessing
1	Multiprocessing refers to processing of multiple processes at same time by multiple CPUs.
2	It utilizes multiple CPUs.
3	It permits parallel processing.
4	Less time taken to process the jobs.
5	It facilitates much efficient utilization of devices of the computer system.
6	Usually more expensive.

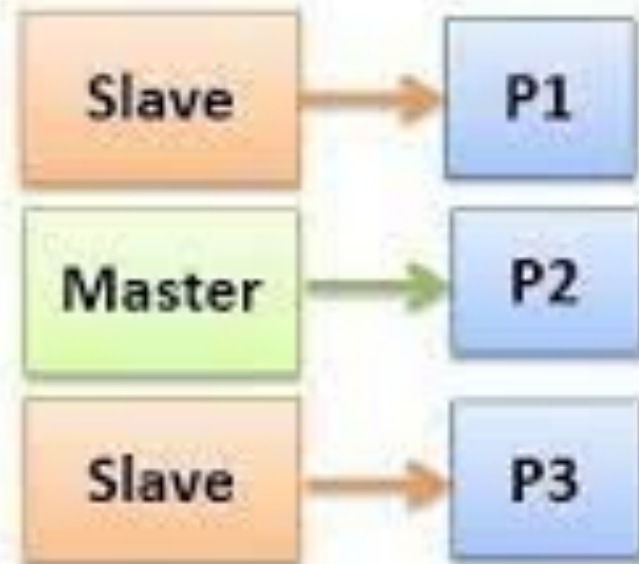
## Symmetric Multiprocessing



(Shared Memory)

**Vs**

## Asymmetric Multiprocessing



(No Shared Memory)



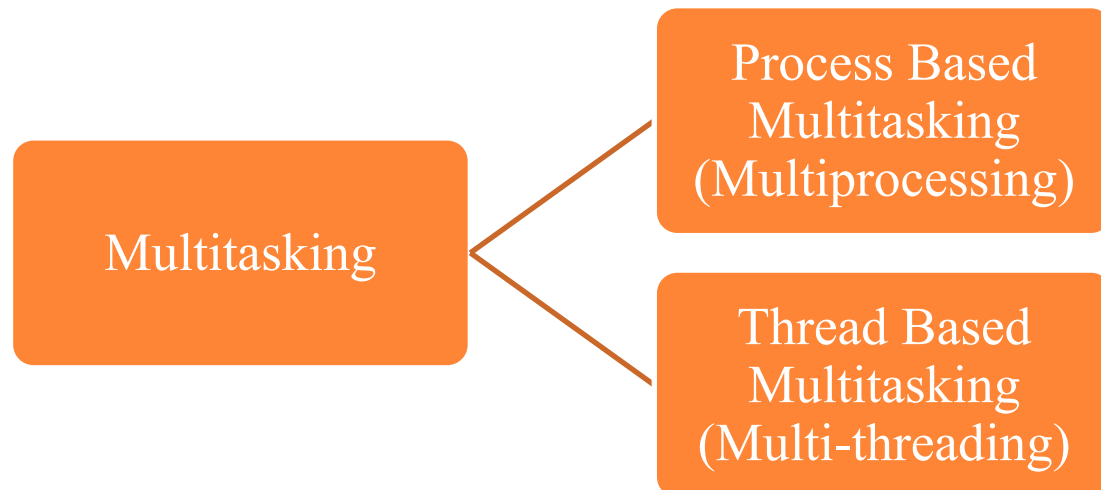
# CONCEPTS OF O.S.

## ▣ **Multiprogramming OS**

It is the ability of an operating system to execute more than one program on a single processor machine.

## ▣ **Multitasking OS (Time Sharing OS)**

It is the ability of an operating system to execute more than one task simultaneously on a single processor machine.




- Multitasking is a process of executing multiple tasks simultaneously. We use multitasking to utilize the CPU. Multitasking can be achieved by 2 ways:
  - Process-based Multitasking (Multiprocessing)
  - Thread-based Multitasking (Multithreading)
- Multiprocessing is the use of two or more central processing units (CPUs) within a single computer system, means execution of multiple concurrent software processes at any one instant
- Multithreading is the ability to run several parts of a program in parallel, so you can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel

The OS divides processing time not only among different applications, but also among each thread within an application



# MULTIPROGRAMMING VERSUS MULTITASKING

Multiprogramming	Multitasking
In multiprogramming, multiple processes run concurrently at the same time on a single processor.	Multitasking is when more than one task is executed at a single time utilizing multiple CPUs
It is based on the concept of context switching.	It is based on the concept of time sharing.
Multiple programs reside in the main memory simultaneously to improve CPU utilization so that CPU doesn't sit idle for a long time.	It enables execution of multiple tasks and processes at the same time to increase CPU performance.
It utilizes single CPU for execution of processes.	It utilizes multiple CPUs for task allocation.
It takes more time to execute the processes.	It takes less time to execute the tasks or processes.
The idea is to reduce the CPU idle time for as long as possible.	The idea is to allow multiple processes to run simultaneously via time sharing.
	 <b>Difference Between.net</b>

## MULTIPROGRAMMING

It is the ability of an operating system to execute more than one program on a single processor machine. More than one task/program/job/process can reside into the main memory at one point of time. A computer running excel and fire-fox browser simultaneously is an example of multiprogramming.

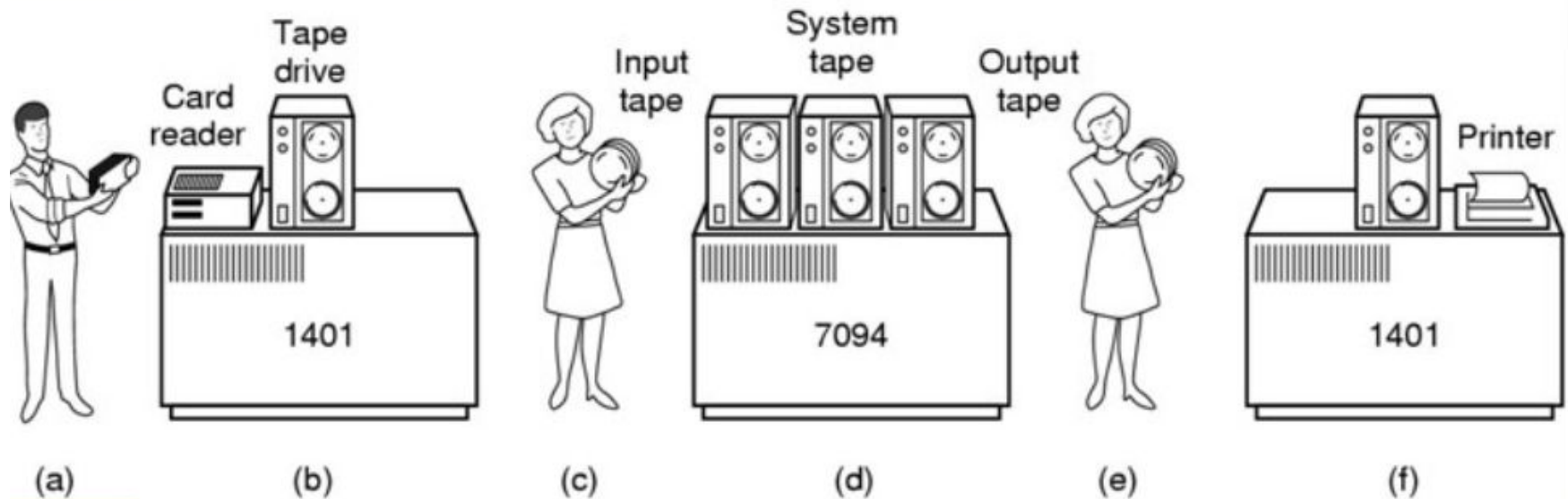




## Different types of operating systems

- Batch Operating System
- Multi programmed
- Multitasking





## Early Batch System

bring cards to 1401  
read cards to tape  
put tape on 7094 which does  
computing  
put tape on 1401 which prints output



# Batch Operating System

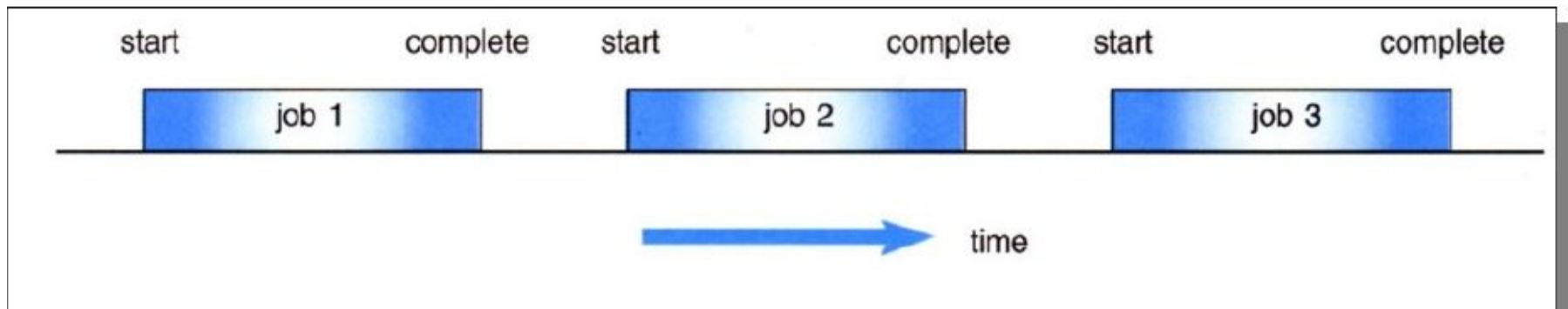
- In Batch processing same type of jobs batch (BATCH- a set of jobs with similar needs) together and execute at a time.
- The OS was simple, its major task was to transfer control from one job to the next.
- The job was submitted to the computer operator in form of punch cards. At some later time the output appeared.
- The OS was always resident in memory. (Ref. Fig. next slide)
- Common Input devices were card readers and tape drives.

- Common output devices were line printers, tape drives, and card punches.
- Users did not interact directly with the computer systems, but he prepared a job (comprising of the program, the data, & some control information).



## BATCH O.S.

- No job can be started until previous job is completed



The Main disadvantage of the Batch Operating System was the

“ Poor CPU Utilization”.

**To overcome this limitation of Batch O.S., Multiprogramming O.S. were designed to improve the CPU Utilization so that CPU do not sit idle.**



The Main disadvantage of the Batch Operating System was the

“ Poor CPU Utilization”.

**To overcome this limitation of Batch O.S., Multiprogramming O.S. were designed to improve the CPU Utilization so that CPU do not sit idle.**

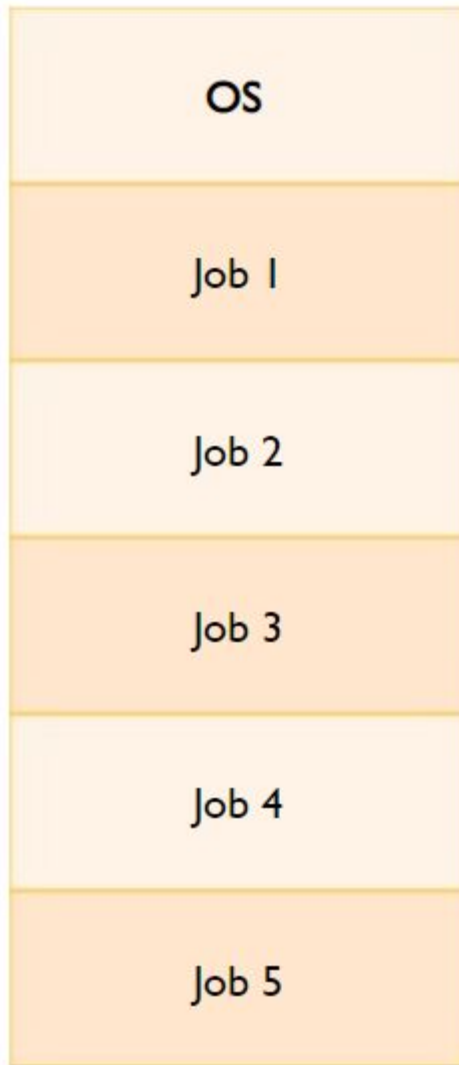


# Multi programmed System

- Multiprogramming is a technique to execute number of programs simultaneously by a single processor.
- In Multiprogramming, number of processes reside in main memory at a time.
- The OS picks and begins to executes one of the jobs in the main memory.
- If any I/O wait happened in a process, then CPU switches from that job to another job.
- Hence CPU in not idle at any time.







- Figure depicts the layout of multiprogramming system.

- The main memory consists of 5 jobs at a time, the CPU executes one by one.

### Advantages:

- Efficient memory utilization
- Throughput increases
- CPU is never idle, so performance increases.



