

# CSA04-OPERATING SYSTEMS

## Concept Mapping

### Topic 1: Introduction to Operating Systems

- Introduction
- OS History
- Mainframe Systems
- Desktop Systems
- Multiprocessor Systems
- Distributed Systems
- Clustered Systems
- Realtime Systems
- Handheld Systems

### Topic 2: OS Structure and services

- Operating Systems Structure
- Systems Components
- Operating System Services
- Systems Calls
- Systems Programs

### Topic 3: Systems Design and Implementation

- Functionalities And Characteristics of os
- Hardware Concepts Related to os
- CPU States
- I/O Channels
- Memory Hierarchy
- Microprogramming

### Topic 4: process and Threads

- Process Concept
- Operations on Processes
- Co-operating Processes
- Inter Process Communication, Context Switching Diagram
- Threads, Multithreading Models

### Topic 5: Process Scheduling

- Scheduling Criteria
- Process Scheduling Algorithms.
- Multiprocessor Scheduling
- Real Time Scheduling
- Algorithm Evolution (Evaluation)
- Models.

### Topic 6: Process Synchronization

- Critical Section Problem.
- Synchronization Hardware
- Classical Problems of Synchronization Semaphore
- Monitors

1

### Topic 7: Deadlocks

- Deadlock Characterization
- Methods For Handling Deadlocks
- Prevention
- Avoidance

### Topic 8: Deadlock Detection

- Recovery From Deadlock
- Bankers Algorithm.

2

### Topic 9: Dynamic Storage Management

- Background
- Swapping
- Linkers And Dynamic Linking.
- Contiguous Memory Allocation

### Topic 10: Paging

- Basic Methods
- Hardware Support with Diagram
- Protection
- Structure of Page Table

3

### Topic 11: Segmentation

- Basic Method.
- Hardware
- protection And Sharing
- Fragmentation
- Segmentation With Paging.

4

### Topic 12: Virtual Memory

- Background With Diagram
- Demand Paging
- Page Replacement
- Thrashing And Working Sets
- Disk Devices

5

### Topic 13: File Systems

- File Concepts
- File Types and Structures
- Contiguous
- Sequential
- Indexed
- File Systems Interface
- Access methods
- Directory Structure (Single level, Two level, Three Structured, Graph structured)

6

7

### Topic 14: File Systems Implementation

- Case Study
- Unix File Systems
- Partitioning
- Mounting & Unmounting
- File Sharing
- Unix Semantics
- Buffer Cache
- I nodes
- Systems Calls –Alloc ,Free , Name

9

### Topic 15: NFS and I/O Systems

- DMA
- Application /o Interface.
- Kernel I/O
- Device Drivers
- Block & Character Devices, Streams
- Switch Tables

10

### Topic 16: Disk Scheduling

- FCFS, SSTF, SCAN,
- C-SCAN, C-LOOK, LOOK
- Disk Management
- RAID

11

### Topic 17: Linux Systems.

- Basic Concepts
- Systems Administration
- Requirements Of Administrator
- Administrator Roles

12

### Topic 18: Case Study- Linux setup.

- Setting up Linux Multifunction Server
- DNS
- Hierarchy Of Setting Name Servers
- Setting Up Local Network Services.

13

### Topic 19: Virtualization

- Basic Concept
- Setting up Xen
- VMware
- Linux Host
- Adding Guest OS

14

### Topic 20: Recent Trends in OS

- Quantum Operating Systems.
- Scheduling In Parallel & Distributed Systems
- Internet Scale Computing

15

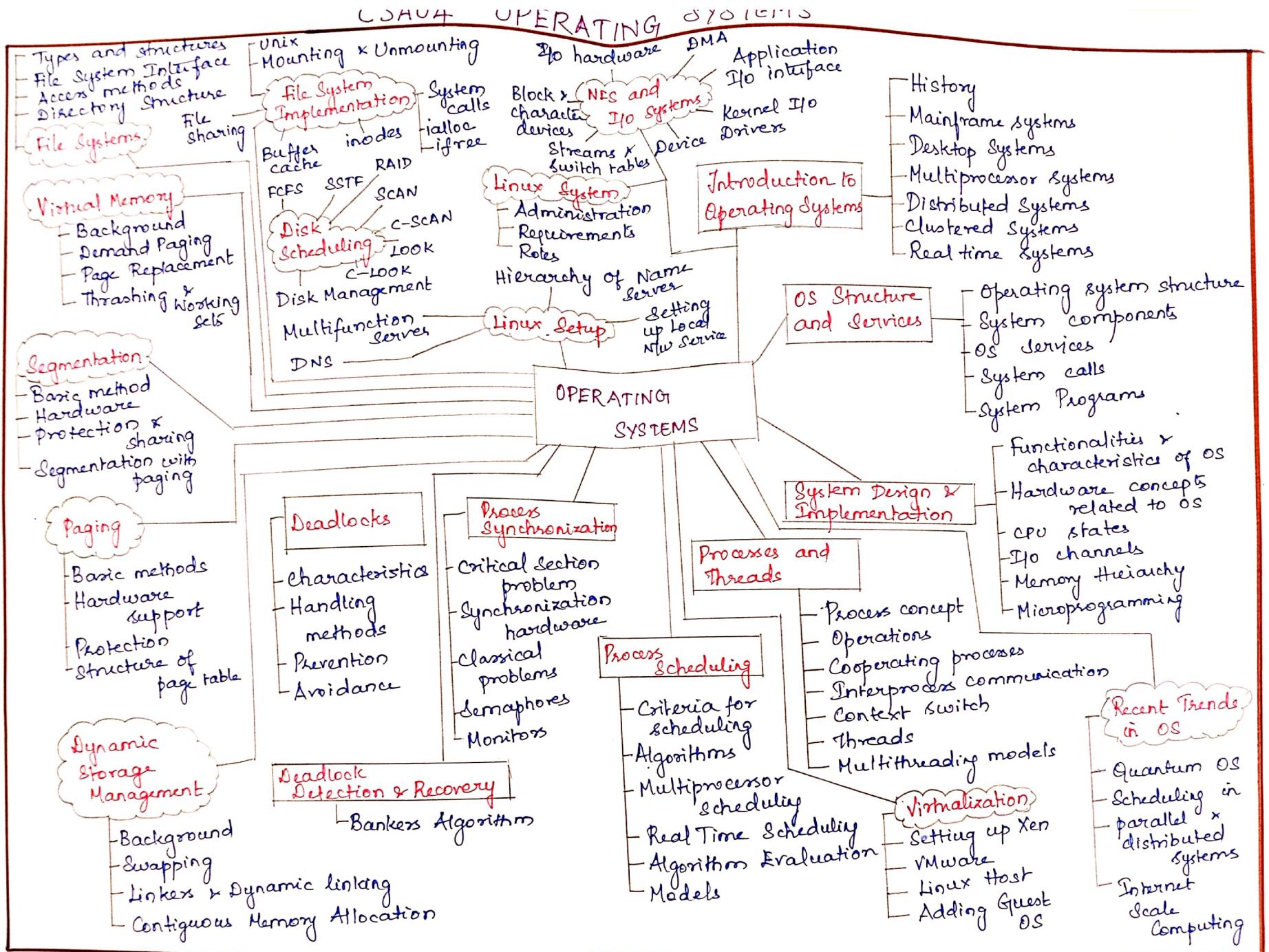
16

17

18

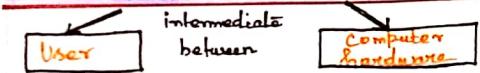
19

20

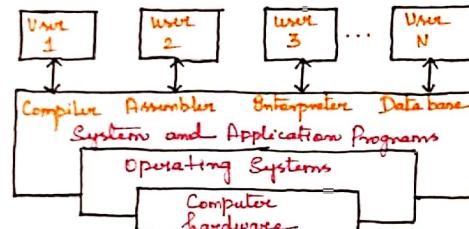


# Topic 1: INTRODUCTION TO OPERATING SYSTEMS

## OPERATING SYSTEM



## Components of Computer Systems



## Basic elements

- Processor
- Main Memory
- I/O modules
- System bus

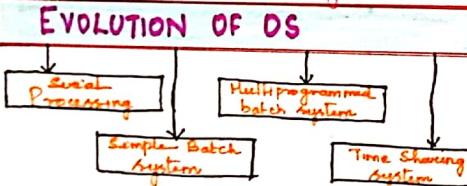
## Convenience

- Efficiency
- Ability to evolve

## Objective

- Resource Allocation
- Memory Management
- Security

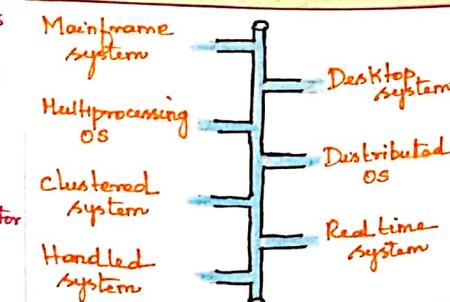
## FUNCTIONS



## \* Serial Processing:

- No OS.
- Run from Console
- Scheduling
- Setup time

## TYPES OF OS



## \* Simple Batch System:

- User submit job to computer operator
- Use of Processor point of view
- Monitor's point of view

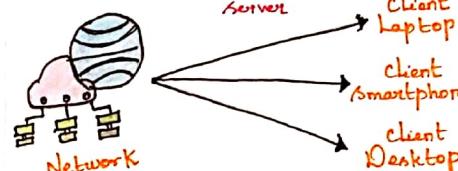
## \* Mainframe System

Adv. → Very fast, Reliability  
Disadv. → Expensive, Cannot solve complex problem



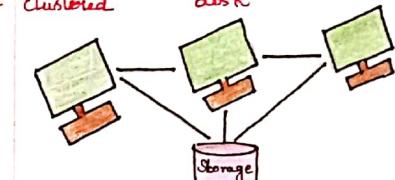
## \* Desktop System

Adv. → Control program operate user machine  
Disadv. → Centralization, Network congestion



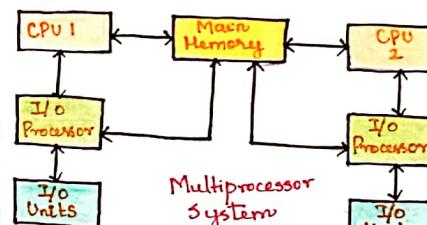
## \* Clustered Systems

Multiple CPUs clustered Share high performance disk Common storage



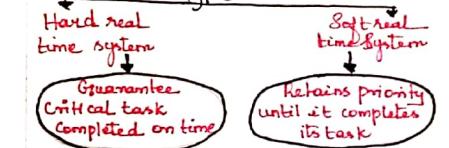
## \* Multiprocessing Operating System

More than one CPU used within 1 computer Adv. → Increase reliability, throughput  
Disadv. → More complex & sophisticated



## \* Real Time System

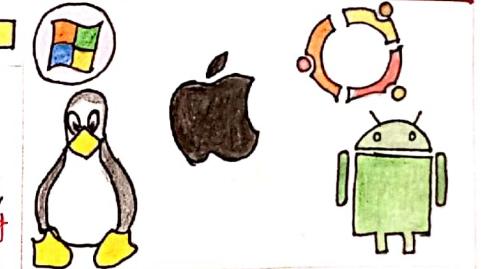
Time taken by the system to respond to an I/P



## \* Handheld Systems

Small amount of memory (PDA) Slow processor do not use virtual memory technique

## APPLICATIONS



## TOPIC 2: US DRUG JURISDICTION AND CRIMINAL

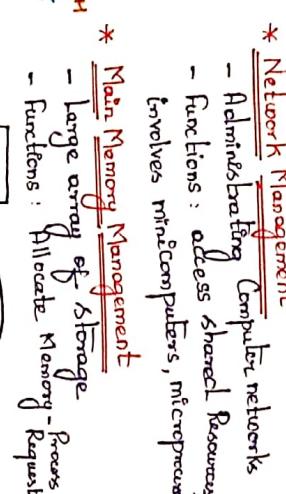
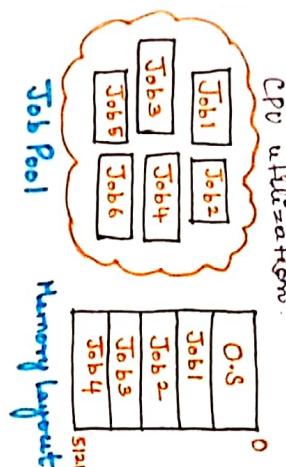
DS STIRTUBE

- \* Very generous in their makeup

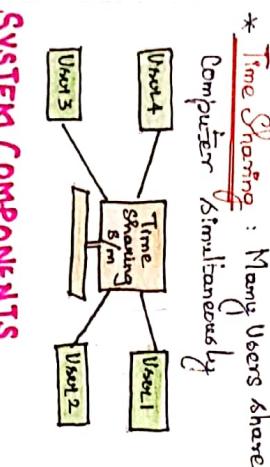
  - 1. process
  - 2. Suspense

- \* Internally  
Common Features [ Multiprogramming  
Time Sharing ]

- Multiprogramming : increases



- \* Network Management
    - Administering Computer networks
    - Functions: access shared Resources
    - involves minicomputers, microprocessors
  - Main Memory Management
    - Large array of Storage
    - Functions: Allocate Memory - Request



- \* Secondary Storage Management**

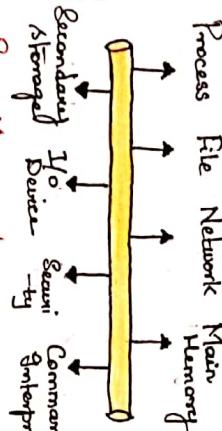
```

graph LR
    CPU[CPU] -- "Data" --> MMU[MMU]
    MMU -- "PA" --> HMMU[HMMU]
    HMMU -- "H-H" --> HHDisk[H-H]
    HHDisk -- "SUSPENDED STATE" --> HHDisk
    
```

The diagram illustrates the flow of data from the CPU through memory management units (MMU and HMMU) to the Hard Disk (H-H). The Hard Disk is shown in a state where it is suspended, indicated by a box labeled "SUSPENDED STATE" overlaid on the disk icon.

## System Components

- \* Large & Complex system partitioned into small parts



- \* Process Management

- Manages many processes simultaneously.
  - Execution is sequential
  - Example: Search Engine like Chrome.

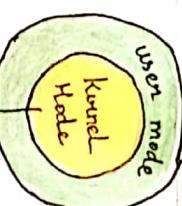
OS SERVICES

- Functions :
    - process creation and deletion
    - process synchronization

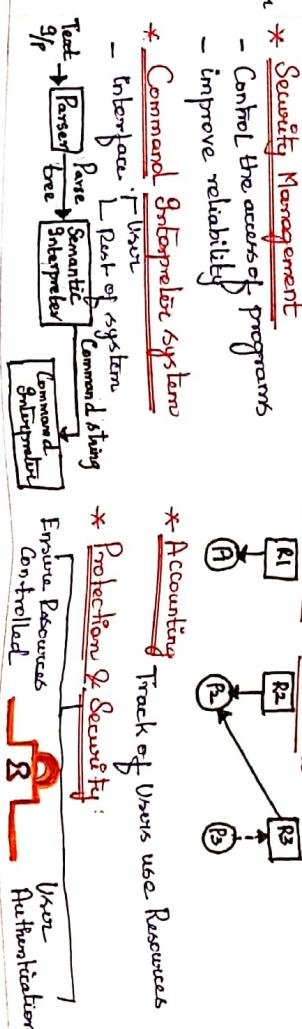


## SYSTEM CALLS

- Services available



<u>File</u>	<u>Management</u>	<u>Status</u>	<u>Programming</u>
	Create	Date	Language
	Delete	Time	Compile
	Copy	Amount of	Assembler
	Rename	Memory space	Debugger
Print	No. of lines	Debugging	Subroutines
Dump	Listing		
List	Debugging		



G

controlled

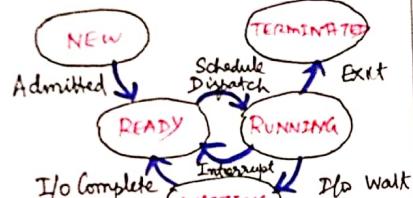
- ## Authentication



## TOPIC 4: PROCESS and THREADS

- PROCESS:
- Program in execution
  - Need resources
  - change state during execution

### PROCESS STATES:



### PROCESS CONTROL BLOCK (PCB)

- Process representation in operating system
- It contains
  - Process State
  - Process Number
  - Program Counter
  - Registers
  - List of open files
- Information associated with the process

### PROCESS SCHEDULING:-

- Maximize CPU Utilization
- Switch CPU among processes
- Here some process running at all times

### SCHEDULING QUEUES:-

- Process residing in memory waiting to execute
- List of processes waiting for an I/O device.

### SCHEDULED:-

- Job Scheduler
- Select processes and loads into memory for execution
- CPU Scheduler
- Allocate CPU to one of the Selected Processes
- To reduce overhead of multiprogramming

### PROCESS CLASSIFICATION

Two Types

- I/O BOUND**
- CPU BOUND**

I/O BOUND

- Spends most of the time doing I/O requests infrequently

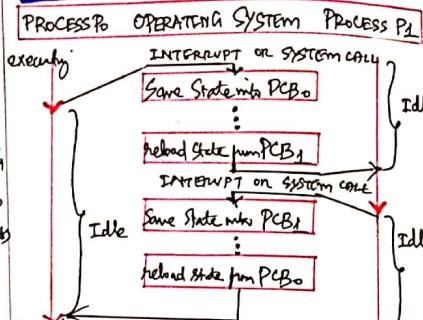
CPU BOUND

- Generates I/O requests frequently

CONTEXT SWITCH:-

- Save the current context of process
- Load the context of new process in the PCB.

### DIAGRAMS:



### OPERATIONS ON PROCESSES:-

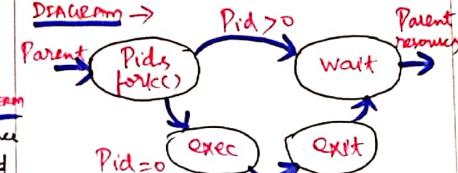
#### PROCESS CREATION

- Create Parent and Child Processes.
- Process identifiers assigned to each process

#### CHILD:-

- Can be a duplicate of Parent or can have a new program

#### DIAGRAM →



#### PROCESS TERMINATION:-

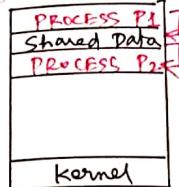
- Can be terminated if
  - \* Child exceeds resources
  - \* Child task not needed
  - \* Parent is exiting

### COOPERATING PROCESSES:

- It can affect or be affected by other processes running on the system
- Cooperating processes may share data with each other

#### Advantages:

- \* Information sharing
- \* Computation Speed up
- \* Modularity
- \* Convenience.



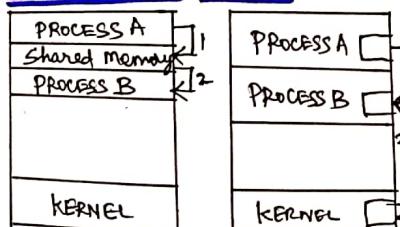
### INTERPROCESS COMMUNICATION (IPC)

- Communicating with the process (Shared / not Shared)

#### MODELS OF IPC:-

- \* Shared Memory
- \* Message Passing
  - Allow process to exchange information.

### COMMUNICATION MODELS:-



#### SHARED MEMORY

- Memory shared by cooperating process
- Information read/write, from/to the shared region
- to avoid redundant copies

### TYPES OF BUFFER

BOUNDED BUFFER

UNBOUNDED BUFFER

- \* Fixed Buffer Size
- \* No practical limit

### MESSAGE PASSING SYSTEMS:

- \* Primitives
  - Send (message)
  - Receive (message)

### METHODS OF IMPLEMENTATION:

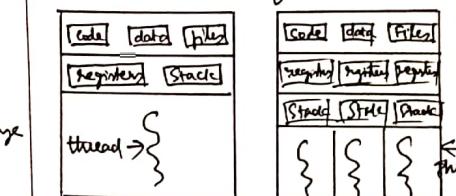
- \* Direct / Indirect Communication
- \* Synchronous / Asynchronous Communication
- \* Automatic / explicit buffering.

### THREAD: (LIGHT WEIGHT PROCESS)

- executing program into a single thread of control
- \* Threads
  - Use threads
  - kernel threads

### MULTITHREADING:-

- executing multiple threads simultaneously.



### \* Single & multithreaded Processes

- \* Responsiveness
- \* Resource sharing
- \* Economy
- \* Utilization of Multithreaded MC

### MULTITHREADING MODELS:-

- \* One-to-one
- \* Many-to-one
- \* Many-to-many

### REAL TIME APPLICATION:-

- Multithreaded
  - Used in the web browser
  - Used in the games
  - AUDIO
  - TEXT
  - VIDEO

## TOPIC 5 : PROCESS SCHEDULING

### CPU SCHEDULING:

- Basis of multiprogrammed OS.
- Switching the CPU among processes

### OBJECTIVE OF MULTIPROGRAMMING

- Maximizing CPU Utilization

### SCHEDULING CRITERIA

MAXIMIZE

MINIMIZE

#### \* CPU UTILIZATION

- CPU remains as busy as possible
- 100% utilization

#### \* THROUGHPUT

- no. of processes that finish their execution per unit time

### CPU SCHEDULING → 4 Circumstances

Preemptive

Non-preemptive

- ① Process switches from running state to ready state
- ② Process terminates or completes
- ③ Process switches from waiting state to ready state

- ④ Process switches from running state to waiting state

- ⑤ A process terminates or completes

FCFS -

First Come  
First Served

CPU  
SCHEDULING  
ALGORITHMS

SJF -  
Shortest Job  
First

Priority  
Scheduling

Round Robin  
Scheduling

### (i) FCFS → First Come First Served

- Easiest & most simple CPU Scheduling algorithm
- Process requests the CPU gets the CPU allocation first.
- It can be managed with a FIFO Queue.
- It is easy to implement & use

#### Example:-

PROCESS	BURST TIME OR EXECUTION TIME	PRIORITY
P <sub>1</sub>	24	2
P <sub>2</sub>	3	1
P <sub>3</sub>	3	3

#### Grant chart:-

	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	
0		24	27	30

- (i) Waiting time of P<sub>1</sub> = 0
- (ii) Waiting time of P<sub>2</sub> = 24 AWT = (0+24+27)/3 = 51/3 = 17 ms
- (iii) Waiting time of P<sub>3</sub> = 27 TAT (Turn around time) = BT + WT
- (iv) TAT of P<sub>1</sub> = 24+0=24
- (v) TAT of P<sub>2</sub> = 3+24=27
- (vi) TAT of P<sub>3</sub> = 3+27=30
- ATAT = (24+27+30)/3 = 84/3 = 28 ms

### (ii) SJF → Shortest Job First

- Process with shortest execution time.
- Reduces average waiting time
- Shortest job executed first and shortest turn around time.

#### Grant chart:-

	P <sub>2</sub>	P <sub>3</sub>	P <sub>1</sub>
0	3	6	

- (i) WT of P<sub>1</sub> = 6
- (ii) WT of P<sub>2</sub> = 0 AWT = (6+0+3)/3 = 9/3 = 3 ms
- (iii) WT of P<sub>3</sub> = 3 = 3 ms
- (iv) TAT of P<sub>1</sub> = 24+6=30
- (v) TAT of P<sub>2</sub> = 3+0 = 3
- (vi) TAT of P<sub>3</sub> = 3+3 = 6
- ATAT = (30+3+6)/3 = 39/3 = 13 ms

### (iii) PRIORITY SCHEDULING

- Scheduling process based on Priority

- Scheduler selects the tasks as per the priority
- equal priority carried out on a FCFS or Round Robin basis.

#### Example:-

PROCESS	BURST TIME	PRIORITY
P <sub>1</sub>	24	2
P <sub>2</sub>	3	1
P <sub>3</sub>	3	3

#### Grant chart:-

	P <sub>2</sub>	P <sub>1</sub>	P <sub>3</sub>
0	3		27 30

- (i) WT of P<sub>1</sub> = 3
- (ii) WT of P<sub>2</sub> = 0 AWT = (3+0+27) = 30/3 = 10 ms
- (iii) WT of P<sub>3</sub> = 27 = 10 ms
- (iv) TAT of P<sub>1</sub> = 24+3=27
- (v) TAT of P<sub>2</sub> = 3+0=3
- (vi) TAT of P<sub>3</sub> = 3+27=30
- ATAT = (27+3+30)/3 = 60/3 = 20 ms

### (iv) ROUND ROBIN ALGORITHM:-

- Algorithm comes from Round Robin
- mostly used for Scheduling in multitasking
- It is a hybrid model which is cyclic - driven
- Time Slice should be minimum assigned for Sparte task to be processed.

TIME SLICE = 5

#### Grant chart:-

	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>1</sub>	P <sub>1</sub>	P <sub>1</sub>	
0	5	8	11	16	21	26	30

- (i) WT of P<sub>1</sub> = 6
- (ii) WT of P<sub>2</sub> = 5 AWT = (6+5+8)/3 = 19/3 = 6.3 ms
- (iii) WT of P<sub>3</sub> = 8 = 6.3 ms
- (iv) TAT of P<sub>1</sub> = 24+6=30
- (v) TAT of P<sub>2</sub> = 3+5=8
- (vi) TAT of P<sub>3</sub> = 3+8=11
- ATAT =  $\frac{30+8+11}{3} = \frac{49}{3} = 16.3 \text{ ms}$

### MULTI-PROCESSOR SCHEDULING

#### OR MULTI-PROCESSOR SCHEDULING

- \* consists of more than one processor
- \* multiple CPUs share the load so that various processes run simultaneously.
- \* In multiprocessor scheduling, many processes are identical and sharing process at any time.

### REAL TIME SCHEDULING:-

- \* Pt carry real time tasks
- \* Tasks performed immediately with a certain degree of urgency
- \* classified into Hard Real time tasks & Soft Real time tasks

### ALGORITHM EVALUATION:-

- To compare all four scheduling algorithms, SJF → Shortest Job First scheduling algorithm produces best results.

### PROCESS SCHEDULING MODELS

- Process scheduling in the Solaris 2, windows 2000 and Linux OS.
- Introduce threads to the process model
- It allows single process to multiple threads of control
- classified into User level threads & Kernel level threads (unaware of the other)
- Thread library schedules user level threads to run on an available LWP. a scheme is known as Process Local Scheduling