# Multitasking/Time Sharing System

- Time sharing, or multitasking, is a logical extension of multiprogramming.
- Multiple jobs are executed by switching the CPU between them.
- In this, the CPU time is shared by different processes, so it is called as “Time sharing Systems”.
- Time slice is defined by the OS, for sharing CPU time between processes.
- Examples: Multics, Unix, etc.,



# Time Quantum=3

P1,  
ET=6

P2,  
ET=9

P3,  
ET=11

Terminal  
A

Terminal  
B

Terminal  
C

Central processor

Multiuser operating system share Processor time



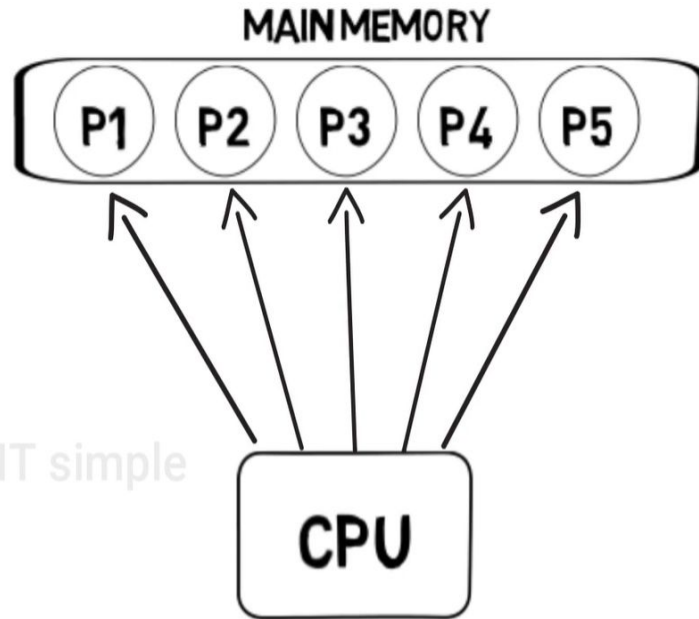
Time Slices

3 3 3 3 3 3 3 3 2





making IT simple



TIME QUANTUM = 6 NANO SECONDS

= PERIOD OF TIME FOR WHICH A  
PROCESS IS ALLOWED TO RUN



Each Process is executed for 6 nanoseconds.

ET(P1): 12   ET(P2):8   ET(P3): 7   ET(P4):6   ET(P5):9

**Round1**::P1=6, P2=6, P3=6, P4=6, P5=6.....

**Round2**: P1=6, P2=2, P3=1, P5=3.....



<b>Multiprogramming</b>	<b>Multitasking</b>
In multiprogramming, multiple processes run concurrently at the same time on a single processor	Multitasking is when more than one task is executed at a single time by utilizing multiple CPUs
It is based on the concept of context switching	It is based on the concept of time sharing
Multiple programs reside in the main memory simultaneously to improve CPU utilization so that CPU does not sit idle for a long time	It enables execution of multiple tasks and processes at the same time to increase CPU performance
It utilizes single CPU for execution of processes	It utilizes multiple CPUs for task allocation
It takes more time to execute the processes	It takes less time



# Operating System Structure

---

## Multiprogramming environment

- Multiprogramming idea is as follows:
  1. The operating system keeps **several jobs** in memory simultaneously .
  2. One job **selected** and run via job scheduling.
  3. When it has to **wait** (for I/O for example), OS switches to another job
  4. Eventually, the first job finishes waiting and gets the CPU back.
  5. As long as at least one job needs to execute, the CPU is never idle.

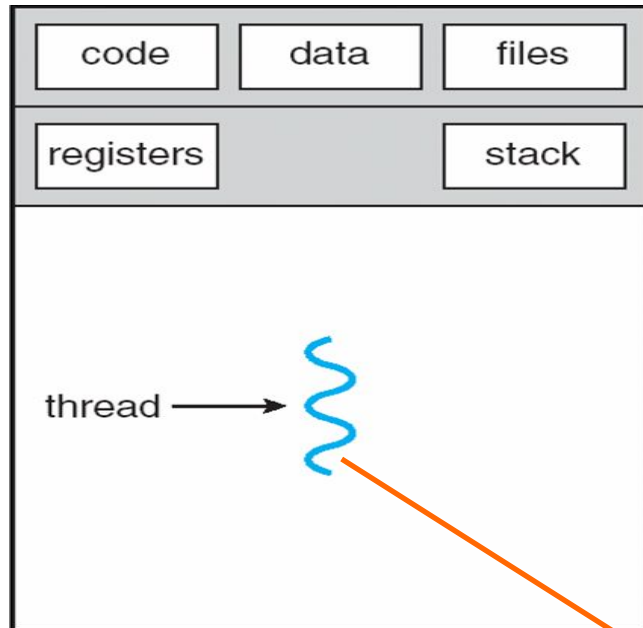
## Time sharing (or multitasking) system:

- Multiple jobs are executed by switching the CPU between them. **frequently that the users can interact with each program while it is running.**
- In this, the CPU time is shared by different processes, so it is called as “Time sharing Systems”.
- Time slice is defined by the OS, for sharing CPU time between processes.
- CPU is taken away from a running process when the allotted time slice expires.

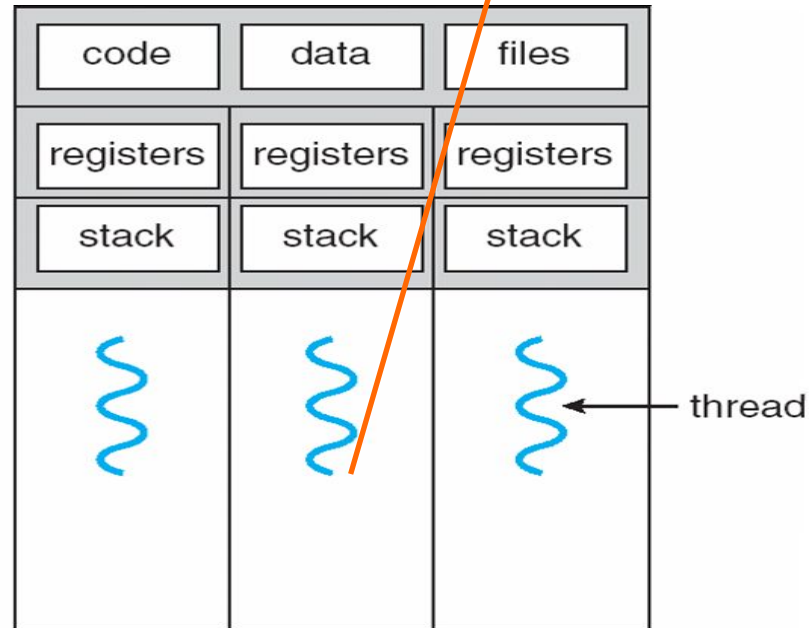
ex: Unix, etc.



# MULTI-THREADING...



single-threaded process



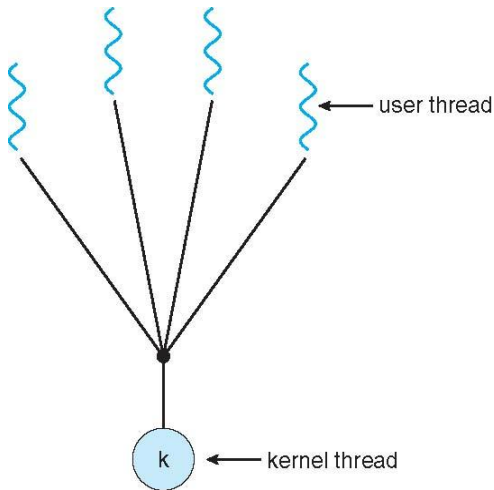
multithreaded process

Light Weight  
Process (LWP)

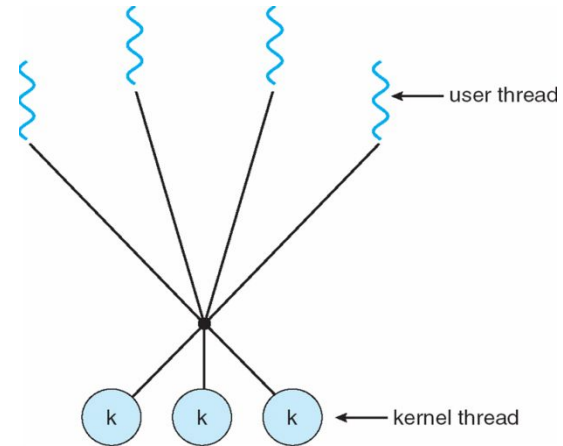
Heavy Weight  
Process (HWP)



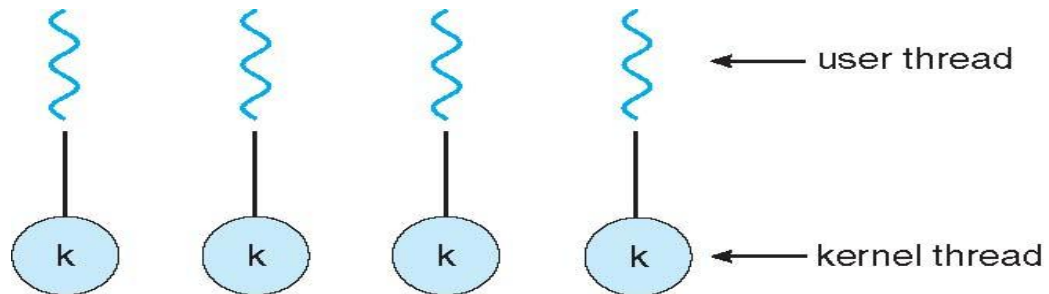
# MULTITHREADING MODELS



Many-to-One



Many-to-Many



One-to-One



# TYPES OF O.S. BASED ON No. OF USERS:

Operating System can also be classified as,-

- ▣ **Single User Systems**
- ▣ **Multi User Systems**



# SINGLE USER SYSTEMS:

- Provides a platform for only one user at a time.
- They are popularly associated with Desk Top operating system which run on standalone systems where no user accounts are required.
- Example: DOS



# MULTI-USER SYSTEMS:

- Provides regulated access for a number of users by maintaining a database of known users.
- Refers to computer systems that support two or more simultaneous users.
- Another term for *multi-user* is *time sharing*.
- Ex: All mainframes and are multi-user systems.
- Example: Unix

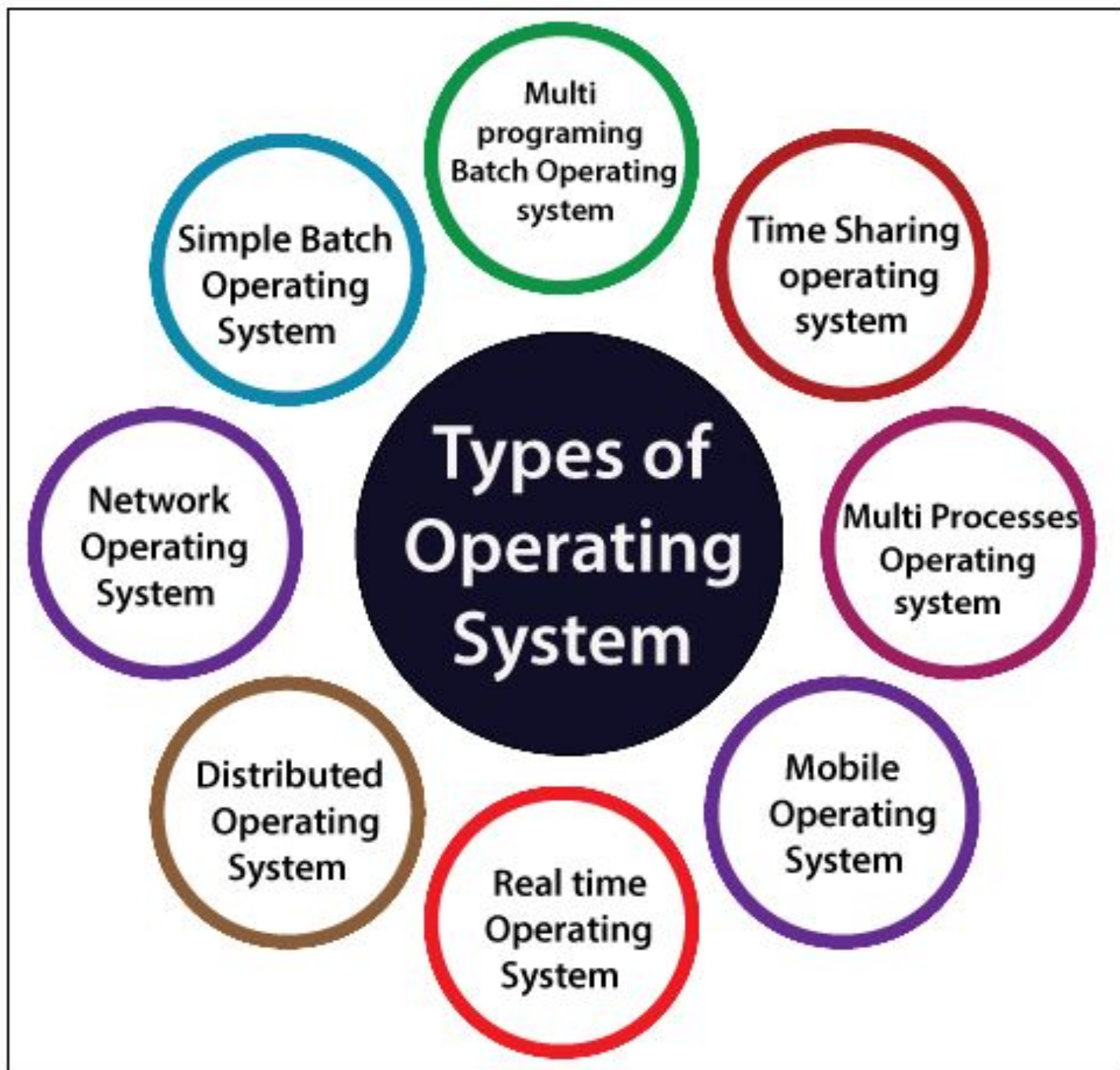




## REAL TIME O.S.

- It is used when response time of the job is bounded i.e. user must get the response within bounded time.
  
- It is of two types:
  - Hard Real Time O.S. —used in Air Traffic Control Systems, Military Applications, Traffic Control systems
  - Soft Real Time O.S.—used in ATM etc.





# CPU SCHEDULING

**Before we start CPU Scheduling, we must know the concept of Process and Programs.**

- Program is *passive* entity stored on disk (**executable file**), process is *active*
  - Program becomes process when executable file loaded into memory

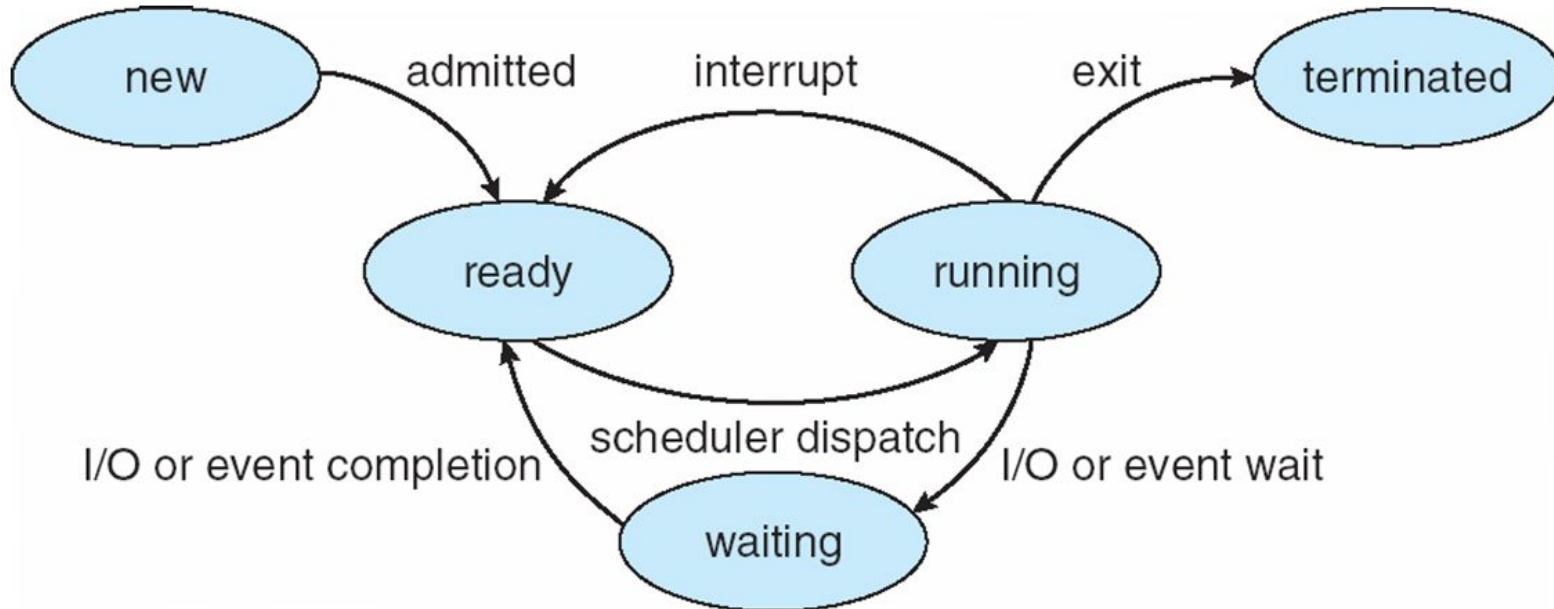


# PROCESS.....

- As a process executes, it changes **state**
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a processor
  - **terminated**: The process has finished execution



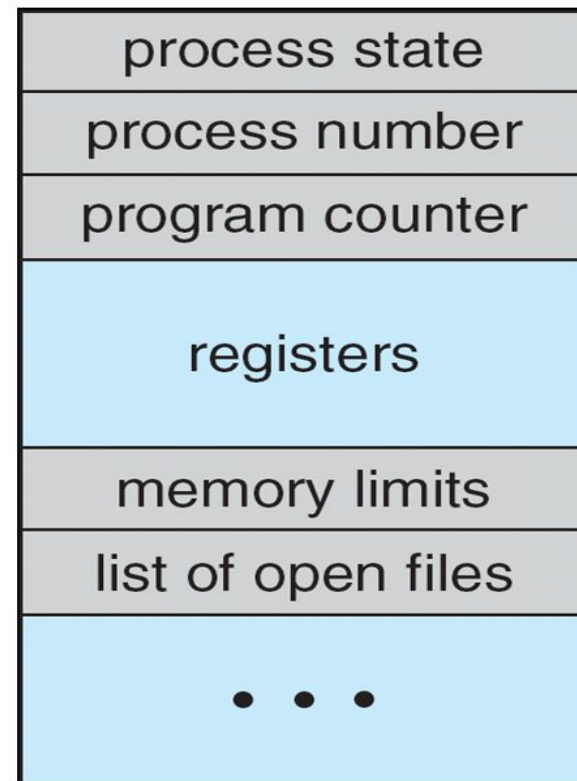
# PROCESS STATES.....



# PCB (PROCESS CONTROL BLOCK)

Information associated with each process is stored in its PCB.

- ❑ 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



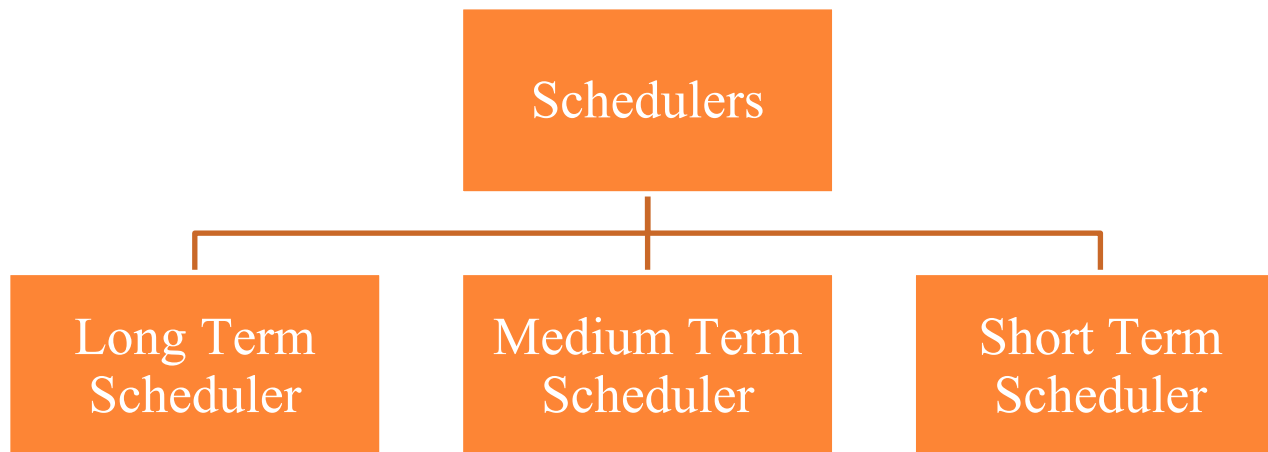
# PROCESS SCHEDULING

- Maximize CPU use, quickly switch processes onto CPU for time sharing
- **Process scheduler** selects among available processes for next execution on CPU
- Maintains **scheduling queues** of processes
  - **Job queue** – set of all processes in the system
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - **Device queues** – set of processes waiting for an I/O device
  - Processes migrate among the various queues





# TYPES OF SCHEDULERS



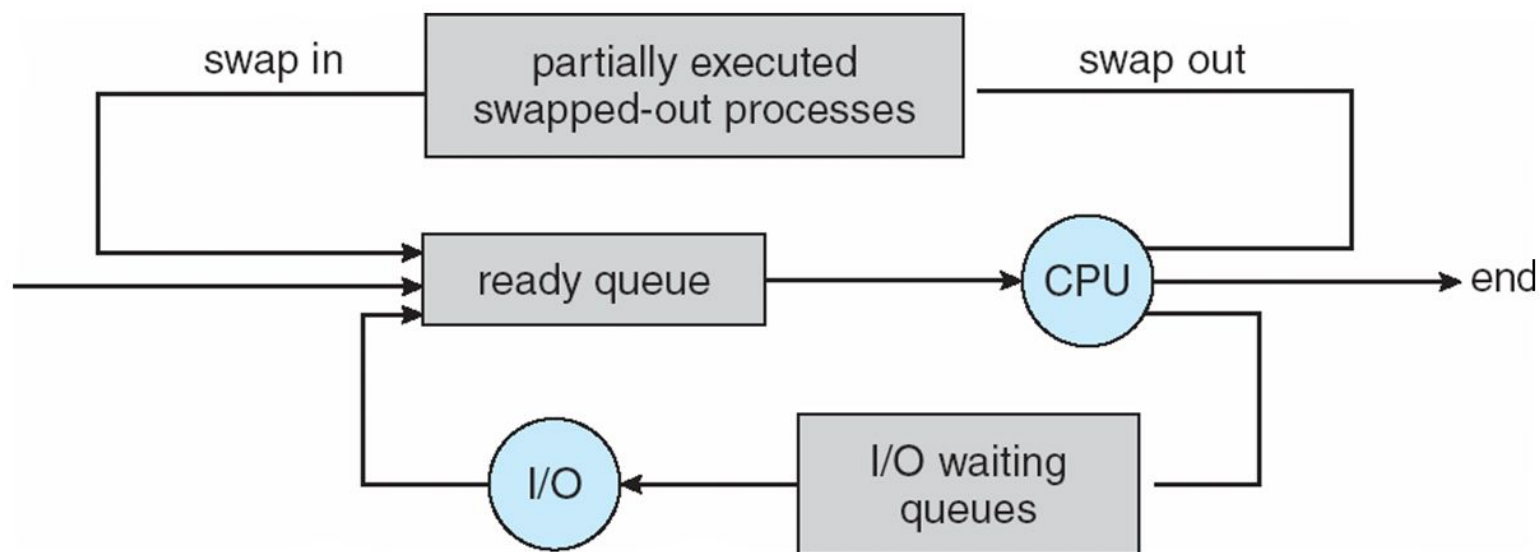
## LONG TERM SCHEDULERS

- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
  - Long-term scheduler is invoked infrequently (seconds, minutes)  $\Rightarrow$  (may be slow)
  - The long-term scheduler controls the **degree of multiprogramming**
- Processes can be described as either:
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts
- Long-term scheduler strives for good *process mix*



# MEDIUM TERM SCHEDULERS

- **Medium-term scheduler** can be added if degree of multiple programming needs to decrease
  - Remove process from memory, store on disk, bring back in from disk to continue execution: **swapping**



## SHORT TERM SCHEDULERS

- ▣ **Short-term scheduler** (or **CPU scheduler**) – selects which process should be executed next and allocates CPU
  - Sometimes the only scheduler in a system
  - Short-term scheduler is invoked frequently (milliseconds)  $\Rightarrow$  (must be fast)

It is called “Short term” because it works frequently.



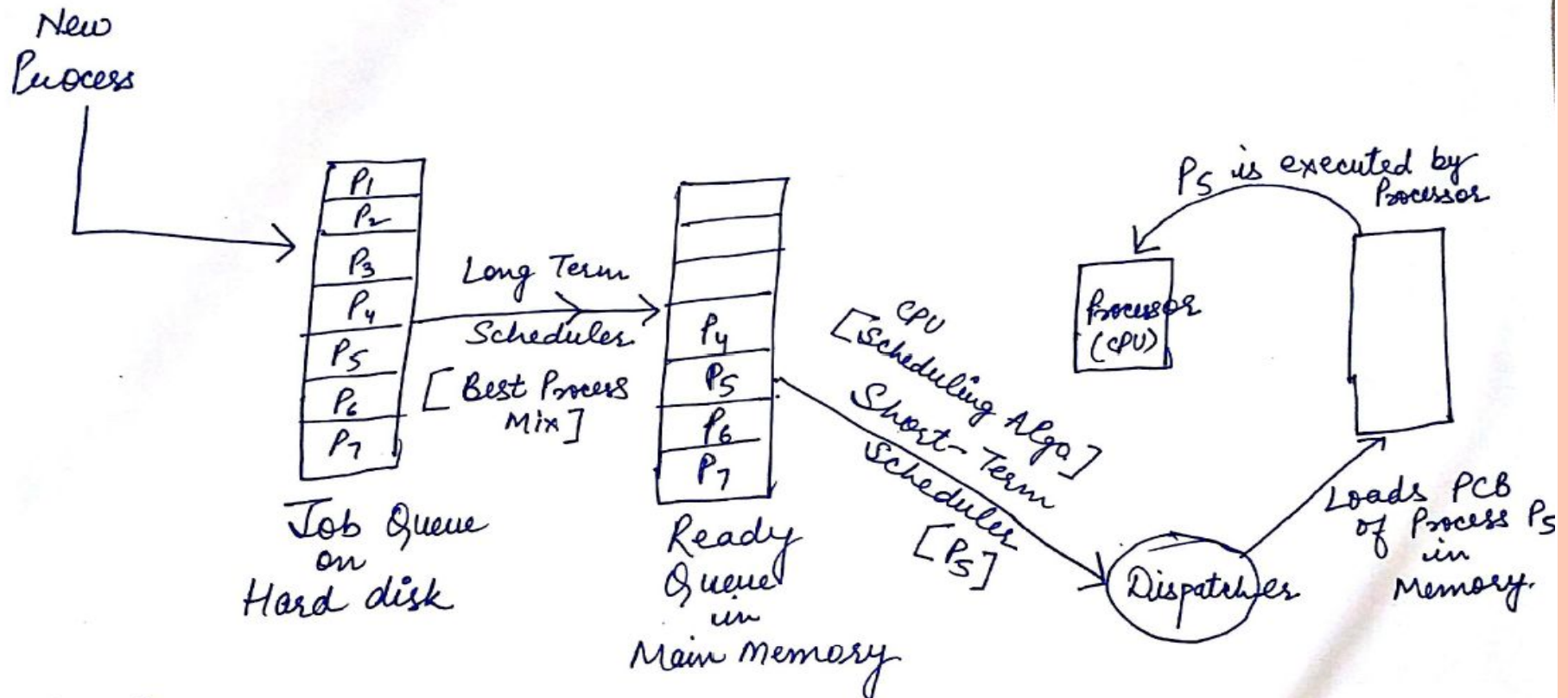
# CONTEXT SWITCHING

- When CPU switches to another process, the system must **save the state** of the old process and load the **saved state** for the new process via a **context switch**
- **Context** of a process represented in the PCB
- Context-switch time is overhead; the system does no useful work while switching
  - The more complex the OS and the PCB □ the longer the context switch
- Time dependent on hardware support
  - Some hardware provides multiple sets of registers per CPU □ multiple contexts loaded at once