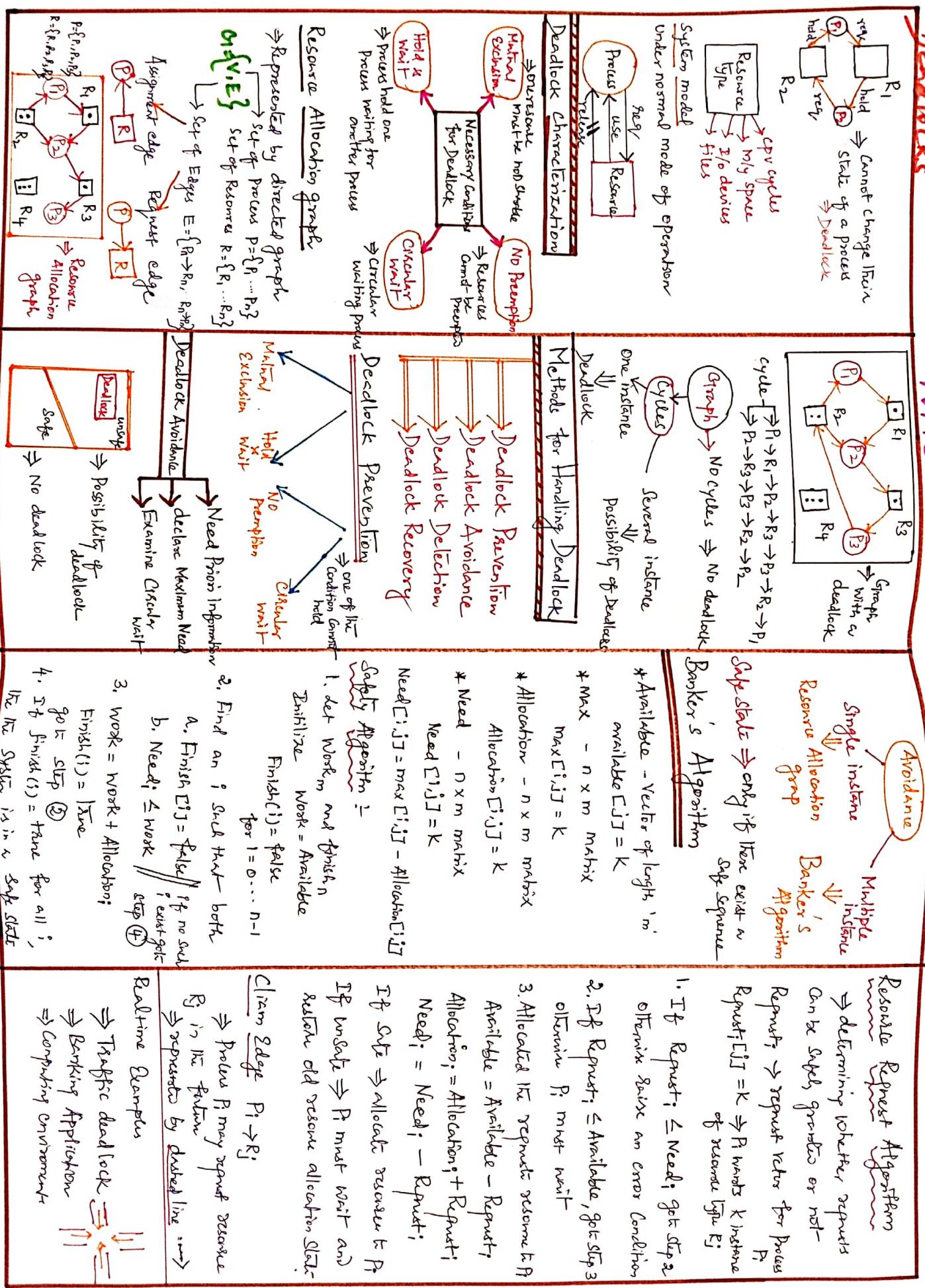
### REAL TIME APPLICATION:-

- buying a Train ticket, movie ticket on the ticket counter. (FCFS)
- Simple educational web games
- Elimination of teams in the tournament (Round Robin)



# Deadlocks

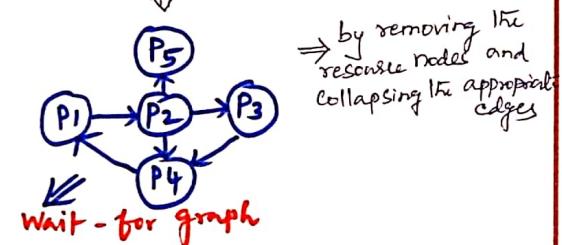
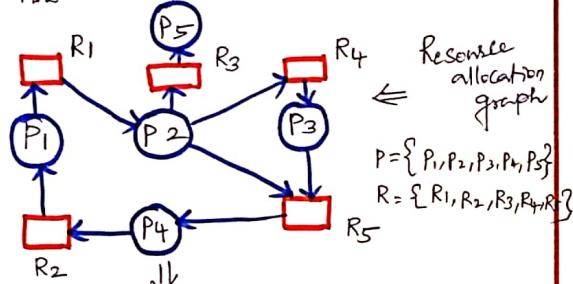
## Topic 7



## TOPIC 8

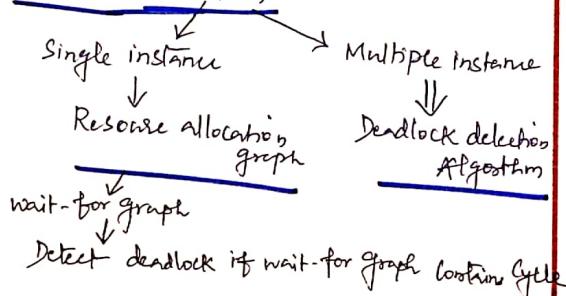
### Deadlock Detection

→ An algorithm that examines the state of the system to determine whether a deadlock has occurred or not



- nodes are processes
- $P_i \rightarrow P_j$  if  $P_i$  is waiting for  $P_j$
- deadlock exists in the system if and only if the wait-for graph contains a cycle
- periodically invoke an algorithm to find the cycle in the graph

### Deadlock detection



### Deadlock detection

- \* Available - Vector of length 'm'  
 $\text{available}[i, j] = K$
- \* Allocation -  $n \times m$  matrix  
 $\text{Allocation}[i, j] = K$
- \* Request -  $n \times m$  matrix  
 $\text{Request}[i, j] = K$

1. Initialize

- i)  $\text{work} = \text{Available}$
- ii)  $\text{for } i = 0, \dots, n-1$   
if  $\text{Allocation}[i, :] \neq 0$   
 $\text{finish}[i, :] = \text{false}$   
otherwise  
 $\text{finish}[i, :] = \text{true}$

2. Find an  $i$  such that

- a)  $\text{finish}[i, :] = \text{false}$
- b)  $\text{Request}[i, :] \leq \text{work}$

If no such  $i$  exist, go to step 4.

3.  $\text{work} = \text{work} + \text{Allocation}[i, :]$   
 $\text{Finish}[i, :] = \text{true}$

Go to step 2.

4. If  $\text{finish}[i, :] = \text{false}$  for some  $i, 0 \leq i < n$   
then the system is in a deadlocked state  
if  $\text{finish}[i, :] = \text{false} \rightarrow P_i$  is deadlock

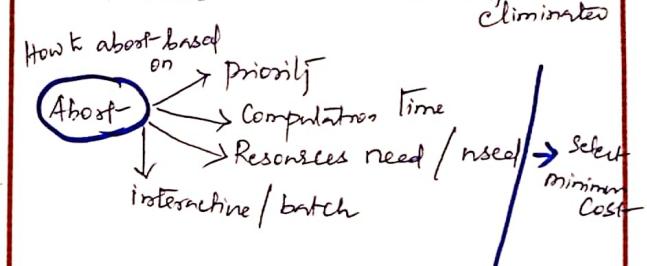
### Recovery from Deadlock

→ Abort one or more processes to break the circular wait (Process termination)

→ Preempt some resources from one or more deadlock processes (Resource preemption)

### Process termination

→ Abort all deadlocked processes



### Resource Preemption

→ eliminate deadlocks using resource preemption

→ Selecting a victim, which processes are to be preempted?

→ Rollback - if preempt a resource, what should be done with that process?

→ Starvation - How do ensure that starvation will not occur?

→ Selecting a victim → determine the order of preemption to minimize cost

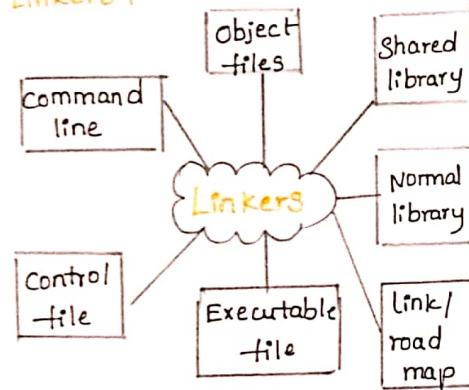
→ Rollback → roll back the process to safe state and restart from that state

→ Starvation → can be picked as a victim only finite no. of times

→ using resource preemption, successively preempt some resources from process → using these resources to other processes until the deadlock cycle is broken

# TOPIC 9 : DYNAMIC STORAGE MANAGEMENT

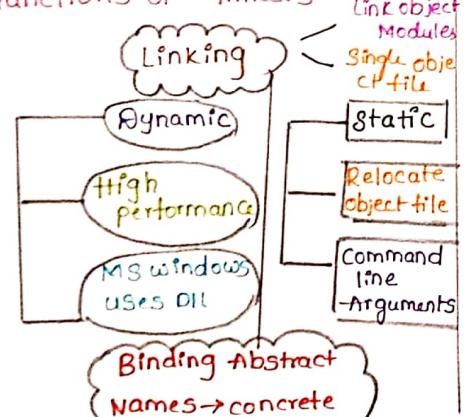
**Linkers :-**



Linkers → program in system → program into single file

- Tasks one or more object files.
- combine these files into executable files.
- output complete self sufficient

**functions of linkers**



- collect all pieces of program
- memory organization to first pieces.
- Get more memory to run program.

**Dynamic linking :-**

- way to implement Dynamic linking - Jump table.
- contains small function called when program items.
- link library determines dynamic

**Advantages :-**

- memory requirements reduced
- All loaded into memory once
- more than one application use signal

**Dynamic Storage Management:-**



1. Allocate a given no. of bytes

2. Free a previously allocated block

**Status Allocation :-**

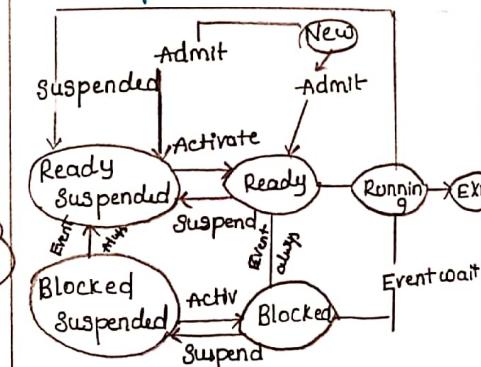
- used when memory association and freezing are partially predictable

- status based organization keeps free space in one place

**Heap Allocation :-**

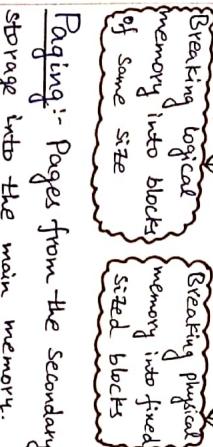
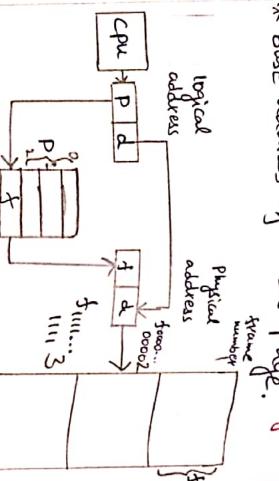
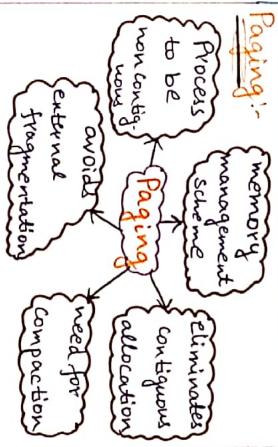
- used when allocation and release are unpredictable.
- Memory contains free areas & allocated areas
- uses free list to keep track of storage.

**Swapping :-**

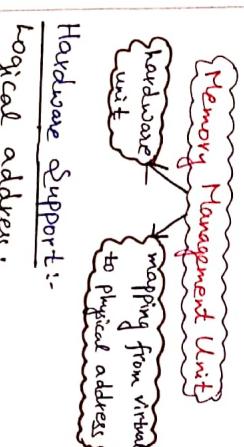
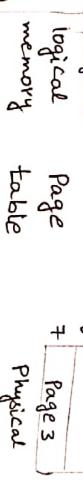


## Topic 10 : Paging

\* Base address of each page.



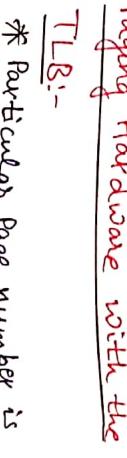
**Paging:-** Pages from the secondary storage into the main memory.



Page number (P)	Page offset (d)
10	10

**Page Table:**

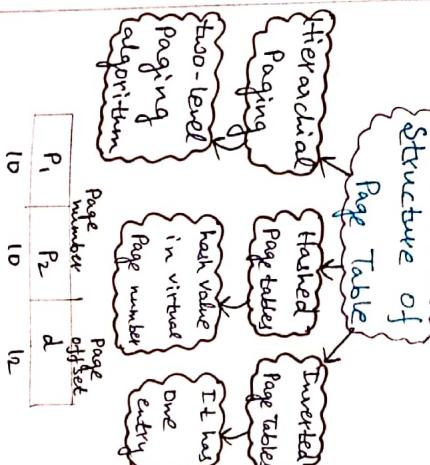
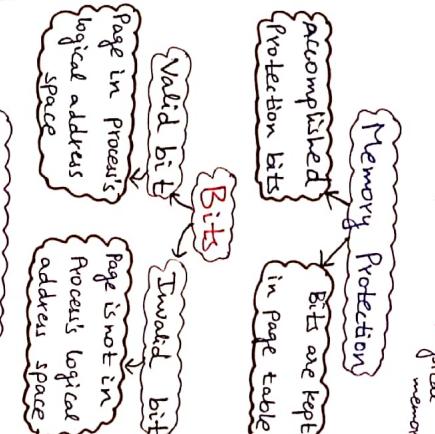
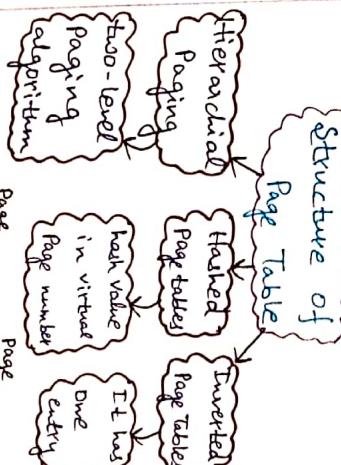
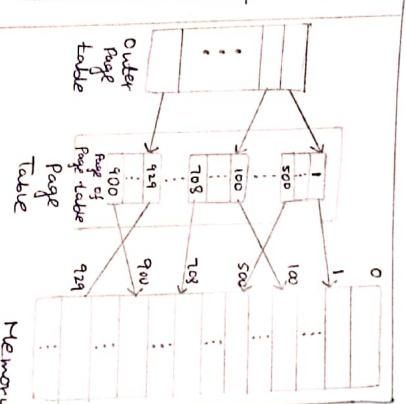
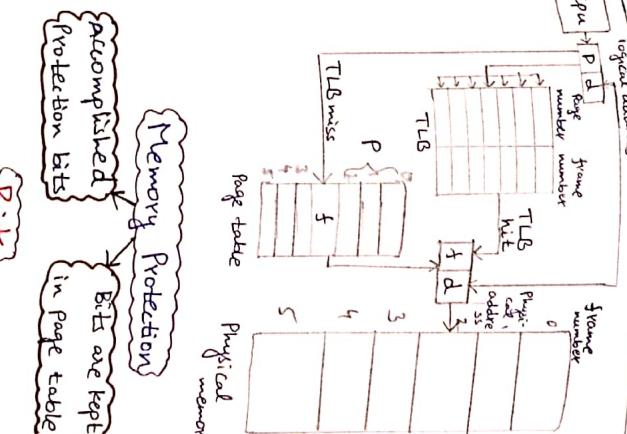
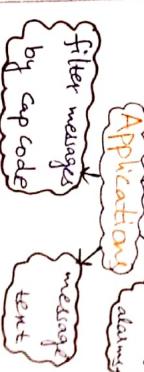
\* Page number used as index



Page number	Page offset
P <sub>1</sub>	d

\* Particular Page number is found in the TLB

\* Translation works from the outer page table



## TOPIC 11:- Segmentation

### Segmentation :-

**Segmentation :-**

memory management scheme

Segment :- Breaking logical memory into blocks of the different size.

**Segment :-** Breaking logical memory into blocks of the different size.

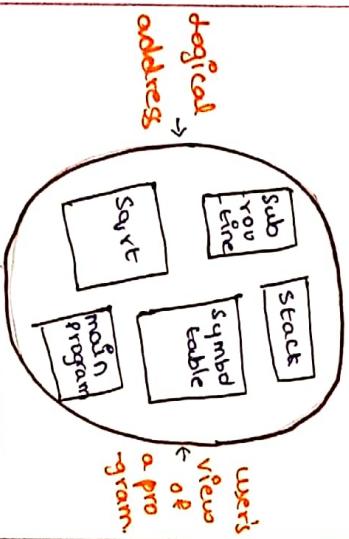
logical address → segment

logical address → segment

logical address

segment

length



logical address

segment

length

consists of a two tuple

segment-number, offset.

segment - number

offset.

segment - number, offset.

offset.

Hard ware

physical memory

sequence of bytes

Segment - dimensional address

physical memory

sequence of bytes

Segment - dimensional address

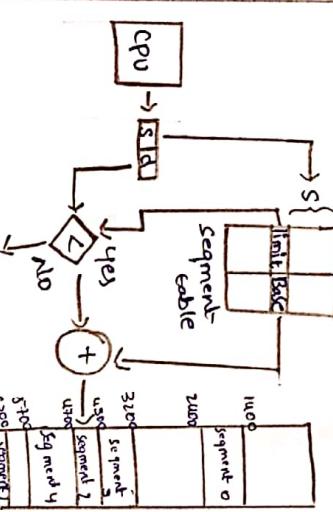
physical memory

sequence of bytes

**Base Address** :- starting physical address where the segments reside in memory

**Limit** :- specifies the length of the segment.

**Physical address** → segment base + segment offset in memory



trap: addressing physical memory.

**Segmentation hardware**

**Segment - number** :- No. of segments required to represent the segment.

**Segment set (S)** :- No. of bits required to represent the size of the segment.

**Advantages of Segmentation**

- No internal fragmentation
- Less space consumed in comparison to page table

**Disadvantages of Segmentation**

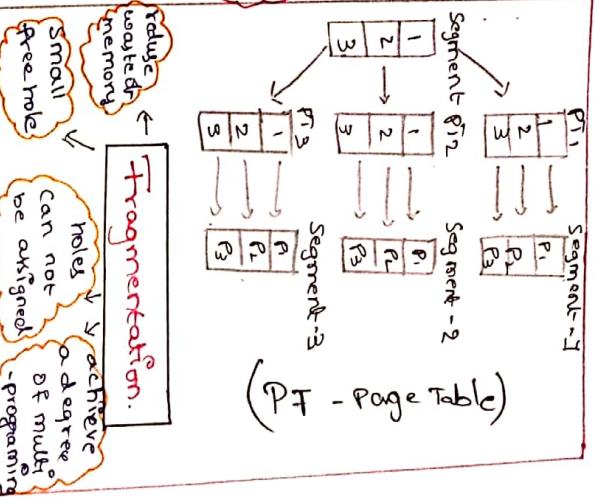
- External fragmentation
- Process are loaded & re-moved from the memory

**Segmentation with Paging**

- Main memory is divided into variable size segments
- Segments are further divided
- Fixed sized

**Segmentation**

- Segment No
- Page No
- Paging
- Segment Number
- appropriate segment number within the segment
- Used as different set with the pages
- exactly same page within the segment



**Fragmentation**

- internal fragmentation
- external fragmentation

**Applications:**

- The code
- C-compiler
- Spacers used by each thread
- Global variables
- Heap
- C library

## TOPIC 12: VIRTUAL MEMORY

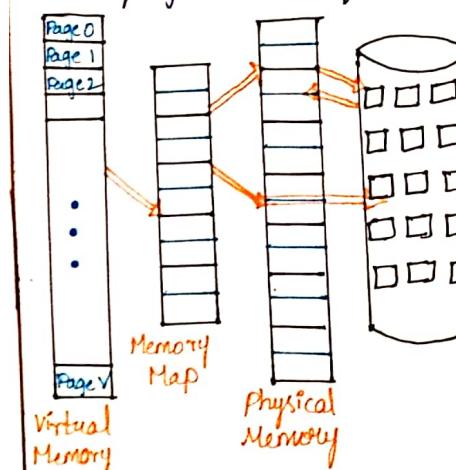
### Virtual Memory :-

#### Background :

- \* Error Code, Unusual routines, Large data structures.
- \* Entire program code not needed at same time.
- \* Ability to execute partially-loaded program.
- \* less input/output needed to load or swap programs.

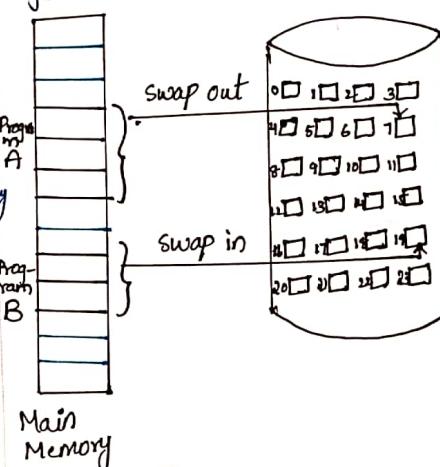
Virtual Memory :- Separation of user logical memory from physical memory.

- \* Only part of the program needs to be in memory for execution.
- \* More efficient process creation.
- \* More program running concurrently



### Demand Paging :-

- \* Process of loading pages.
  - \* Pages are demand loaded.
  - \* It works by Virtual Memory.
  - \* Primary (RAM) and Secondary (HD) storages.
  - \* Pages loads on demand.
  - \* Process removed and loaded.
  - \* Valid and Invalid Pages.
- **dogy Swapper** - never swaps a page into memory unless page will be needed.



### Page Fault :-

- \* Needed / Required file is not in primary memory.
- \* Secondary memory, no primary memory.
- \* to retrieve the file from the

memory.

### Page Replacement Algorithm :-

- \* Decided which memory page be two operations.
- \* 'swap in' & 'swap out'
- \* Page Hit → Page Fault.

### Types of Algorithms :-

- \* FIFO
- \* Optimal Page Replacement
- \* LRU Page Replacement
- \* LRU Approximation

#### → FIFO [First In First Out]

- \* Replace the longest time

Ex:- 701 2030 → Pages

7	7	7	2	2	2	2	*	Page-Fault
0	0	0	0	3	3	3	*	Page-Hit
1	1	1	1	1	1	0	*	*
*	*	*	*	*	*	*	*	*

#### → LRU :- [Least Recently Used]

7	7	7	2	2	2	2	*	Page-Fault
0	0	0	0	0	0	0	*	*
1	1	1	3	3	3	3	*	*
*	*	*	*	*	*	*	*	*

#### → optimal : [Later Case]

Ex:- 701 20307 → Pages

7	7	7	7	7	7	7	7	*
0	0	0	0	0	0	0	0	*
1	2	3	3	3	3	3	3	*
*	*	*	*	*	*	*	*	*

### LRU Approximation :- Systems

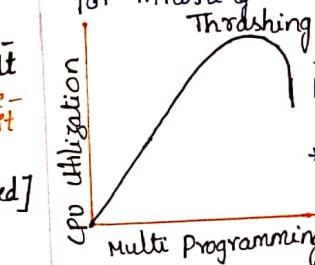
Provide sufficient hardware support for true LRU page replacement.

### Additional Reference Bits Algorithm :-

- recordings the reference bits at regular intervals.

### THRASHING & Working Set :-

- \* Spending most of time in memory.
- \* Page faults may happen.
- \* Service performance issues.
- \* Due to recovery it delays in execution.
- \* Impact of OS execution performance.
- \* CPU usage becomes low.
- \* High level multi programming
- & lack of frames.
- \* Above two are the reason for thrashing.



- \* Thrash is the poor performance of virtual memory.
- \* virtual memory are overloaded.

### DISK Devices :-

- \* Magnetic Disk - Data on magnetized media.
- \* RAID - Redundant array of independent Disks.
- \* CDROM - Compact disk Read only Memory.
- \* HDD - Hard disk drive.
- \* SSD - Solid State drive.

### APPLICATION :-



## TOPIC 13

### FILE SYSTEMS

#### File Concepts

Files - Named collection of related information  
 file → Program → source form  
 → data Object  
 File → Text file form  
 → source file executable file  
**file Attributes**  
 Name / Identifier Type Location  
 Size Protection

#### File Operations

Creation Writing → Reading  
 Truncating → Repositioning  
 Deleting

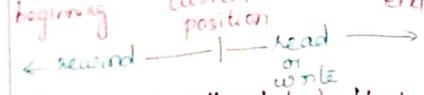
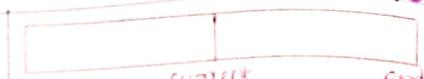
**Information of a file**  
 File pointer file open count Disk location Access rights

#### File Types (Extensions)

- executable → .exe, .com, .bin
- object → .obj
- source code → .c, .cc, .java
- batch → .bat, .sh
- markup → .xml, .html
- word processor → .xml, .docx
- library → .lib, .dll
- print/view → .pdf, .jpg
- archive → .tar, .zip
- multimedia → .mp3, .avi

#### File Structures

- Internal file structure
- Logical records vary in length
- Packing several logical records to physical blocks - solution



- Disk space allocated in blocks
- Internal fragmentation problem
- Larger block size - greater internal fragmentation.

#### File Access Methods

Contiguous Direct Indexed Sequential

#### Sequential access: (Contiguous)

- Information processed in order
- Operations → read, write
- Based on tape model of a file
- `read_next()` → read next portion
- `write_next()` → append to a file

#### Direct access:

- Relative access.
- File → fixed length logical records
- Read / write in no particular order
- Based on disk model of a file
- File → numbered sequence of blocks or records.
- Immediate access to large information.
- Example → Databases → airline reservation
- Operations → `read_next()`, `write_next()`
- Relative block number
- `read(n)`, `write(n)`

#### Indexed access:

- Index - Pointers to various blocks
- Search the index
- Access file directly.
- Disadvantage - index file too large
- Solution - create index for index file.

Sequential access	Implementation for direct access
read	$cp = 0$
read_next	$read cp; cp = cp + 1$
write_next	$write cp; cp = cp + 1$

#### File System Interface

- Most visible aspect of OS.
- Online storage and access to data and programs.
- File Systems

#### Directory structure

Collection of files → store related data → organize provide information

#### File protection

- access and control by multiple users.

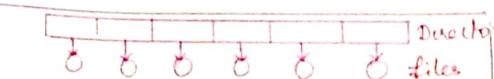
#### Directory structure

- Defining the logical structure of a directory.

- Types
  - Single level
  - Two level
  - Tree Structured
  - Acyclic Graph Directory

#### Single level directory:

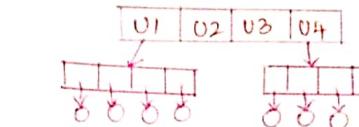
- Simplest directory structure
- All files in same directory



limitation - when number of files increase.

#### Two level Directory

- Separate directory for each user levels
- User file directory - Each user
- Master file directory - overall system



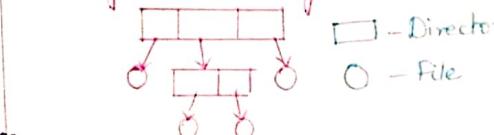
Limitation - Isolates users

- Difficult to cooperate

#### Tree Structured Directory

- Extend directory structure to a tree of arbitrary height.

Directories - Subdirectory - files



- Tree - Root directory

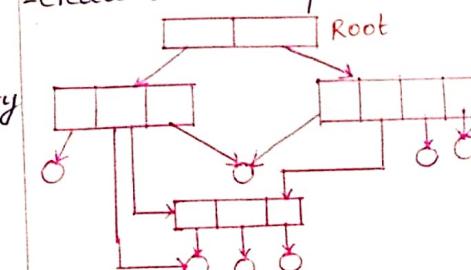
- File - Unique path name.