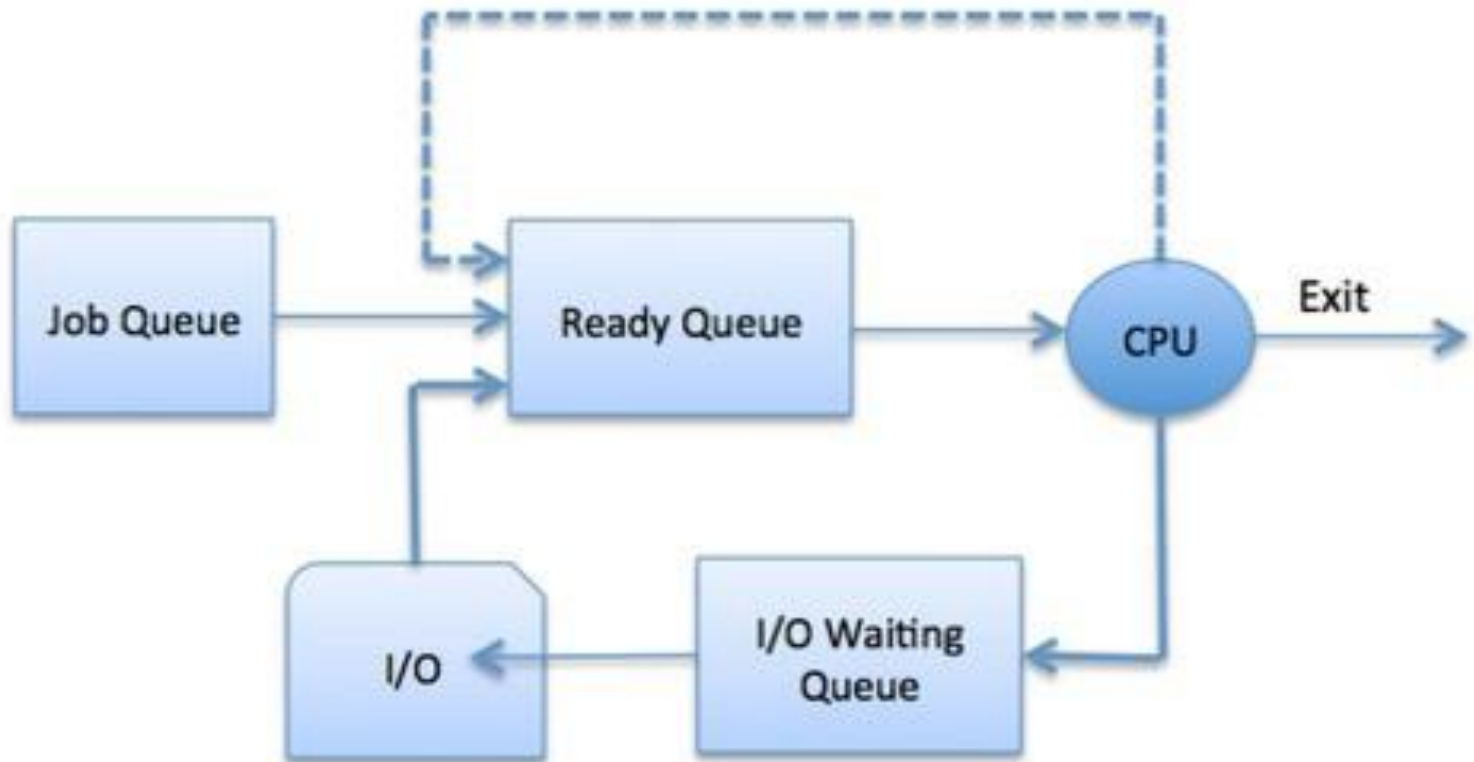
# DISPATCHER..

It unloads the PCB of the old-Process from main memory and loads the PCB of new process so that newly arrived process can execute

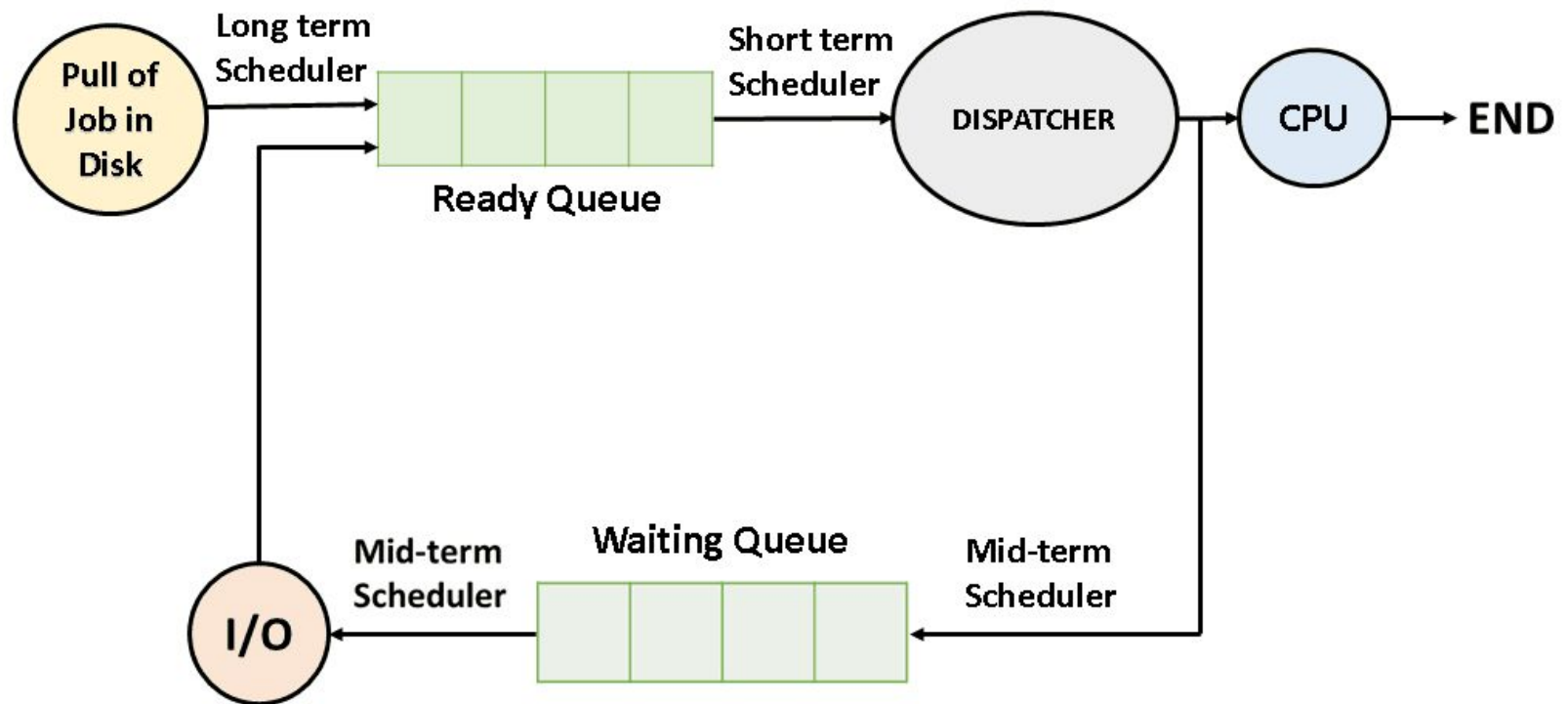


P<sub>3</sub>, P<sub>1</sub>, P<sub>5</sub>, P<sub>7</sub> → CPU Bound Process

P<sub>2</sub>, P<sub>4</sub>, P<sub>6</sub>, P → I/O Bound Process







created by NotesJam



# CPU SCHEDULING



## CPU SCHEDULING CRITERIA

- ❑ **CPU utilization** – keep the CPU as busy as possible
- ❑ **Throughput** – # of processes that complete their execution per time unit
- ❑ **Turnaround time** – amount of time to execute a particular process
- ❑ **Waiting time** – amount of time a process has been waiting in the ready queue
- ❑ **Response time** – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)



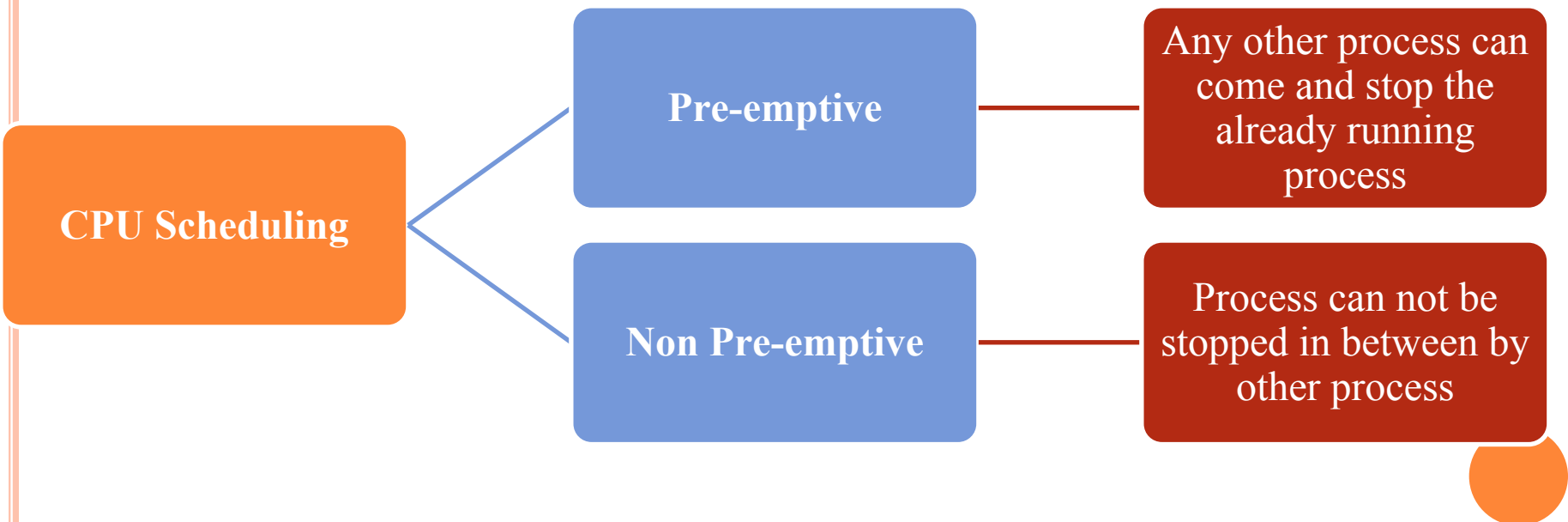
# SCHEDULING ALGORITHM..OPTIMIZATION CRITERIA

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time



# TYPES OF SCHEDULING ALGORITHM

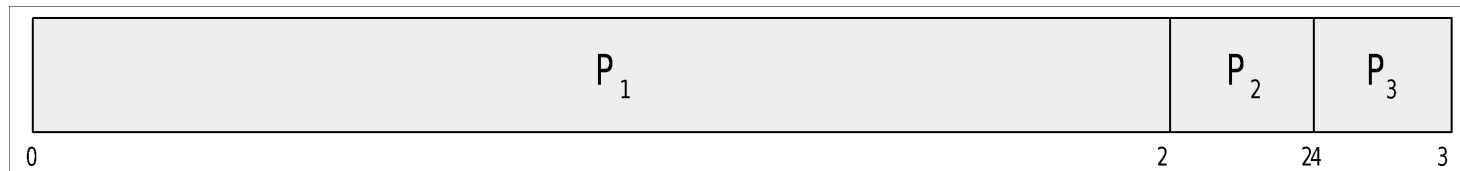
- CPU Scheduling algorithms can be either Pre-emptive or Non-Preemptive.



# FCFS (FIRST COME FIRST SERVE) SCHEDULING

<u>Process</u>	<u>Burst Time</u>
$P_1$	24
$P_2$	3
$P_3$	3

- Suppose that the processes arrive in the order:  $P_1, P_2, P_3$   
The Gantt Chart for the schedule is:



- Waiting time for  $P_1 = 0$ ;  $P_2 = 24$ ;  $P_3 = 27$
- Average waiting time:  $(0 + 24 + 27)/3 = 17$



# FCFS (FIRST COME FIRST SERVE)

## SCHEDULING.....

Suppose that the processes arrive in the order:

$$P_2, P_3, P_1$$

□ The Gantt chart for the schedule is:



- Waiting time for  $P_1 = 6$ ;  $P_2 = 0$ ;  $P_3 = 3$
- Average waiting time:  $(6 + 0 + 3)/3 = 3$
- Much better than previous case
- **Convoy effect** - short process behind long process
  - Consider one CPU-bound and many I/O-bound processes





## SJF (SHORTEST JOB FIRST) SCHEDULING

- Associate with each process the length of its next CPU burst
  - Use these lengths to schedule the process with the shortest time
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request
  - Could ask the user



# Shortest Job First

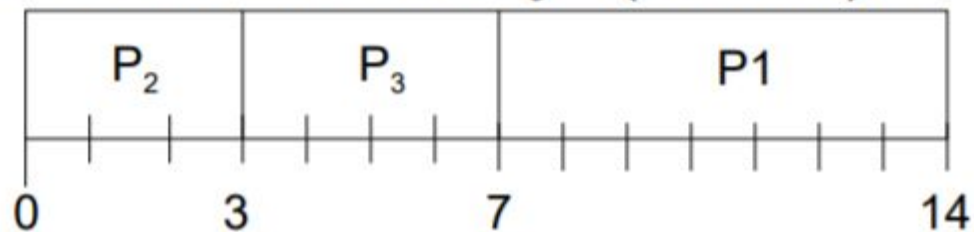
- Associate with each process the length of its next CPU burst.
- Use these lengths to schedule the process with the shortest time.
- Two schemes:
  - **Non-Preemptive:** once CPU given to the process it cannot be preempted until completes its CPU burst.
  - **Preemptive:** if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the Shortest-Remaining-Time-First (SRTF).
- **SJF is optimal:** gives minimum average waiting time for a given set of processes.



### Normal SJF

<u>Process</u>	<u>Burst Time</u>
$P_1$	7
$P_2$	3
$P_3$	4

- The Gantt Chart for SJF (Normal) is:



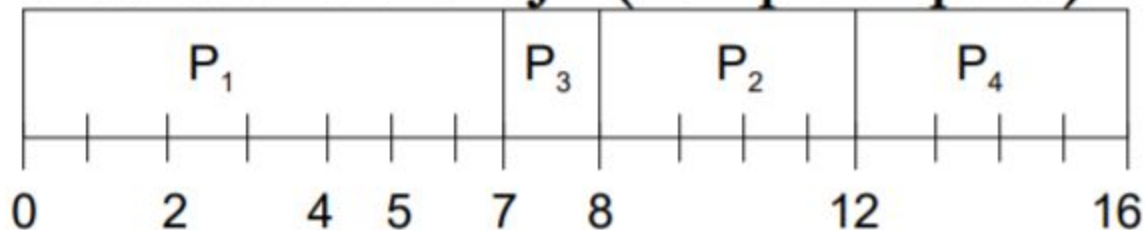
- Average waiting time =  $(0 + 3 + 7)/3 = 3.33$



## Non-Preemptive SJF

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
$P_1$	0.0	7
$P_2$	2.0	4
$P_3$	4.0	1
$P_4$	5.0	4

- The Gantt Chart for SJF (non-preemptive) is:



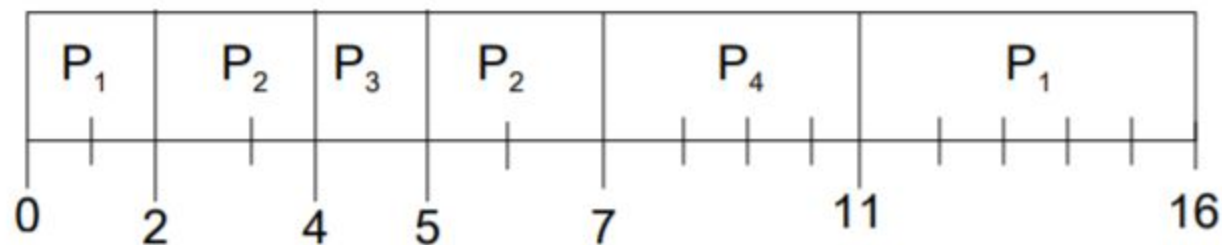
- Average waiting time =  $(0 + 6 + 3 + 7)/4 = 4$

# SRTF (SHORTEST REMAINING TIME FIRST) SCHEDULING

## Preemptive SJF

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
<i>P1</i>	0.0	7
<i>P2</i>	2.0	4
<i>P3</i>	4.0	1
<i>P4</i>	5.0	4

- The Gantt Chart for SJF (preemptive) is:



- Average waiting time =  $(9 + 1 + 0 + 2)/4 = 3$

# PRIORITY SCHEDULING

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer  $\equiv$  highest priority)
  - Preemptive
  - Nonpreemptive
- SJF is priority scheduling where priority is the inverse of predicted next CPU burst time
- Problem  $\equiv$  **Starvation** – low priority processes may never execute
- Solution  $\equiv$  **Aging** – as time progresses increase the priority of the process



# Priority Scheduling - Example

*Lower priority # == More important*

Process	Duration	Priority #	Arrival Time
P1	6	4	0
P2	8	1	0
P3	7	3	0
P4	3	2	0



P2 waiting time: 0  
P4 waiting time: 8  
P3 waiting time: 11  
P1 waiting time: 18

The average waiting time (AWT):  
 $(0+8+11+18)/4 = 9.25$   
(worse than SJF's)

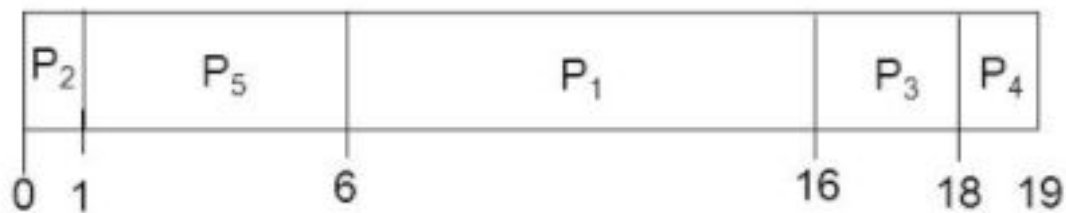




## Example 1 of Non-Preemptive Priority

<u>Process</u>	<u>Burst Time</u>	<u>Priority</u>	<u>arrival time</u>
$P_1$	10	3	0
$P_2$	1	1	0
$P_3$	2	4	0
$P_4$	1	5	0
$P_5$	5	2	0

➤ Priority Scheduling (non-preemptive)



➤ Average waiting time =  $(0 + 1 + 6 + 16 + 18)/5 = 8.2$

# Example: Preemptive Priority Sch.

Average Waiting time:

$$P1 = 22 - 1 = 21$$

$$P2 = 21 - 2 = 19$$

$$P3 = 19 - 3 = 16$$

$$P4 = 9 - 5 = 4$$

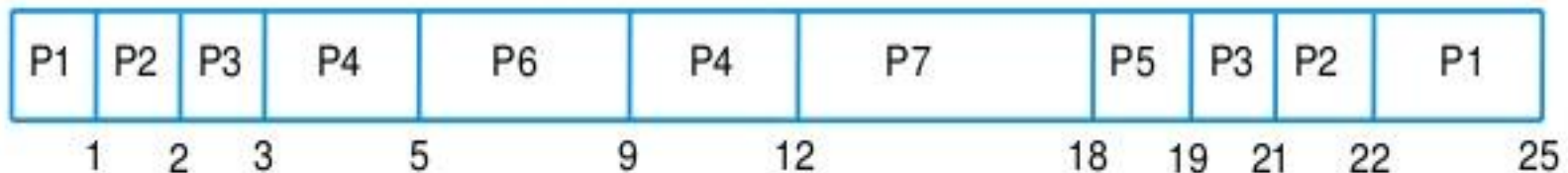
$$P5 = 18 - 4 = 14$$

$$P6 = 5 - 5 = 0$$

$$P7 = 12 - 6 = 6$$

Average Waiting time = 11.43ms

P	Priority	AT	CBT
1	6	0	<del>4</del> <del>3</del> 0
2	5	1	<del>2</del> <del>1</del> 0
3	4	2	<del>3</del> <del>2</del> 0
4	1	3	<del>5</del> <del>3</del> 0
5	3	4	<del>1</del> 0
6	0	5	<del>4</del> 0
7	2	6	<del>6</del> 0



# ROUND ROBIN SCHEDULING

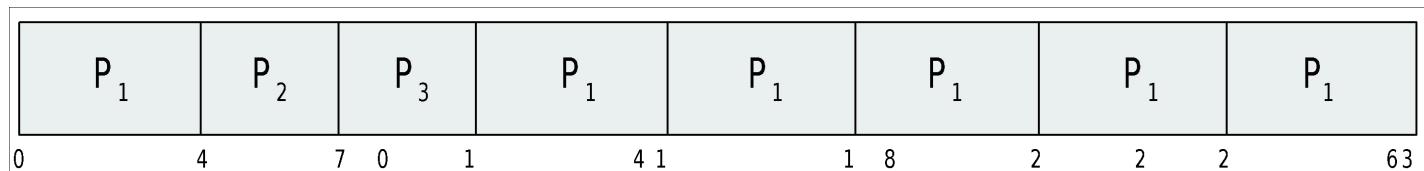
- Each process gets a small unit of CPU time (**time quantum  $q$** ), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are  $n$  processes in the ready queue and the time quantum is  $q$ , then each process gets  $1/n$  of the CPU time in chunks of at most  $q$  time units at once. No process waits more than  $(n-1)q$  time units.
- Timer interrupts every quantum to schedule next process
- Performance
  - $q$  large  $\Rightarrow$  FIFO
  - $q$  small  $\Rightarrow q$  must be large with respect to context switch, otherwise overhead is too high



# ROUND ROBIN SCHEDULING

Process	Burst Time
$P_1$	24
$P_2$	3
$P_3$	3

- The Gantt chart is: [time quantum=4]



- Typically, higher average turnaround than SJF, but better *response*
- $q$  should be large compared to context switch time
- $q$  usually 10ms to 100ms, context switch  $< 10$  usec





# Round-Robin Scheduling

<u>Process</u>	<u>Burst Time</u>
----------------	-------------------

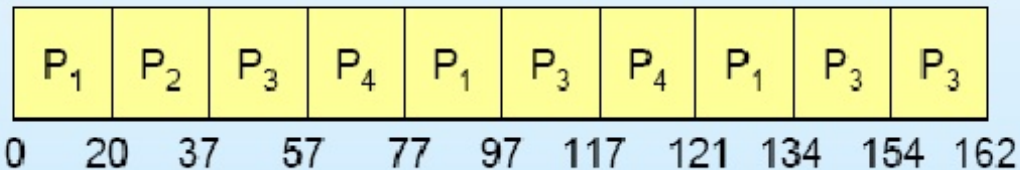
$P_1$	53
-------	----

$P_2$	17
-------	----

$P_3$	68
-------	----

$P_4$	24
-------	----

The Gantt chart is:



Time Quantum=20



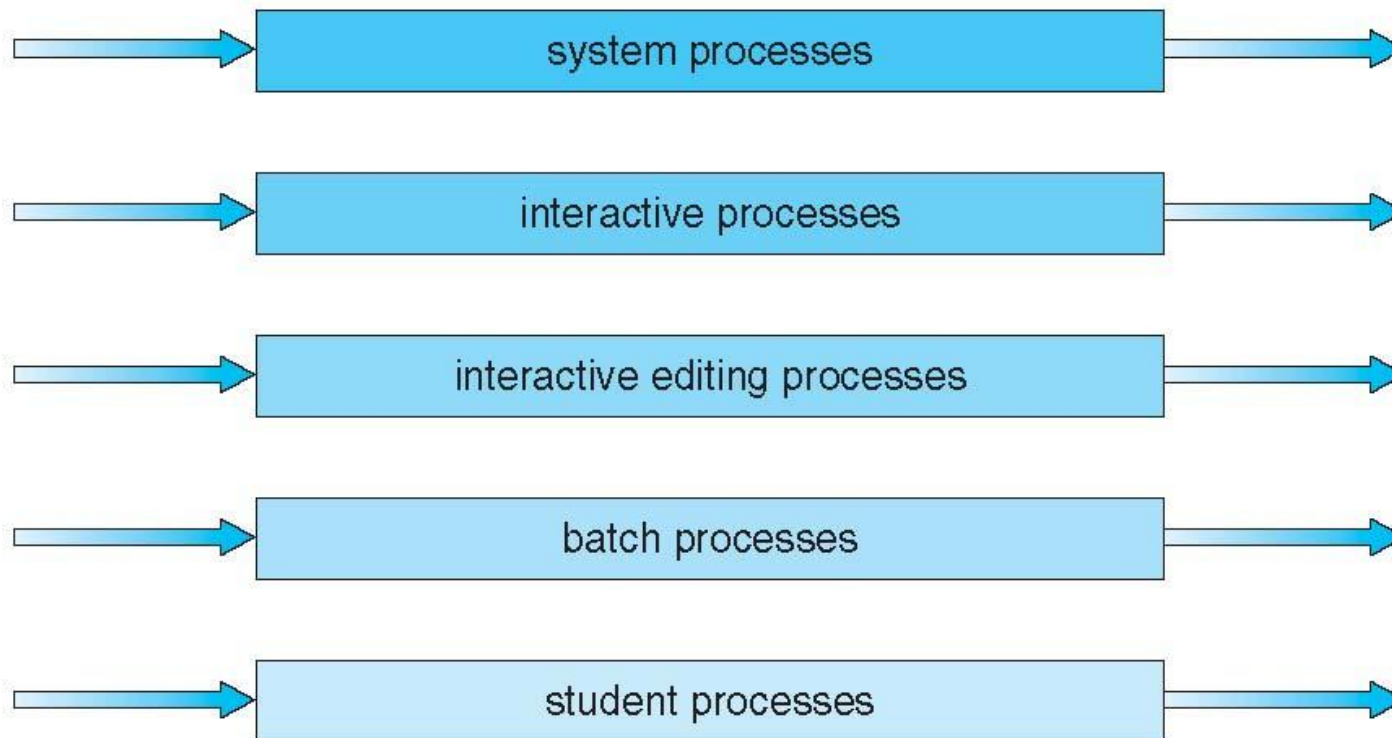
# MULTILEVEL QUEUE SCHEDULING

- Ready queue is partitioned into separate queues, eg:
  - **foreground** (interactive)
  - **background** (batch)
- Process permanently in a given queue
- Each queue has its own scheduling algorithm:
  - foreground – RR
  - background – FCFS
- Scheduling must be done between the queues:
  - Fixed priority scheduling; (i.e., serve all from foreground then from background). Possibility of starvation.
  - Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR
  - 20% to background in FCFS