#### Acyclic Graph Directory

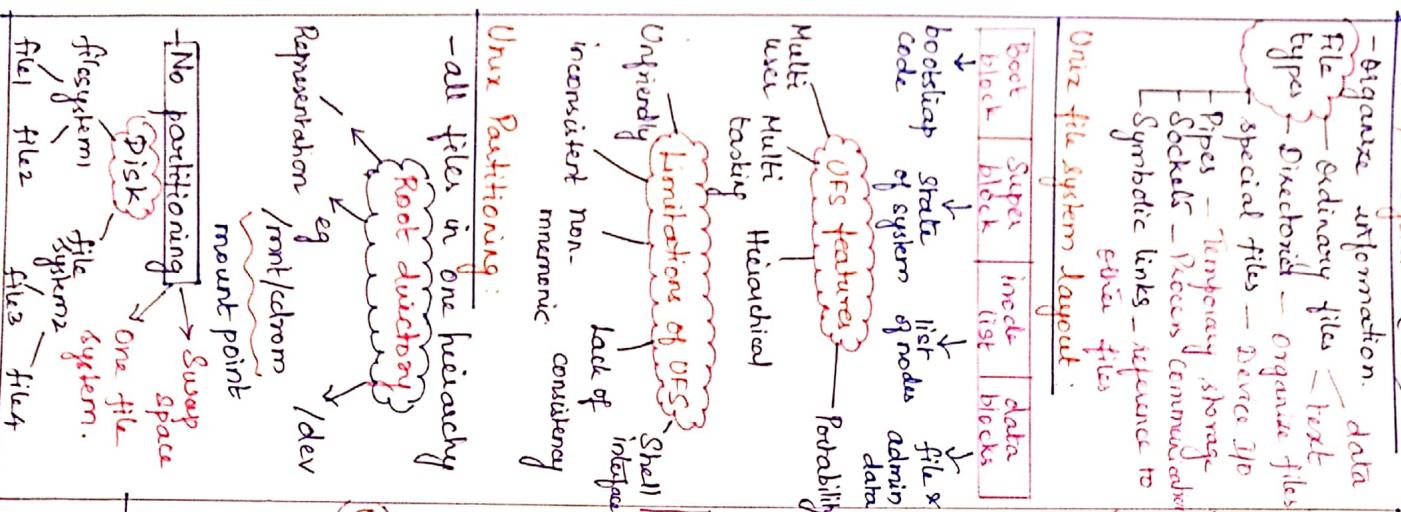
- Acyclic graph - Graph with no cycles.

- Sharing of subdirectories and files

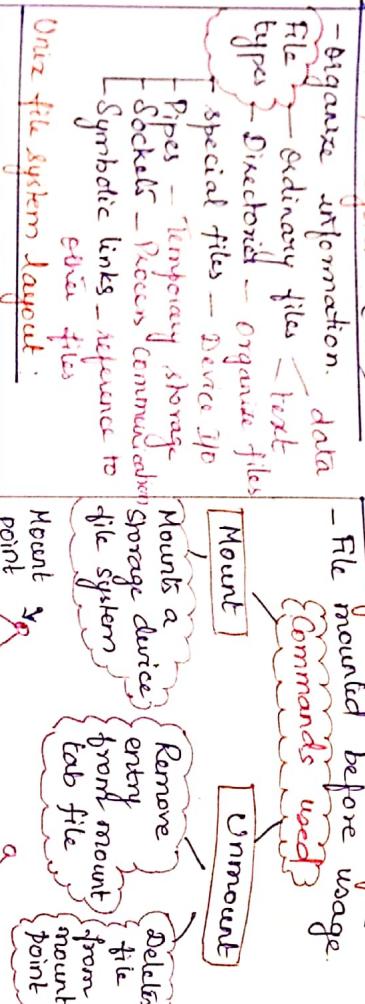
- Create a link to point to files



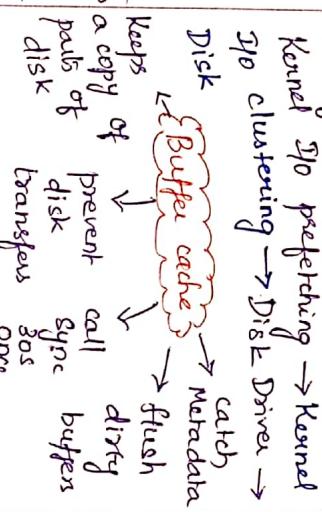
## UNIX file system (CDFS)



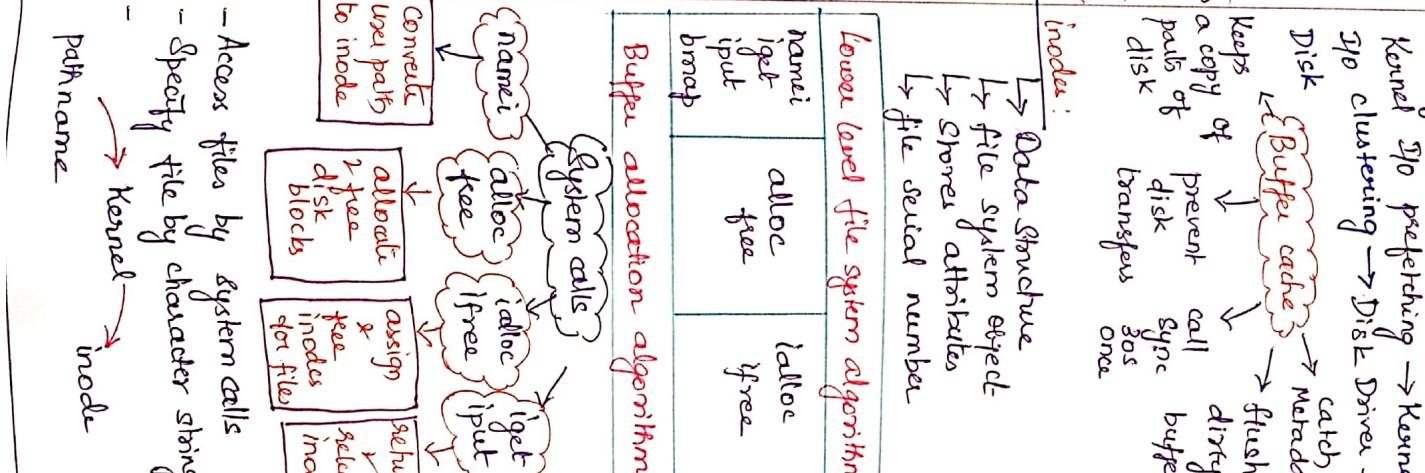
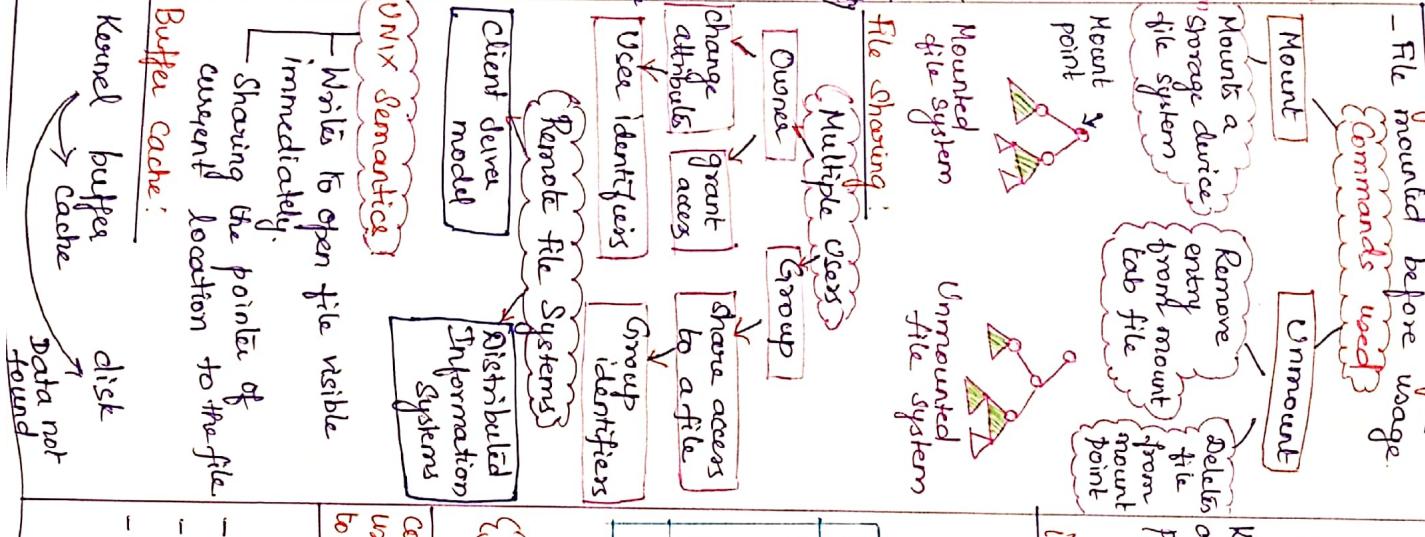
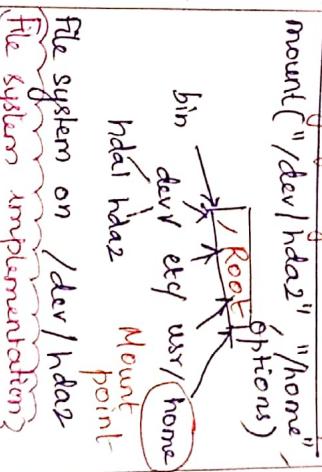
## Mounting and Unmounting system



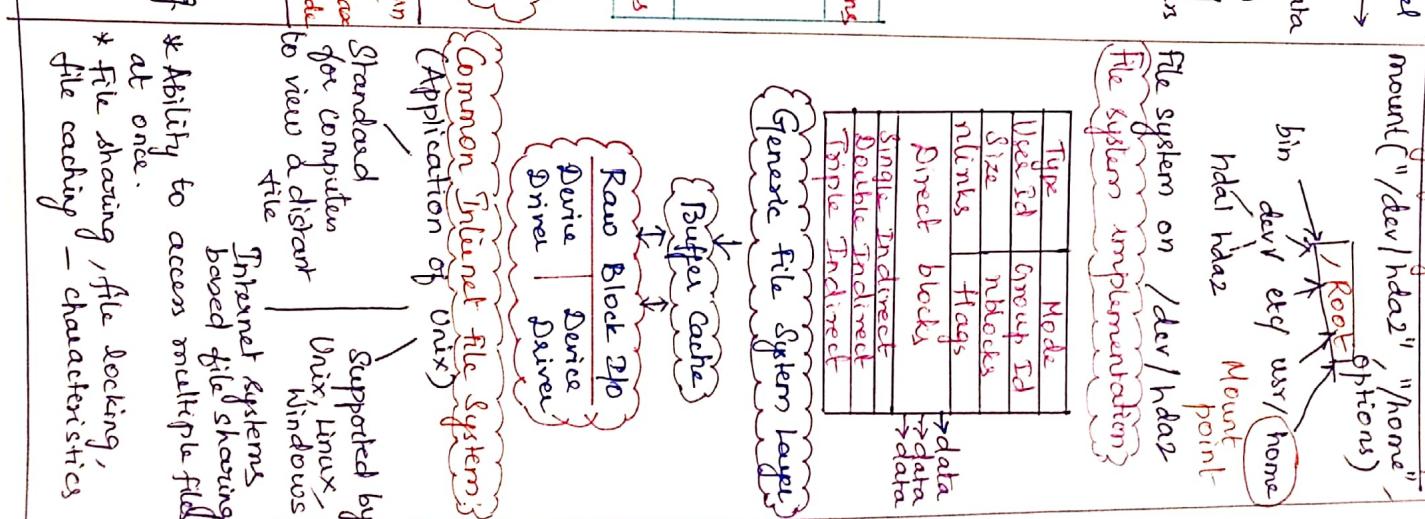
## file system → Buffer cache →



## Mounting file systems - Example



## File system - Example

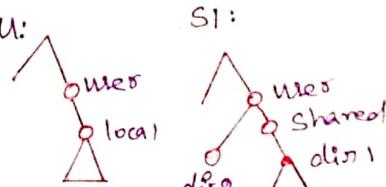


## TOPIC 15: Network File Systems, I/O Systems

### NFS:

- \* Share files between users on different distributed file system

U:



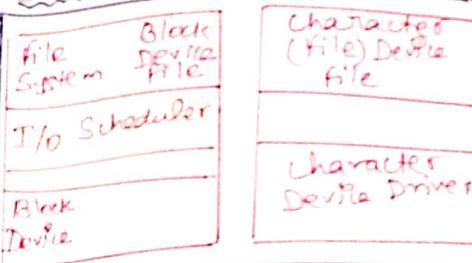
S1:

Application I/O Interface:  
\* Interface between application programs and I/O devices.

Abstraction Transmutation I/O Layering

\* Hide the differences among device controller  $\rightarrow$  I/O.

Device Block Structure:



Streams:-

User Process

Stream Head

Read queue Write queue

Read queue Write queue

Read queue Write queue

Read queue Write queue

Driver End.

Device

Stream Structure

Character and Block Device

Switch tables:

Arrays of operational pointers.

Sub array of jump pointers.

Real Time Implementation

Hadoop API  $\rightarrow$  Map Reduce Hbase

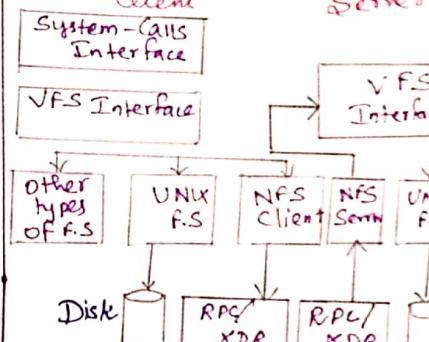
file Based Appln.

Supported by most os.

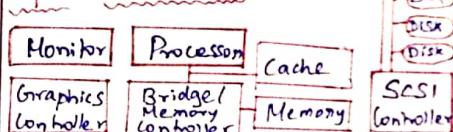
SQL Based Tool

BI Appln & Query Builder.

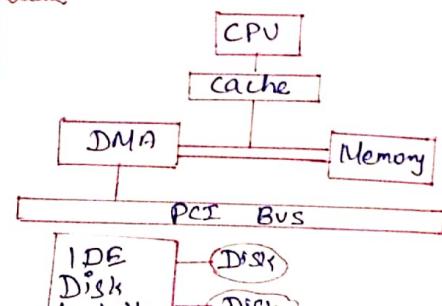
NFS Architecture:



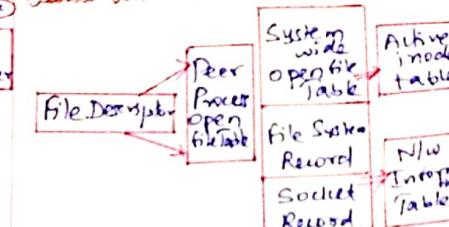
I/O Hardware:



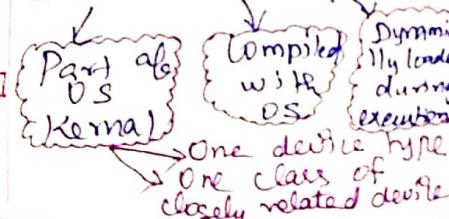
DMA:



UNIX I/O Kernel Structure:



Device Drivers:



Character Devices:-

Register set of functions

Implements drivers

Performs various

file I/O operations

Block and character Devices:-

Commands

Reader

Writer

Socket

Ex:

\* Keyboard  $\rightarrow$  functions

get() put()



## Topic 17: LINUX SYSTEM

### COMPONENTS OF LINUX SYSTEMS

System management processes	Kernel programs	Utility programs	Completion programs
<b>System related libraries.</b>			

**Loadable kernel modules**

Kernel: - Responsible for managing loadable kernel modules

System utilities :- Perform individual tasks.

System libraries :- Standard set of functions

Management tools :- Facilitated management tasks.

Odd version number - DEVELOPMENT

Even version number - PRODUCTION

KERNEL

Maintained by Linux to ensure compatibility

Standard set of packages

Red Hat, Debian, SUSE

Canonical

Linux distributions

General Public License (GPL) packages.

Installation, Management utilities, Ready to Install (RPM)

Kernel

Utilities, Ready to Install (RPM)

Red Hat, Debian

Ubuntu, SUSE

CentOS, Fedora, Red Hat

Debian, SUSE

### BLOCK DEVICES

Acks as pool of buffers for active I/O

Serves as a cache for completed I/O

THREE DEFINED CATEGORIES OF USERS :- OWNER, GROUP, OTHER

R = Read a file  
W = Write a file  
X = Execute a file

UNIX SYSTEM ADMINISTRATION

User, Groups & Access Permissions

Packet also does Firewall management filtering

Protocol - Represent an individual

File object - Represent an open file

Superblock object - Represent an entire file system entry

Device Drivers - Represent individual

Network protocols - Represent individual

Binary formats - Represent individual

Character device driver - Represent individual

Block (SCSI) device driver - Represent individual

I/O scheduler - Represent individual

TTY driver - Represent individual

Line discipline driver - Represent individual

Network device driver - Represent individual

### UNIX SYSTEM ADMINISTRATOR

\* Responsible for components

\* Maintain Develop, Support

\* Client Side and Server Side

REQUIREMENTS OF ADMINISTRATOR

Extensive knowledge in field of database

MySQL, Oracle, DB2 or ADABAS

Handle Mail Services & Extensive knowledge like Python, PHP or Ruby

Experience handling Mails

Experience dealing with Virtual Machines and containers

Refine Analytical and Structural way of thinking

Refine user / access rights & manage

Identify any problems & find solutions

Restore smooth operations

Functionality of Everyday Applications

Manages Network Servers

Implementation Hardware / Software Update

Provide Technical Assistance

Maintain System Security

Kernel: - Responsible for managing loadable kernel modules

System utilities :- Perform individual tasks.

System libraries :- Standard set of functions

Management tools :- Facilitated management tasks.

Odd version number - DEVELOPMENT

Even version number - PRODUCTION

KERNEL

Maintained by Linux to ensure compatibility

Standard set of packages

Red Hat, Debian, SUSE

Canonical

Linux distributions

CASE STUDY - LINUX SETUP

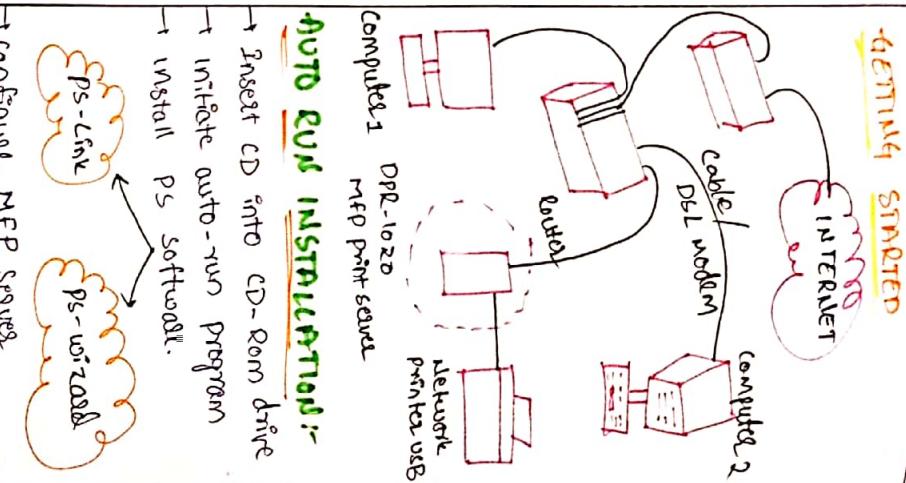
## TOPIC 11: SETTING UP LINUX MACHINES IN SEQUENTIAL

# SETTING UP A LINUX MULTIFUNCTION SERVER MATERIALS & SOFTWARE INSTALLATION

- printer to connect the DPR-1020 is operating correctly.
  - printer operating correctly.
  - switch its power off.
  - confirm your network operating normally.
  - use CAT-5 ethernet cable for network connection.
  - printer is turned off, connect the USB printer.
  - switch on the printer.
  - insert power adapter into the DC 5V power socket.
  - other power adapter into a power outlet.
  - supply power to print server blue LED on print server turned.
  - print sever's self-test will proceed.

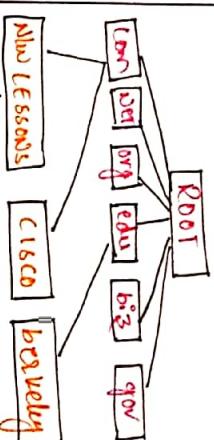
**Power on SELF-TEST**

  - DPR-1020 powered on, automatically perform self-test.
  - during the time of LED test may causes error.
  - immediately follow the KED tests.



AUTO RUN INSTALLATION

- use CAT-5 ethernet cable for network connection.
  - printer is turned off, connect the USB printer.
  - switch on the printer.
  - insert power adapter into power source socket.



BRILL SUMM. 2000

- supply power to print server
  - blue LED on print server
  - print servers self-test will proceed.

**Power on SELF-TEST**

  - DPR-1020 powered on, automatically perform self-test.
  - during the time of LED test may causes error.
  - immediately follow the KED tests.

Print / scan / copy / fax

- configure MFP server
  - change IP address
  - support the multi-function printer / scan / copy / fax
  - easily add a printer to your computer
  - view quick installation guide
  - view manual.



**Power on self-test**

→ DPR - load powered on, automatically perform self-test, → during the time of LED test may causes error.

- Power on SELF-TEST**

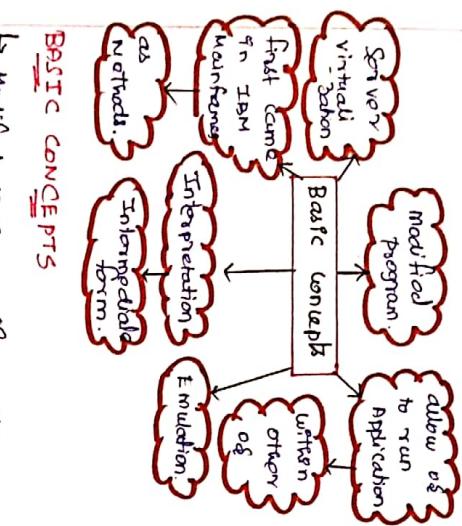
  - DPR-1020 powered on, automatically perform self-test, during the time of LED test may causes error.
  - Informed: only follow the LED Tests.

→ Host names into the corresponding numeric IP addresses.

- \* → Root zone of the DNS.
  - (TOP-LEVEL DOMAIN)
    - Highest level of domain names in the root zone of the DNS.
  - \* AUTHORITATIVE NAME SERVER:
    - provides actual answers to your DNS queries such as
    - \* mail server IP addresses
    - \* recursive - IP addresses
    - & CACHING NAME SERVER:
      - is a server handles requests from clients.
  - SETTING UP LOCAL NETWORK SERVICES
    - Configuring Network
      - setting up local network server
      - DHCP
        - setting up local network server
        - fast-config server
    - CONFIGURING DEBIAN
      - DHCP is used during the Debian installation, then configure the source with static IP addresses
      - o Static IP addresses
        - ex:- Debian
  - DHCP → Dynamic Host Configuration Protocol
    - it is used to solve a number of problems associated with local environment.

# VIRTUALIZATION

## VIRTUALIZATION



### BASIC CONCEPTS

- ↳ Modified program (form) than normal os.
- ↳ different user can operate its end two different environments.
- ↳ virtual hosting environments.
- ↳ separate applications into containers.

### SETTING UP XEN

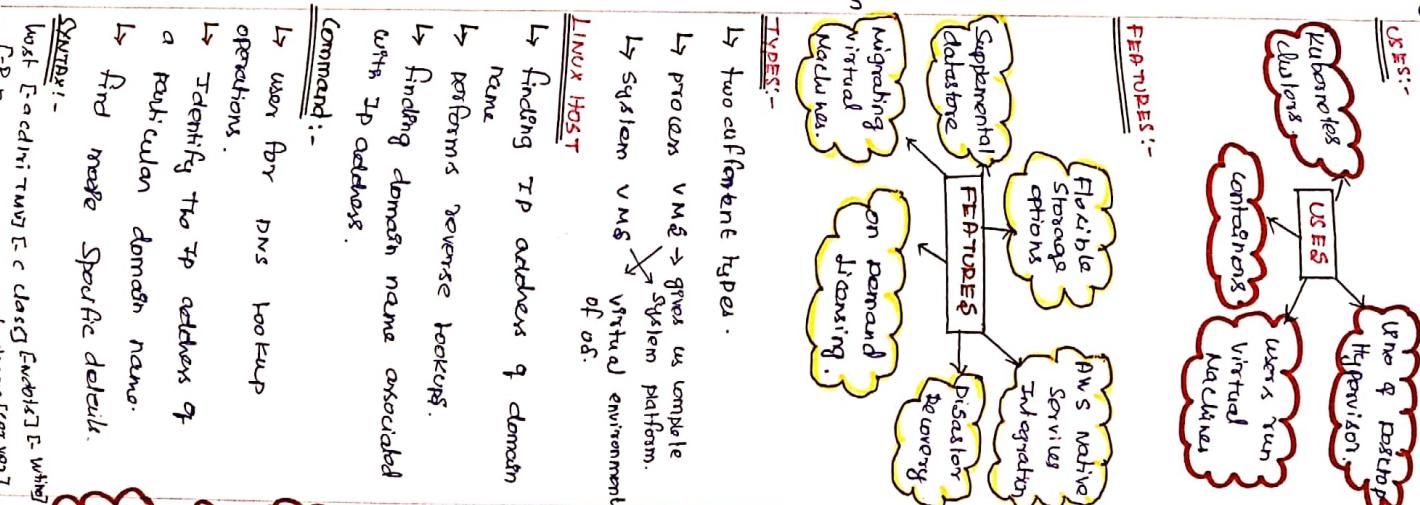
place two bootable media, that contains xen server

VMWARE		
Processor	Processor	Processor
Kernel	Kernel	Kernel
VMM	VMM	VMM
Virtual Machine Manager		

### PROGRAMMING INTERFACE

#### VMWARE

- ↳ run each application in its own os.
- ↳ make better use of physical servers in CPU.
- ↳ more efficient use of space and energy.



ADDITIVE QUESTIONS

- ↳ To install the OS from an ISO image.

Saved the ISO image in any location

Create New Virtual Machine

Select typical to accept configurations.

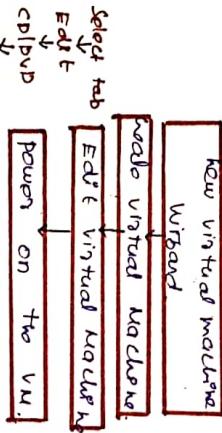
Select Installer disc Image file (ISO)

Click Browse, navigate to the location

New Virtual machine Wizard

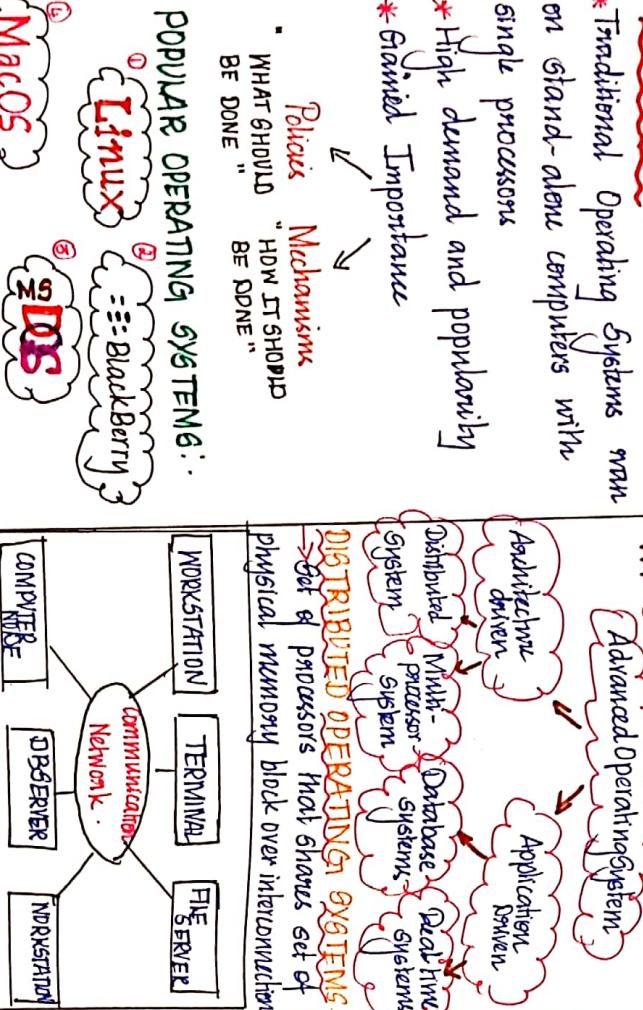
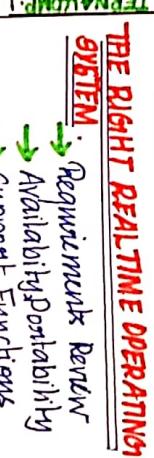
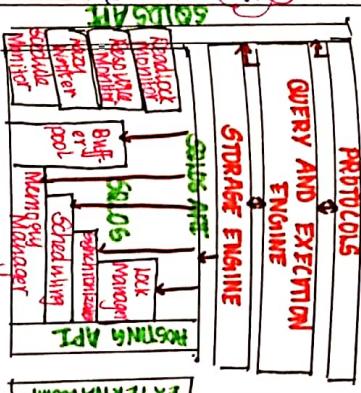
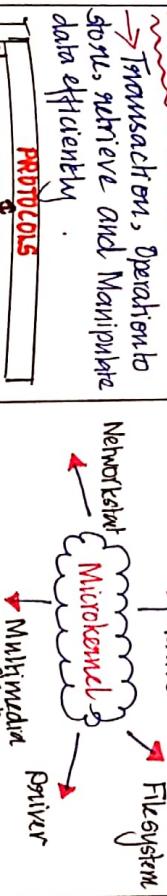
Select tab Edit virtual machine