# MULTILEVEL QUEUE SCHEDULING..

highest priority



lowest priority



# MEMORY MANAGEMENT



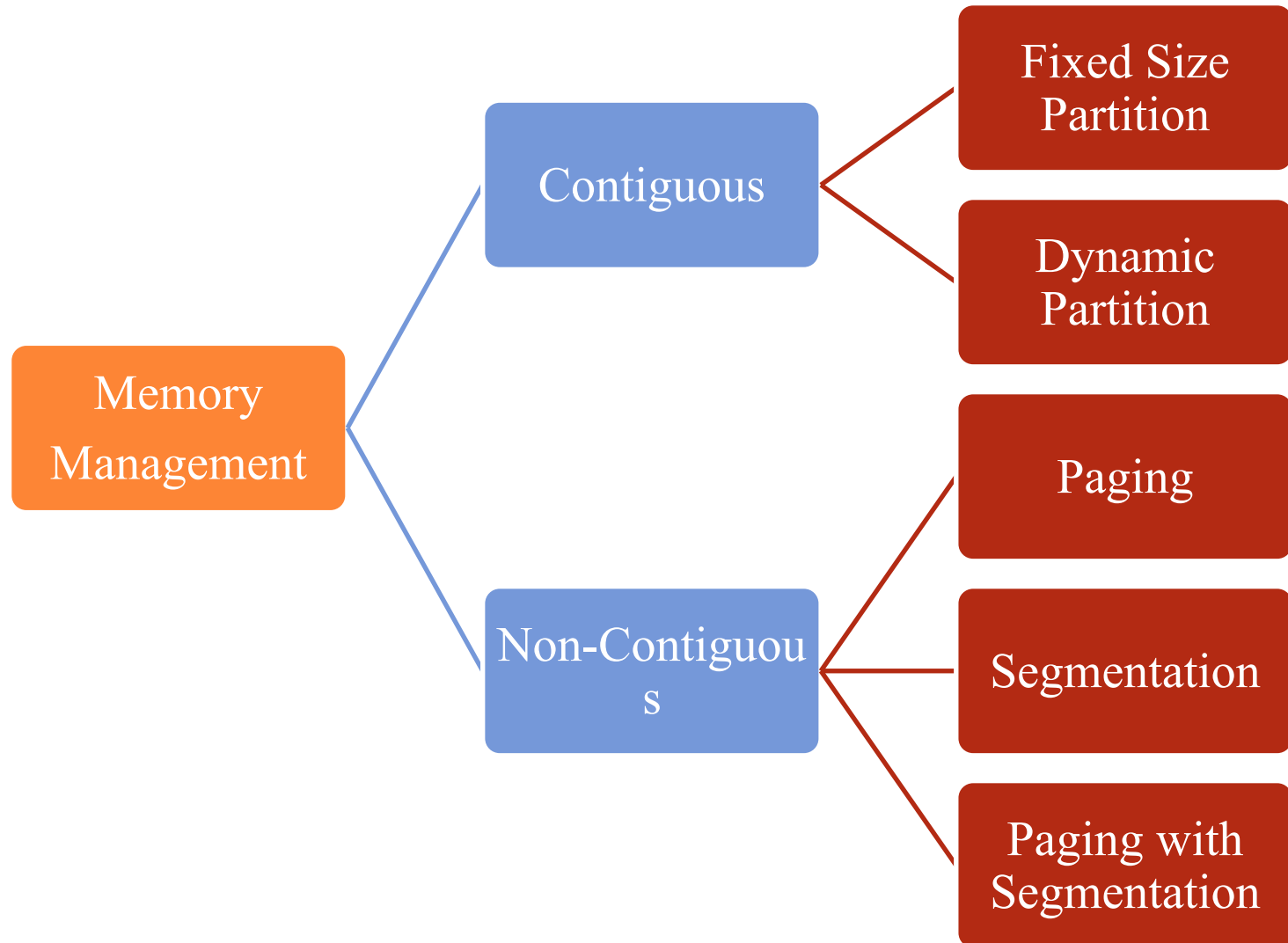


# MEMORY MANAGEMENT

- One of the main functions of O.S. is memory management. O.S. allocate available memory efficiently to multiple processes.
  
- **Main functions**
  - Allocate memory to processes when needed
  - Keep track of what memory is used and what is free
  - Protect one process's memory from another

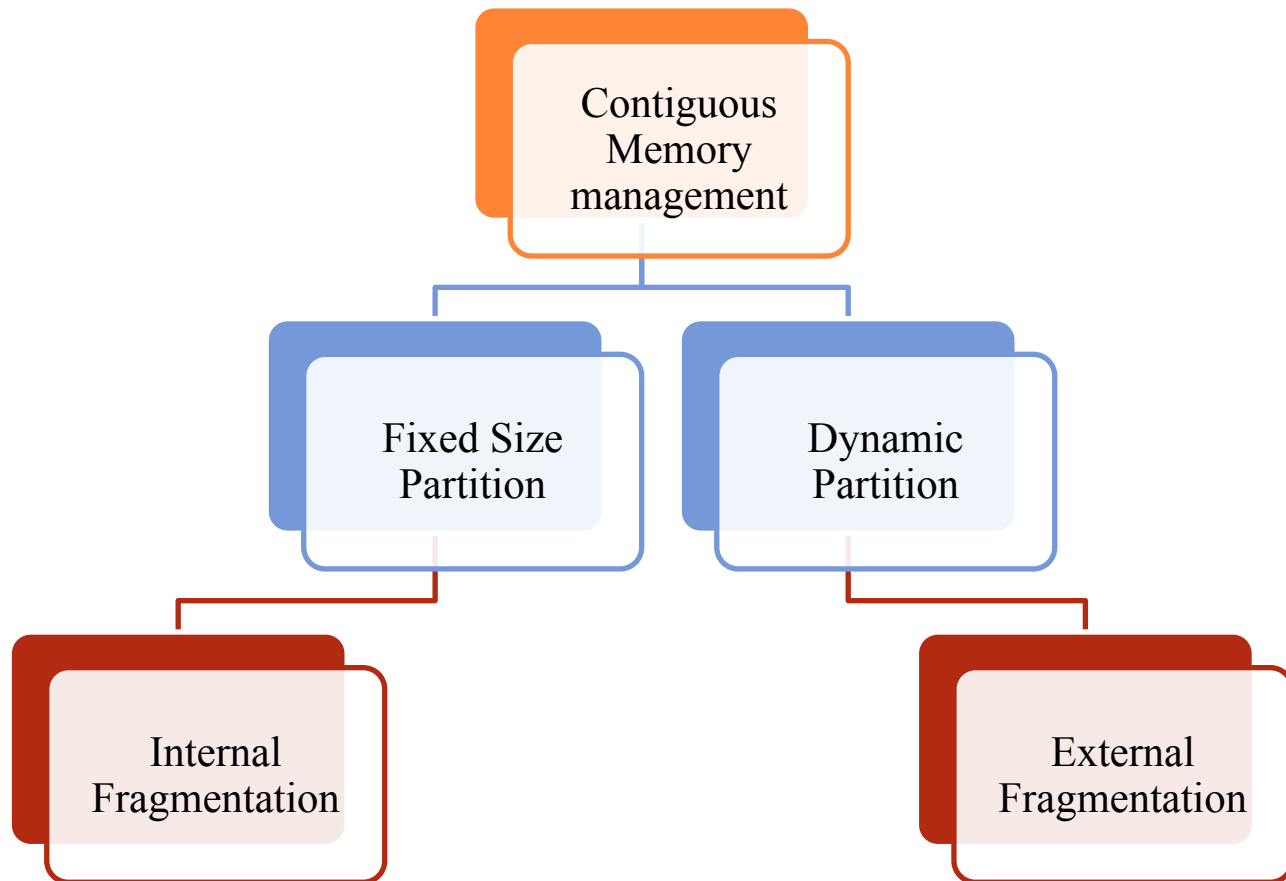


# TYPES OF MEMORY MANAGEMENT SCHEMES



# CONTIGUOUS MEMORY MANAGEMENT

Each process is allocated a single contiguous chunk of memory..



# CONTIGUOUS MEMORY MANAGEMENT...(FPS)

## ❑ **Fixed Partition Scheme (FPS)**

- Memory broken up into fixed size partitions Each partition can have exactly one process When a process arrives, allocate it a free partition

## ❑ **Problems:**

- Maximum size of process bound by max. partition size
- Large **Internal fragmentation** possible



# FIXED SIZE PARTITION

## Fixed Partition Scheme-

Suppose , memory is divided into equal size blocks of 4 KB. So, size of each block=4KB

If a process P1 needs 3KB memory, we allocate first block to it.

If a process P2 needs 5 KB memory, we allocate two blocks to it.

## Internal Fragmentation (Wastage of memory)

[illegible]

## DYNAMIC SIZE PARTITION

In Dynamic Partition Scheme- memory is treated as a big empty hole initially, As the Process arrives, required memory is allocated to it in continuous order.

Holes are blocks of available memory

- Holes of various size are scattered throughout memory.
- When a process arrives, it is allocated memory from a hole large enough to accommodate it.
- Operating system maintains information about: – allocated partitions – free partitions (i.e., holes)



# DYNAMIC SIZE PARTITION

In Dynamic Partition Scheme- Memory is treated as a big empty hole initially,

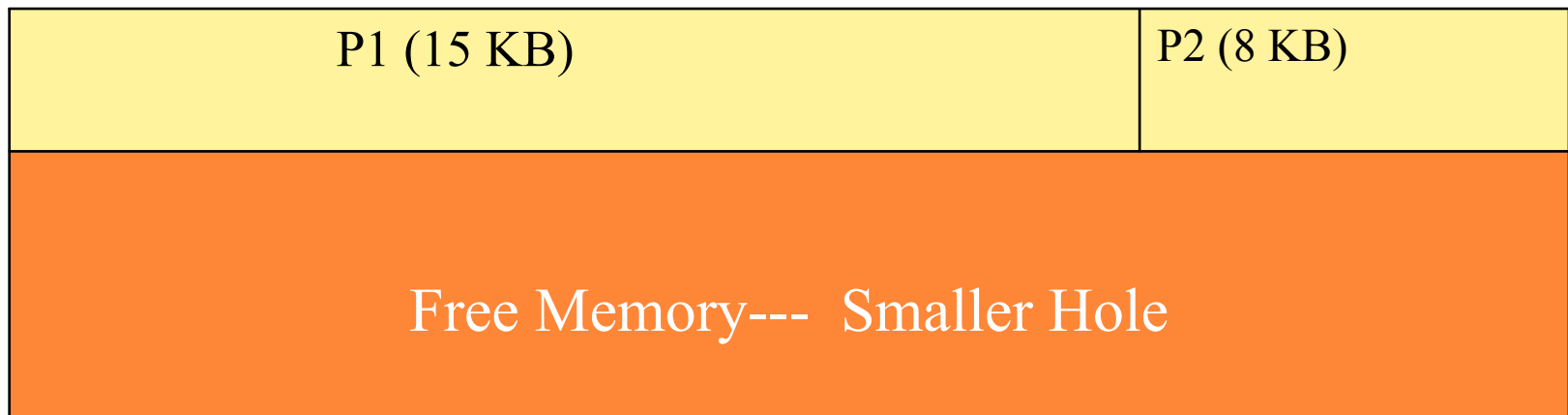
Free Memory ---Big hole Initially

As the Process arrives, required memory is allocated to it in continuous order.



## DYNAMIC SIZE PARTITION

As the Process arrives, required memory is allocated to it in continuous order. Now, if P1 (15 KB) and P2 (8 KB) arrives, then....





## DYNAMIC SIZE PARTITION

Now P3 (12 KB) , P4 (11KB) , P5 (10 KB) and P6 (13 KB) Arrives,  
then Memory Allocation would be...

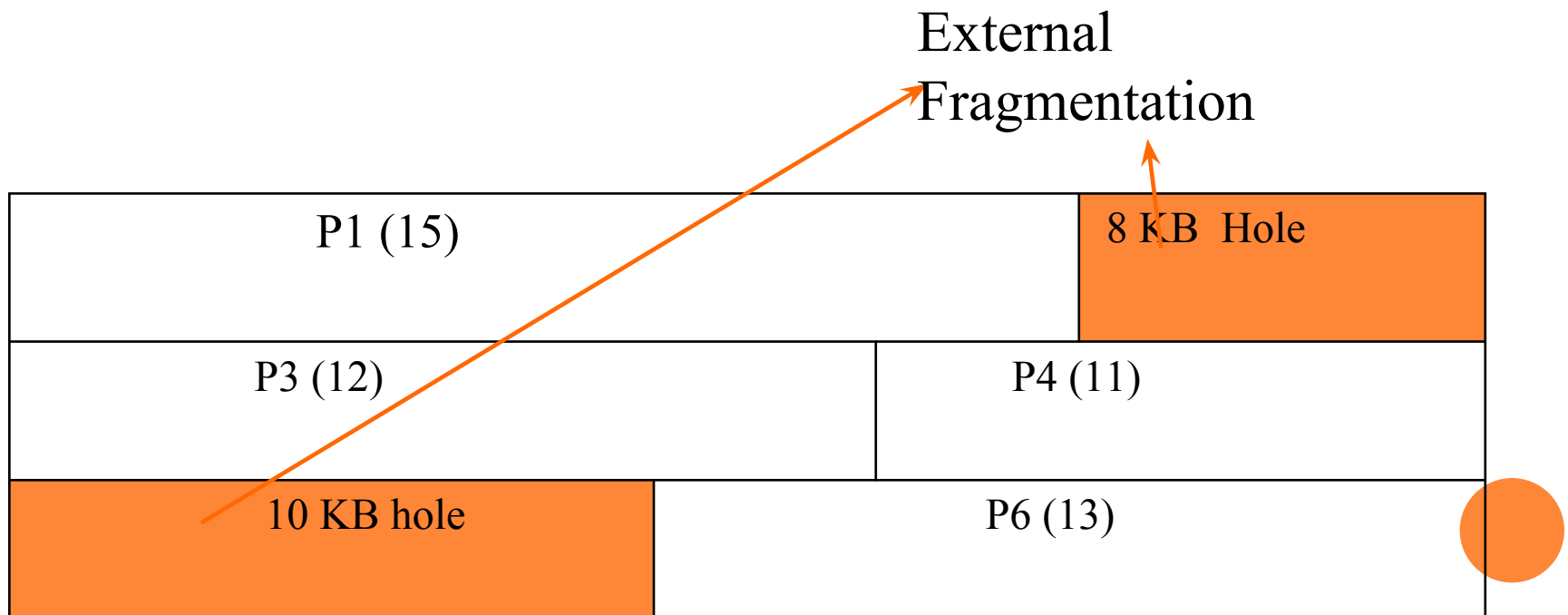
P1 (15 KB)		P2 (8 KB)
P3 (12 KB)	P4 (11 KB)	
P5 (10 KB)	P6 (13 KB)	

Now Memory is fully .....Allocated....



## DYNAMIC SIZE PARTITION

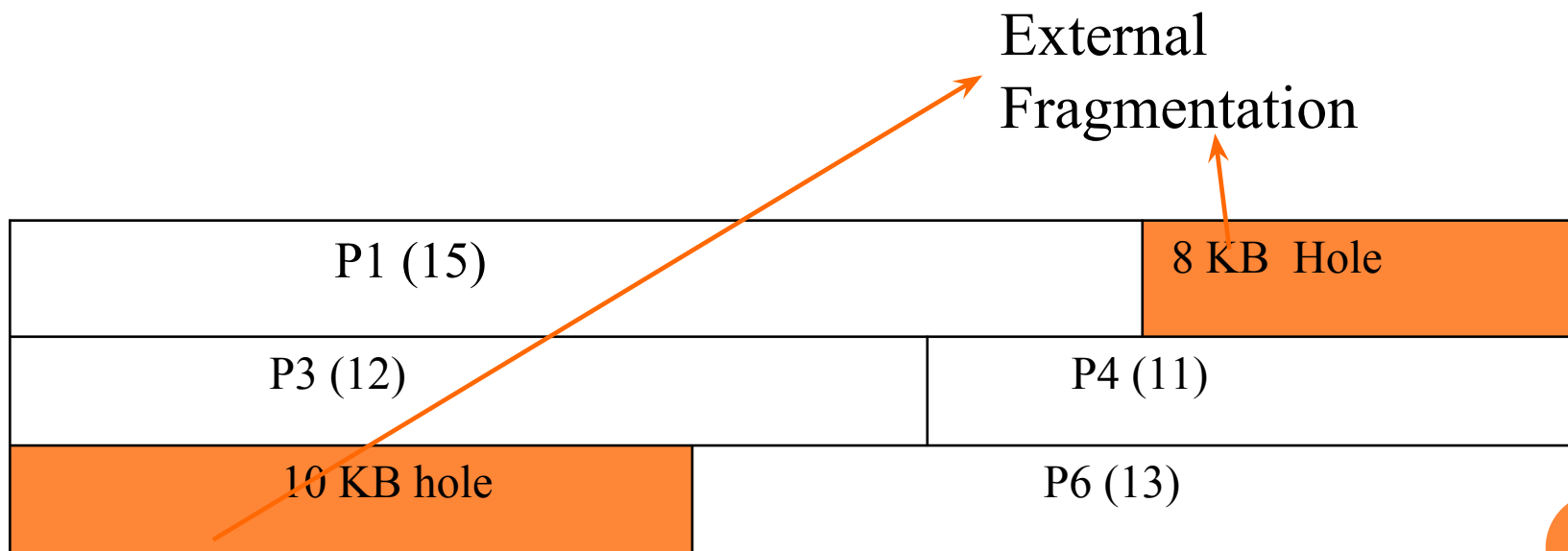
Now, If P2 and P5 completes the execution and leave the memory...18 KB space becomes free..but not in continuity...[Problem of External Fragmentation]



# DYNAMIC SIZE PARTITION

## Problem of External Fragmentation:

If a Process P7 of 12 KB arrives, we can not allocate memory to it because 12 KB memory is not free in continuous order.....[we have two free holes : 10 KB and 8 KB, so total 18 KB free memory is available, but in scattered form]



# DYNAMIC SIZE PARTITION

## Solution of External Fragmentation:--Compaction

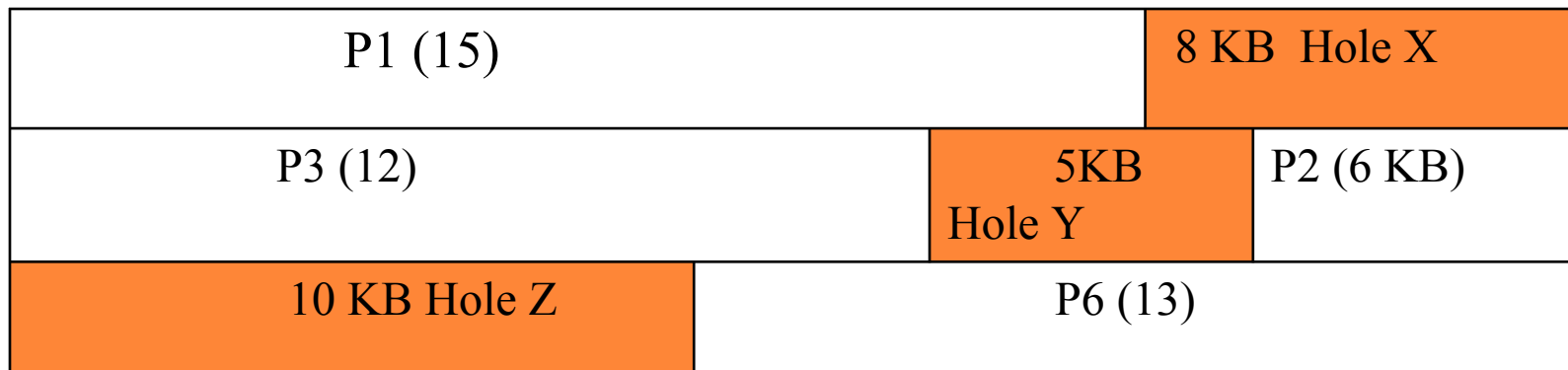
Compaction compacts the freely available holes at one place...and allocated memory at another place...

P1 (15)		P2 (8)
P2 (4)	P4 (11)	P6 (8)
P6 (5)	(18 KB free memory hole) After Compaction	

# DYNAMIC STORAGE ALLOCATION PROBLEM

In the diagram depicted below, we have many free holes...and a process P8 having size=5KB arrives, then there are three policies to allocate free holes:

- **First-fit:** Allocate the first hole that is big enough. (Hole X)
- **Best-fit:** Allocate the smallest hole that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole. (Hole Y)
- **Worst-fit:** Allocate the largest hole; must also search entire list. Produces the largest leftover hole (Hole Z)



## TYPES OF FRAGMENTATION.....

- ❑ **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous.
  - Reduce external fragmentation by compaction – Shuffle memory contents to place all free memory together in one large block.
- ❑ **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.



# INTRODUCTION OF UNIX



# UNIX

- A multi-user networked operating system developed by  
“**Ken Thompson**”
- “**Operating System**”
  - Handles files, running other programs, input/output
  - Looks like DOS...but more powerful
  - The internet was designed on it, thus networking is an intrinsic part of the system
- “**Multi-user**”
  - Every user has different settings and permissions
  - Multiple users can be logged in simultaneously





# ADVANTAGES OF UNIX

□ Advantages of UNIX are...

- **Multitasking** – multiple programs can run at one time.
- **Multi-user** – allows more than a single user to work at any given time. This is accomplished by sharing processing time between each user.
- **Safe (Security)** – prevents one program from accessing memory or storage space allocated to another program, and enables file protection, requiring users to have permission to perform certain functions, such as accessing a directory, file, or disk drive.



# UNIX HISTORY

The Unix operating system found its beginnings in MULTICS, which stands for Multiplexed Operating and Computing System. The MULTICS project began in the mid 1960s as a joint effort by General Electric, Massachusetts Institute for Technology and Bell Laboratories. In 1969 Bell Laboratories pulled out of the project.

One of Bell Laboratories people involved in the project was Ken Thompson. He liked the potential MULTICS had, but felt it was too complex and that the same thing could be done in simpler way. In 1969 he wrote the first version of Unix, called UNICS. UNICS stood for Uniplexed Operating and Computing System. Although the operating system has changed, the name stuck and was eventually shortened to Unix.

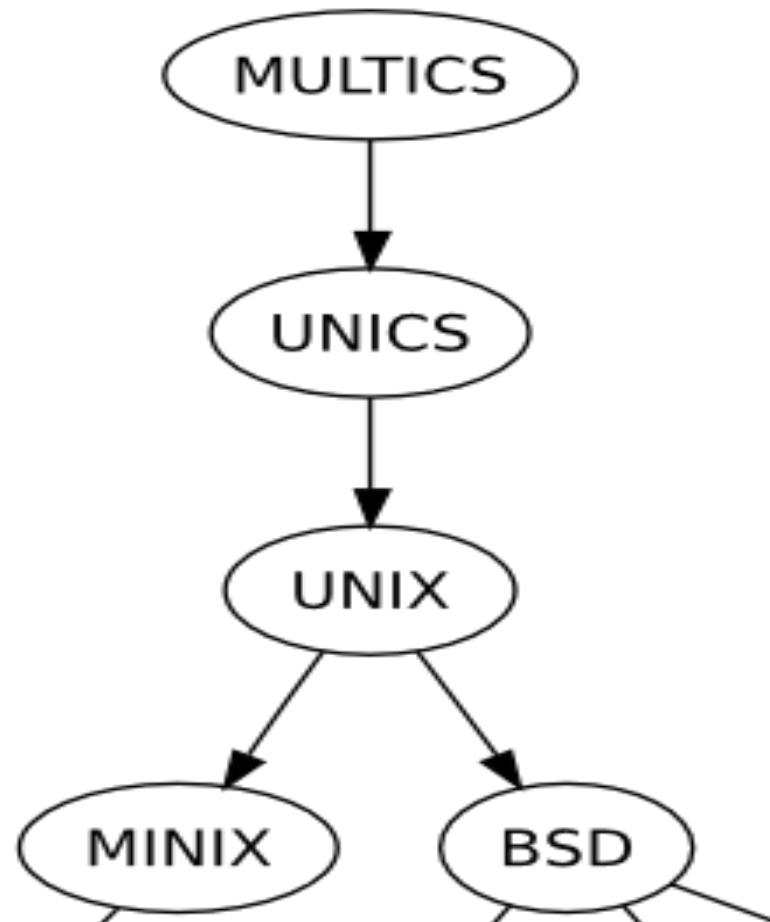
# UNIX HISTORY

Ken Thompson teamed up with Dennis Ritchie, who wrote the first C compiler. In 1973 they rewrote the Unix kernel in C. The following year a version of Unix known as the Fifth Edition was first licensed to universities. The Seventh Edition, released in 1978, served as a dividing point for two divergent lines of Unix development. These two branches are known as SVR4 (System V) and BSD.