Power on the VM



## TOPIC 20: ADVANCES IN OPERATING SYSTEMS

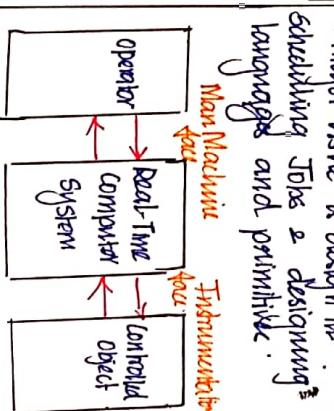
### CHANGING TRENDS IN OS



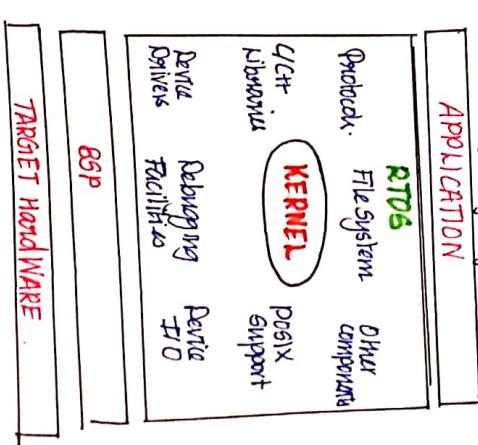
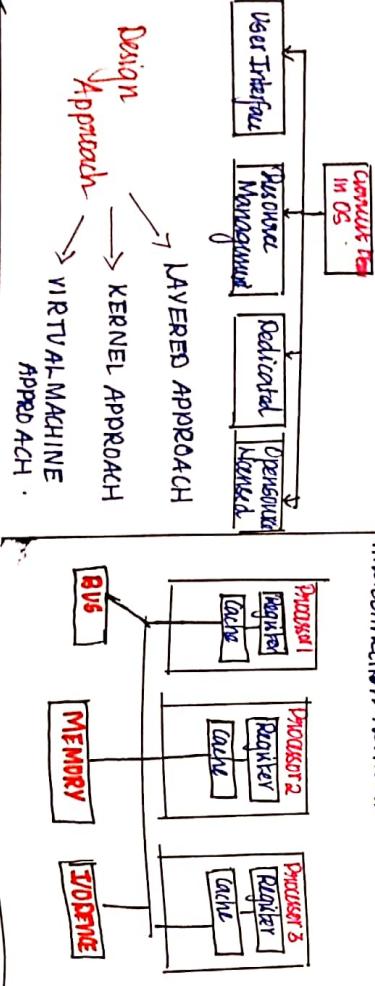
**MULTIPROCESSORS SYSTEMS :-**

→ These are connected through communication Network.

→ Set of processors that shares a set of physical memory blocks over interconnection network.



- \* Advanced Operating System
- \* Database, Realtime, Database Systems
- \* Application Driven
- \* Architecture driven
- \* Multi-processor System



### 1) FIRST COME FIRST SERVED

- \* Jobs executed on first come first served basis
- \* Non-preemptive
- \* Average waiting time high

Process Name	Arrival Time	Burst Time (ms)
P <sub>1</sub>	15	3
P <sub>2</sub>	17	5
P <sub>3</sub>	19	8

Solution:-

Round Chart

P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>
15	18	23

$$\text{Average waiting time of processes} = \frac{WTP_1 + WTP_2 + WTP_3}{3}$$

$$= \frac{0+1+4}{3} = 1.67 \text{ ms}$$

$$\text{Average Turnaround Time} = \frac{TATP_1 + TATP_2 + TATP_3}{3}$$

$$= \frac{3+6+7}{3} = 5.33 \text{ ms}$$

### 2) SHORTEST JOB FIRST(SJF):

- \* Minimum waiting time
- \* Non-preemptive / Preemptive
- \* Required CPU time to be known in advance.

Process Name	Arrival Time	Burst Time (ms)
P <sub>1</sub>	0	8
P <sub>2</sub>	1	4
P <sub>3</sub>	2	9
P <sub>4</sub>	3	5

Solution:-

P <sub>1</sub>	P <sub>2</sub>	P <sub>4</sub>	P <sub>3</sub>
0	8	12	17

### Non-preemptive SJF:-

$$\text{Waiting Time} = \text{Starting Time} - \text{Arrival Time}$$

Average Waiting Time

$$= \frac{(0-0)+(8-1)+(17-2)+(12-3)}{4}$$

$$= \frac{(0+7+15+9)}{4} = 7.75 \text{ ms}$$

Average Turnaround Time

$$= \frac{(8+0)+(4+7)+(9+15)+(5+9)}{4}$$

$$= 14.25 \text{ ms}$$

Turn around time = Burst time + waiting time

### Preemptive SJF:-

Round Chart	P <sub>1</sub>	P <sub>2</sub>	P <sub>4</sub>	P <sub>1</sub>	P <sub>3</sub>
0 2 4 6 8 10 12 14 16 18 20	P <sub>1</sub>	P <sub>2</sub>	P <sub>4</sub>	P <sub>1</sub>	P <sub>3</sub>

$$\text{Average waiting time} = \frac{WTP_1 + WTP_2 + WTP_3}{3}$$

$$= \frac{0+1+4}{3} = 1.67 \text{ ms}$$

$$\text{Average Turnaround Time} = \frac{TATP_1 + TATP_2 + TATP_3}{3}$$

$$= \frac{3+6+7}{3} = 5.33 \text{ ms}$$

$$= \frac{(7+4+24+7)}{4} = 13 \text{ ms}$$

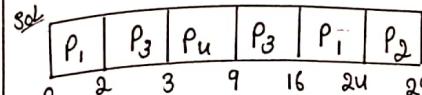
Preemptive algorithm - process can be interrupted even before completion  
Non-preemptive process not interrupted until completion.

### 3) PRIORITY SCHEDULING:-

- \* Process with highest priority to be executed first
- \* Preemptive / Non-preemptive
- \* Can be lowest order or highest order priority based algorithm

### ROUND ROBIN SCHEDULING

process Name	Arrival time	Burst time	priority
P <sub>1</sub>	0	10	6
P <sub>2</sub>	1	5	9
P <sub>3</sub>	2	8	4
P <sub>4</sub>	3	2	2



Lowest order preemptive scheduling

$$= \frac{(16-0-2)+(24-1)+(9-2-1)+(3-3)}{4}$$

$$= \frac{(14+23+6+0)}{4} = 10.75 \text{ ms}$$

$$\text{Average turnaround time} = \frac{(10+14)+(5+23)+(8+6)+(6+0)}{4} = 17.75 \text{ ms}$$

Lowest order non-preemptive scheduling

Solution:-

P <sub>1</sub>	P <sub>4</sub>	P <sub>3</sub>	P <sub>2</sub>
0 10 16 24 29	P <sub>1</sub>	P <sub>4</sub>	P <sub>3</sub>

$$\text{Average waiting time} = \frac{(0-0)+(24-1)+(16-2)+(10-3)}{4} = 13.5 \text{ ms}$$

$$\text{Average turnaround time} = \frac{(10+0)+(5+23)+(8+14)+(6+7)}{4} = 18.25 \text{ ms}$$

### 4) ROUND ROBIN SCHEDULING

process	Arrival time	Burst time
P <sub>1</sub>	0	10
P <sub>2</sub>	1	20
P <sub>3</sub>	2	25

$$\text{AVG. waiting time} = \frac{0+(30-1-10)+(40-2-10)}{3} = 15 \text{ ms}$$

$$\text{Avg. Turnaround time} = \frac{(10+39+53)}{3} = 34 \text{ ms}$$

### 5) MULTILEVEL QUEUE SCHEDULING

\* Preemptive

\* Each process provided fixed time to execute

### 5) MULTILEVEL QUEUE SCHEDULING

\* Process permanently assigned to a queue

process	Arrival time	Burst time	Queue Number
P <sub>1</sub>	0	4	1
P <sub>2</sub>	0	3	1
P <sub>3</sub>	0	8	2
P <sub>4</sub>	10	5	1

- \* Priority of Queue 1 greater than Queue 2
- \* Queue 1 uses Round Robin (2ms) and Queue 2 uses FCFS.

Solution:-

P <sub>1</sub>	P <sub>2</sub>	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>	P <sub>3</sub>	P <sub>1</sub>
0 2 4 6 7 10 15 20	P <sub>1</sub>	P <sub>2</sub>	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>	P <sub>3</sub>

Average waiting time

$$WTP_1 = 4-2-0 \Rightarrow 2$$

$$WTP_2 = 6-2-0 \Rightarrow 4$$

$$WTP_3 = 15-3-0 \Rightarrow 12$$

$$WTP_4 = 10-10 \Rightarrow 0$$

$$\text{Average} = \frac{2+4+12+0}{4} = \frac{18}{4} = 4.5 \text{ ms}$$

### 6) MULTILEVEL FEEDBACK QUEUE SCHEDULING

\* Process burst time 40 sec.

\* First level 2s and each level incremented by 5 sec.

\* Queue 1 - 2 sec

\* Queue 2 - 7 sec

\* Queue 3 - 12 sec

\* Queue 4 - 17 sec

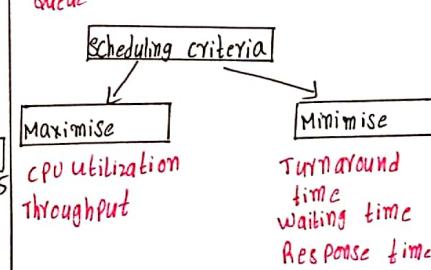
\* Queue 5 - 22 sec

Process interrupted 4 times

4 times & complete on queue 5

on queue 5

Algorithm allows process to move between queue



- 4) Consider the following processes with given Burst Time & Arrival Time

Processes	Burst Time	Arrival Time	Priority
A	25	2	3
B	12	12	1
C	6	18	4
D	18	21	2

Apply the following algorithm and find completion time (CT), Turn around Time (TAT), Waiting time (WT) and Response Time (RT)

- 1) FCFS (First Come First Serve)

- 2) SJF (Short Job First)

- 3) Round Robin with Quantum time

- 4) Priority Scheduling

- 2) Consider the following processes with given Burst time and arrival time

Processes	Burst Time	PrioritY
A	25	3
B	12	1
C	6	4
D	18	2

Apply the following algorithm and find completion time (CT), Turn around time (TAT), Waiting time (WT) and Response time (RT)

- 1) FCFS (First Come First Serve)

- 2) SJF (Short Job First)

- 3) Round Robin with Quantum time

- 4) Priority Scheduling

- 3) Consider the following processes with given Burst time and Arrival time

Process ID	Arrived time	Burst time
1	0	3
2	1	2
3	2	1
4	3	4
5	4	5
6	5	2

If the context switching time of the system is 1 unit after every process, find the scheduling Algorithm United part done waiting time.

- 4) Consider the following processes with given Burst time (BT) and Arrival time (AT).

Calculate average waiting time of the above processes if SJF preemptive scheduling is followed.

Process Name	Burst Time	Arrival Time
P <sub>1</sub>	6	2
P <sub>2</sub>	2	5
P <sub>3</sub>	8	1
P <sub>4</sub>	3	0
P <sub>5</sub>	4	4

Calculate average waiting time of the above processes if SJF Non-preemptive scheduling is followed.

## Deadlock avoidance using Banker's Algorithm

Consider a system with 5 processes P<sub>0</sub> through P<sub>4</sub> and 3 resource types A, B, C.

Allocation	MAX	Available
A B C	A B C	A B C
P <sub>0</sub> 0 1 0	7 5 3	3 3 2
P <sub>1</sub> 2 0 0	3 2 2	
P <sub>2</sub> 3 0 2	9 0 2	
P <sub>3</sub> 2 1 1	2 2 2	
P <sub>4</sub> 0 0 2	4 3 3	

Find out if the system is in a safe state for the following sequence <P<sub>1</sub>, P<sub>3</sub>, P<sub>4</sub>, P<sub>0</sub>, P<sub>2</sub>>

→ Calculate Need Matrix

$$\text{Need}[i,j] = \text{Max}[i,j] - \text{Allocation}[i,j]$$

Process	Allocation	MAX	Need
A B C	A B C	A B C	
P <sub>0</sub> 0 1 0	7 5 3	7 4 3	
P <sub>1</sub> 2 0 0	3 2 2	1 2 2	
P <sub>2</sub> 3 0 2	9 0 2	6 0 0	
P <sub>3</sub> 2 1 1	2 2 2	0 1 1	
P <sub>4</sub> 0 0 2	4 3 3	4 3 1	

$$\text{Initially } \text{work} = \text{Available} = 3 3 2$$

Finish(i) = false for all i=0, ..., 4

Finish(0) = false, finish(1) = false, finish(2) = false

finish(3) = false, finish(4) = false

Check the sequence <P<sub>1</sub>, P<sub>3</sub>, P<sub>4</sub>, P<sub>0</sub>, P<sub>2</sub>>

for P<sub>1</sub> Check if need<sub>P1</sub> ≤ work

$$(1 2 2) \leq 3 3 2 \Rightarrow \text{True}$$

Step 2: work = work + Allocation<sub>P1</sub>

$$= 3 3 2 + 2 0 0$$

$$= 5 3 2$$

finish(P<sub>1</sub>) = True got step 2

for P<sub>3</sub> check if finish(P<sub>3</sub>) = false and  
Step 2: Need<sub>P3</sub> ≤ work  
 $0 1 1 \leq 5 3 2 \Rightarrow \text{True}$   
Step 3: work = work + Allocation<sub>P3</sub>  
 $= 5 3 2 + 2 1 1$   
 $= 7 4 3$   
finish(P<sub>3</sub>) = True got step 2

for P<sub>4</sub>  
Step 2: check if finish(P<sub>4</sub>) = false and  
Need<sub>P4</sub> ≤ work  
finish(P<sub>4</sub>) = false  $\times 4 3 1 \leq 7 4 3 \Rightarrow \text{True}$   
Step 3: work = work + Allocation<sub>P4</sub>  
 $= 7 4 3 + 0 0 2$   
 $= 7 4 5$   
finish(P<sub>4</sub>) = True got step 2

for P<sub>0</sub>  
Step 2: if finish(P<sub>0</sub>) = false & Need<sub>P0</sub> ≤ work  
finish(P<sub>0</sub>) = false  $\times 4 3 1 \leq 7 4 5 \Rightarrow \text{True}$   
Step 3: work = work + Allocation<sub>P0</sub>  
 $= 7 4 5 + 0 1 0$   
 $= 7 5 5$   
finish(P<sub>0</sub>) = True got step 2

for P<sub>2</sub>  
Step 2: if finish(P<sub>2</sub>) = false & need<sub>P2</sub> ≤ work  
finish(P<sub>2</sub>) = false  $\times 6 0 0 \leq 7 5 5 \Rightarrow \text{True}$