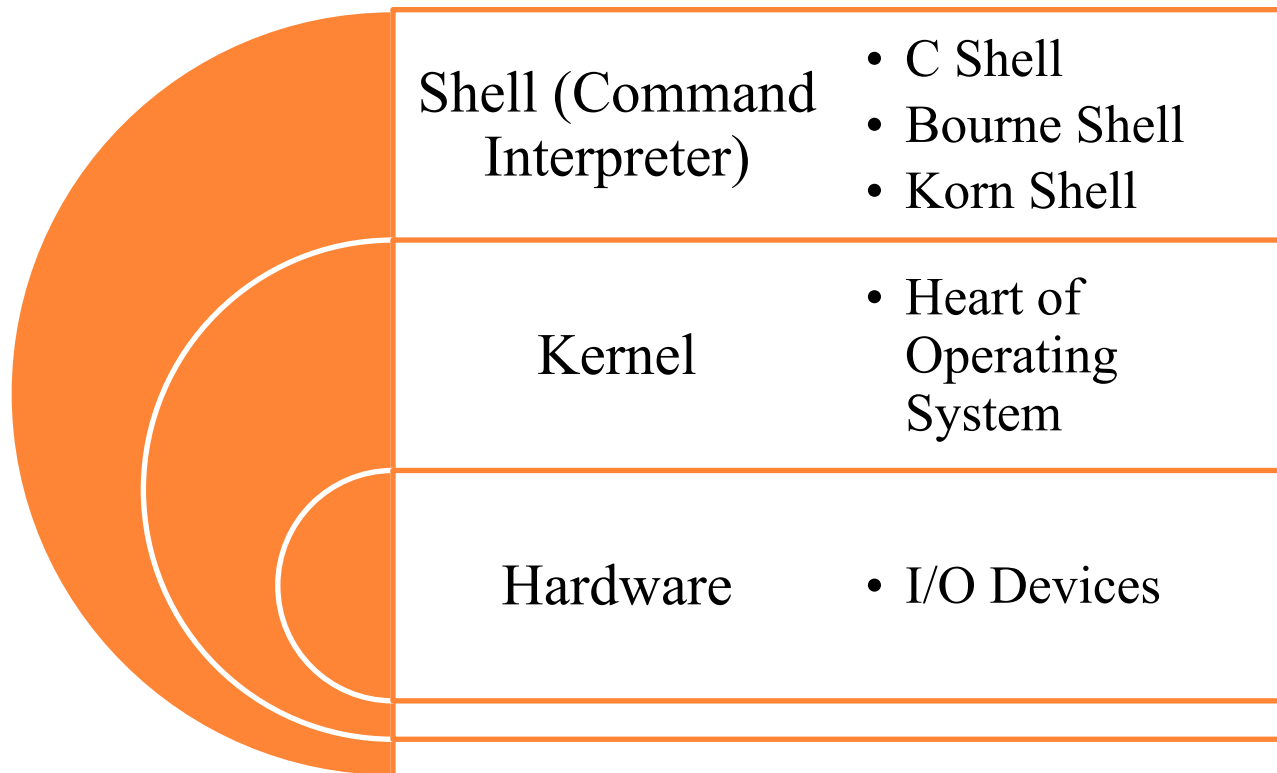
Ken Thompson spent a year's sabbatical with the University of California at Berkeley. While there he and two graduate students, Bill Joy and Chuck Haley, wrote the first Berkely version of Unix, which was distributed to students. This resulted in the source code being worked on and developed by many different people. The Berkeley version of Unix is known as BSD, Berkeley Software Distribution. From BSD came the vi editor, C shell, virtual memory, Sendmail, and support for TCP/IP.



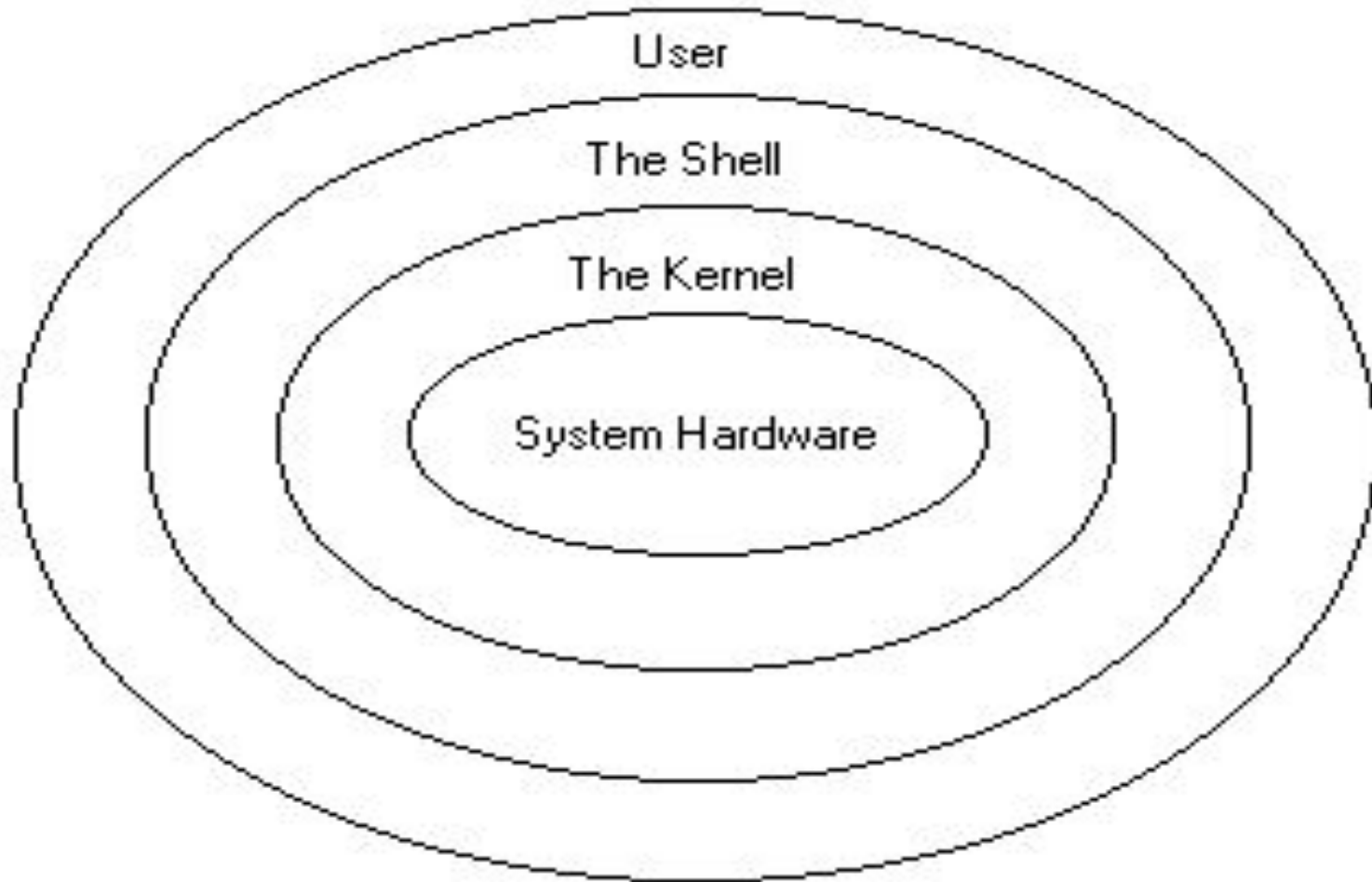
# UNIX HISTORY



# UNIX STRUCTURE



## UNIX STRUCTURE CONTINUED...



# VI EDITOR

- The **vi editor** (short for visual **editor**) is a screen **editor** which is available on almost all Unix systems. **vi** has no menus but instead uses combinations of keystrokes in order to accomplish commands. It is used to create shell scripts and file editing.
- VI Editor works in two modes :
  - **Command Mode**
  - **Insert Mode**



# MODES OF VI EDITOR

- ❑ **Command mode** – This mode enables you to perform administrative tasks such as saving files, executing commands, moving the cursor, cutting (yanking) and pasting lines or words, and finding and replacing. In this mode, whatever you type is interpreted as a command.
- ❑ **Insert mode** – This mode enables you to insert text into the file. Everything that's typed in this mode is interpreted as input and finally it is put in the file





Command	Description
cat	Display File Contents
cd	Changes Directory to dirname
chgrp	change file group
chmod	Changing Permissions
cp	Copy source file into destination
file	Determine file type
find	Find files
grep	Search files for regular expressions.
head	Display first few lines of a file
ln	Create softlink on oldname
ls	Display information about file type.
mkdir	Create a new directory dirname
more	Display data in paginated form.
mv	Move (Rename) a oldname to newname.
pwd	Print current working directory.
rm	Remove (Delete) filename
rmdir	Delete an existing directory provided it is empty.
tail	Prints last few lines in a file.
touch	Update access and modification time of a file.

# WINDOWS vs. UNIX



# WINDOWS O.S.

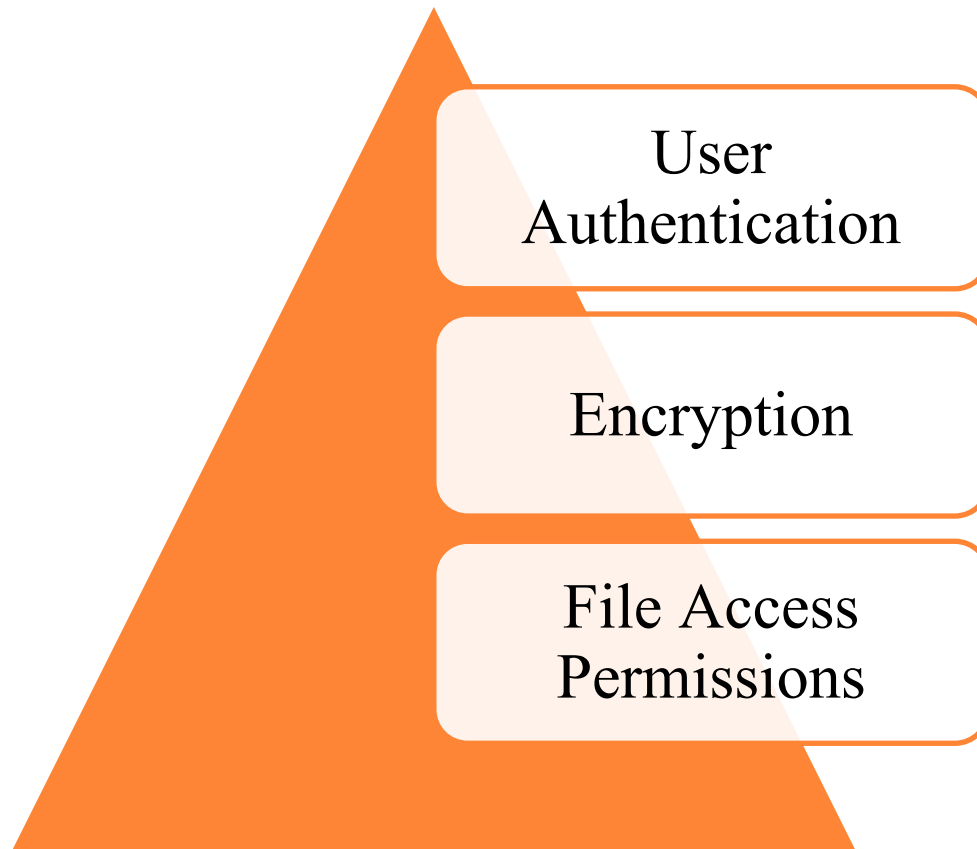
Windows based operating system was founded by Microsoft when they first developed the MS-DOS or PC-DOS operating system to be used with the first IBM personal computers in the year 1981. In 1990, 80486 processor and Intel Pentium chips had been introduced where they provided power and features that could accommodate graphical user interface which could be used to interpose between DOS and users, and so Microsoft released Windows 3.0. In 1993 Microsoft released Window NT (3.1) which operated on 32-bit operating system and had the ability to support windows based applications, older DOS, and OS/2. In 2000, Microsoft released Windows 2000 which had added features such as active directory, the ability to handle more services and supported distribution processing. In the year 2001, Microsoft released latest desktop version of windows called Windows XP which operated on both 32 and 64 bit machines. In 2006 windows vista was released which was an upgrade of windows XP in terms of performance and security. In 2009, Windows 7 was released as an upgrade of windows vista where it provided speed, stability and minimal system requirements, which is currently used. In the year 2012, Microsoft released windows 8 which have redesigned metro-style user interface, virtual hard disk, integrated antivirus, and other applications where as per now it is getting a lot of intention in the world of computers.

## DIFFERENCE ..B/W WINDOWS AND UNIX

- ❑ Source Code of Unix is open-Source i.e. Freely available. Anyone can download it and customize it by making his/her own changes. Source Code for Windows is not free.
- ❑ Windows is based on GUI (Graphical User Interface) and UNIX is based on CLI (Command Line Interface).
- ❑ Security in UNIX is much better than Windows as UNIX uses 3 Layers of Security.



# UNIX SECURITY LAYERS



# CONCLUSION

During this session, we have covered the following topics

- ❑ Operating System Definition
- ❑ Services and Objectives of Operating System
- ❑ Process Concepts and PCB
- ❑ Process States
- ❑ Schedulers and its types
- ❑ Context Switching
- ❑ CPU Scheduling Algorithms
- ❑ Memory Management Schemes
- ❑ UNIX Operating System



THANK YOU !!