Step 3: work = work + Allocation<sub>P2</sub>  
 $= 7 5 5 + 3 0 2$   
 $= 1 0 5 7$

finish(P<sub>2</sub>) = True

if finish(i) = True for all i=0, ..., 4 then  
the system is in a safe state for the sequence  
safe state  $\Rightarrow < P_1, P_3, P_4, P_0, P_2 >$

## Resource Request Algorithm

→ used to check the request can be granted or not  
check if P<sub>i</sub> request (1, 0, 2) be granted or not  
request, (1, 0, 2)

Step 1: request<sub>i</sub> ≤ Need<sub>i</sub>, go to Step 2  
else

$$1 0 2 \leq 1 2 2 \Rightarrow \text{True} \text{ go to step 2}$$

Step 2: request<sub>i</sub> ≤ Available, go to step 3  
else

$$1 0 2 \leq 3 3 2 \Rightarrow \text{True} \text{ go to step 3}$$

$$\begin{aligned} \text{Available} &= \text{Available} - \text{request}_i \\ &= 3 3 2 - 1 0 2 \\ &= 2 3 0 \end{aligned}$$

$$\begin{aligned} \text{Allocation}_i &= \text{Allocation}_i + \text{Request}_i \\ &= 2 0 0 + 1 0 2 \\ &= 3 0 2 \end{aligned}$$

$$\begin{aligned} \text{Need}_i &= \text{Need}_i - \text{Request}_i \\ &= 1 2 2 - 1 0 2 \\ &= 0 2 0 \end{aligned}$$

Now update the allocation & need matrix of previous

Process	Allocation	MAX	Need
A B C	A B C	A B C	
P <sub>0</sub>	0 1 0	7 5 3	7 4 3
P <sub>1</sub>	<u>3 0 2</u>	3 2 2	<u>0 2 0</u>
P <sub>2</sub>	3 0 2	9 0 2	6 0 0
P <sub>3</sub>	2 1 1	2 2 2	0 1 1
P <sub>4</sub>	0 0 2	4 3 3	4 3 1

$$\text{work} = \text{Available} = 2 3 0$$

Now have to check the sequence <P<sub>1</sub>, P<sub>3</sub>, P<sub>4</sub>, P<sub>0</sub>, P<sub>2</sub>> using Safety algorithm

if finish(i) = True for all i=0, ..., 4 process in the system, then the system process P<sub>i</sub> request can be granted immediately

P<sub>i</sub>  $\Rightarrow$  granted

## Banker's Algorithm

## TUTORIAL DAY 2

### Problem - 01

A single processor system has 3 resource types X, Y and Z, which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column following denotes the no. of units of each resource type allocated to each process, and the column request denotes the no. of units of each resource type requested by a process in order to complete execution. Which of these processes will finish LAST?

Process	Alloc			Request		
	X	Y	Z	X	Y	Z
P <sub>0</sub>	1	2	1	1	0	3
P <sub>1</sub>	2	0	1	0	1	2
P <sub>2</sub>	2	2	1	1	2	0

An operating system uses the banker's algorithm for deadlock avoidance. When managing the allocation of 3 resource types X, Y and Z to 3 processes P<sub>0</sub>, P<sub>1</sub> and P<sub>2</sub>, the table given below presents the current system state. Here the Allocation matrix shows the current no. of resources of each type allocated to each process and the Max matrix shows the maximum no. of resources of each type required by each process during its execution.

Process	Allocation			Max		
	X	Y	Z	X	Y	Z
P <sub>0</sub>	0	0	1	8	4	3
P <sub>1</sub>	3	2	0	6	2	0
P <sub>2</sub>	2	1	1	3	3	3

- The are 3 units of type X, 2 units of type Y and 2 units of type Z still available. The system is currently in safe state. Consider the following independent requests for additional resources in the current state:
- Req1 : P<sub>0</sub> requests 0 units of X, 0 units of Y and 2 units of Z still available.  
 Req2 : P<sub>1</sub> requests 2 units of X, 0 units of Y and 0 units of Z which of the following is True?
- Only Req1 can be permitted
  - Only Req2 can be permitted
  - Both Req1 and Req2 can be permitted
  - Neither Req1 nor Req2 can be permitted
- Problem - 03
- Consider the following snapshot of a system using Banker's Algorithm. Answer the following questions using the Banker's Algorithm.
- | Process        | Allocation | Max |
|----------------|------------|-----|
| P <sub>0</sub> | 0          | 0   |
| P <sub>1</sub> | 3          | 2   |
| P <sub>2</sub> | 2          | 1   |
- i) What is the content of the matrix Need?  
 ii) Is the system in a safe state?  
 iii) If a request from process P<sub>1</sub> arrives for (0, 4, 2, 0) can the request be granted immediately?
- Problem - 04
- | Process        | Allocation      | Max         | Available       |
|----------------|-----------------|-------------|-----------------|
| P <sub>0</sub> | 0 0 1 2 0 0 1 2 | 4 4 4 4 4 4 | A B C D A B C D |
| P <sub>1</sub> | 1 0 0 0 1 7 5 0 | 4 4 4 4 4 4 | A B C D A B C D |
| P <sub>2</sub> | 1 3 5 4 2 3 5 6 | 4 4 4 4 4 4 | A B C D A B C D |
| P <sub>3</sub> | 0 6 3 2 0 6 5 2 | 4 4 4 4 4 4 | A B C D A B C D |
| P <sub>4</sub> | 0 0 1 4 0 6 5 6 | 4 4 4 4 4 4 | A B C D A B C D |
- Solve using Banker's Algorithm.
- What is the content of need matrix?
  - Check if the system is in a safe state?
  - Determine the total sum of each type of resource?

- P<sub>0</sub>
- P<sub>1</sub>
- P<sub>2</sub>
- None of the above since the system is in a deadlock

## PAGING & SEGMENTATION Q&A

### PAGING:-

i) Consider a paging system with the page table stored in memory.

a) If a memory reference takes 50 nano seconds, how long does a paged memory reference take?

b) If we add TLB's and 75 percent of all page-table references are found in the TLB's, what is the effective memory reference time?

**Important Formulas**  
Virtual Address Space = Size of process.  
Number of pages divided = Process Size / Page Size

a) Given

Time for memory reference takes place = 50 ns.

Time for paged memory reference.

$$\begin{aligned} &= \text{Time for accessing table} + \\ &\quad \text{Time for accessing the target data in memory.} \\ &= 50 \text{ ns} + 50 \text{ ns} \\ &= 100 \text{ ns.} \end{aligned}$$

b) Assume that finding a page table entry in the TLB takes 0 ns, if the entry is present.

$$\begin{aligned} \text{The effective memory reference time} \\ \text{time}_{avg} &= 0.75 \times (50 \text{ ns} + 20 \text{ ns}) \\ &\quad + 0.25 \times (2 \text{ ns} + 50 \text{ ns} + 50 \text{ ns}) \\ &= 64.5 \text{ ns.} \end{aligned}$$

2) Assuming a 1-KB page size what are the page-numbers and offsets for the following addresses (provided as decimal numbers)?

Solution:-

$$\text{Answer: } \text{size} = 2^n = 1 \text{ KB} = 1024 = 2^{10}$$

# bits in offset part ( $n$ ) = 10.

Steps:-

1. Convert logical address: Decimal to binary.
2. Split binary address to 2 parts (page #, offset), offset: n digits.
3. Convert offset and page #: Binary to decimal.

a) 3085

Decimal = 3085

**Important Formulas:-**  
Size of page table = No. of entries in page table  $\times$  page table entry size

Binary = 000000110000001101

Page # = 011 = 3

Page offset = 0000001101 = 13.

b) 42095

Decimal = 42095.

Binary = 1010010001110111

Page # = 101001 = 41

Page offset = 0001110111 = 11.

c) 215201

Decimal = 215201, Binary = 110100100010100001

Page # = 11010010 = 210

Page offset = 0010100001  
= 161.

d) 650000

Decimal = 650000

Binary = 10011101011000100000

Page # = 100111010 = 634

Page offset = 100010000 = 784.

### SEGMENTATION

i) Consider the following segment table

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Eg 11.2.

∴ it is illegal.

2) Consider a logical address space of 256 pages with 4 KB page size mapped on to physical memory of 64 frames.

- a) How many bits are required in logical address?
- b) How many bits are required in physical address?

Solution

a) Logical address space  $(1s130) = 2^m$ .

$$\begin{aligned} \text{Logical address space } (1s130) &= \\ &= \# \text{ of pages} \times \text{page size}. \\ &= 256 \times 4 \text{ KB} \\ &= 256 \times 4096 (\underline{4 \times 1024}) \\ &= 1048576. \\ &= 2^m \\ &\therefore 2^m = 2^{20} \end{aligned}$$

$$\therefore m = 20 \quad \text{II.}$$

what are the physical addresses for the following logical addresses?

a. 0, 430

b. 1, 10

c. 2, 500

d. 3, 400

e. 4, 112.

Solution:-

Segment Logical Address Offset

a) 0, 430.

$$\begin{aligned} \text{Physical address} &= \text{Base} + \text{logical} \\ &\quad \text{Address offset} \\ &= 219 + 430 (\underline{600 > 430}) \quad \text{b) Let } x \text{ be the no. of physical} \\ &= 649. \quad (\underline{649 > 600}) \quad \text{addresses} \end{aligned}$$

b) 1, 10

c) 2, 500

d) 3, 400

e) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

a) 2, 500

b) 3, 400

c) 4, 112.

Physical addresses

## VITAL

## DAY 2

### PROBLEMS:-

### Important formulas:-

### for Main memory:-

\* Physical address = Size of Main memory.

\* Size of Main memory =  
 Total number of frames  
 $\times$   
 Page size.

\* If number of frames in main memory =  $x^x$ . Then the number of bits in frame number =  $x$  bits.

\* If page size =  $x^x$  bytes, then the number of bits in page offset =  $x$  bits.

\* If the size of Main Memory =  $x^x$  bytes then the number of bits in physical address =  $x$  bits.

### for process:-

\* Virtual Address Space = Size of Process

\* Number of pages the process is divided =  $\frac{\text{Process size}}{\text{Page size}}$ .

\* If process size =  $x^y$  bytes, then the number of bits in virtual address space =  $x$  bits.

### NOTE:-

\*  $n$  bits  $\rightarrow 2^n$  locations.

\* Size of memory =  $2^n$  size of one location.

### PRACTICE PROBLEMS:-

1) Consider the logical address space of 64 pages of 1024 words each, mapped onto a physical memory of 32 frames.

- a) How many bits are there in the logical address?
- b) How many bits are there in the physical address?

2) Assume 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers).

- a. 2058
- b. 43496
- c. 225402
- d. 670000.

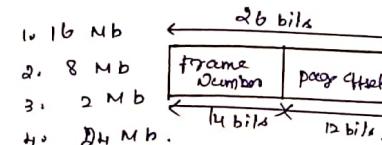
3) Calculate the number of bits required in the address for memory having size of 16 GB. Assume the memory is 4-byte addressable.

4) Calculate the size of memory if its address consists of 32 bits and the memory is 2 byte addressable.

5) Consider a system with byte-addressable memory, 32 bit logical addresses, 4KB page size and page table entries of 4 bytes each. The size of page table in the system? (in Mbytes).

Size of memory =  $2^n$  size of one location.

6) Consider a machine with 64 MB physical memory and a 32 bit virtual address space. If the page size is 4 KB, what is the approximate size of page table?



7) In a virtual memory system, size of virtual address is 32 bits, the size of physical address is 36 bits, page size is 4 kbyte and size of each page table is (entry) 32 bits.

The main memory is byte addressable. Find the maximum no. of bits can be used to store protection and other information in table entry?

### SEGMENTATION:-

#### PROBLEM:-

Consider the following segment table.

Segment No	Base	Length
0.	1219	700
1.	2300	14
2.	90	100
3.	1327	580
4.	1952	96.

which of the following logical address will produce trap addressing error?

1. 0, 480

2. 1, 11

3. 2, 100

4. 3, 425

5. 4, 95.

Calculate the physical address if no trap is produced.

2) A certain computer system has the segmented paging architecture for virtual memory. The memory is byte addressable. Both virtual and physical addresses spaces contain 216 bytes each. The virtual address space is divided into 8 non-overlapping equal size segments. The Memory Management unit (MMU) has a hardware segment table, each entry of which contains the physical address of the page table for the segment. Page tables are stored in the main memory of 2 bytes page table entries.

What is minimum page size in byte so that the page table of a segment requires at most one page to store it?

### Important formulas:-

\* Virtual Address =  $\frac{\text{Process size}}{\text{Space}}$

\* Physical Address =  $\frac{\text{Main memory size}}{\text{Space}}$

\* Size of each segment =  $\frac{\text{Process size}}{\text{No. of segments}}$

# PAGE REPLACEMENT ALGORITHMS

JAY 4

## Page Replacement Algorithm

- \* Select frames to be replaced.
- \* String of memory by process called reference string
- \* Types
  - FIFO
  - LRU
  - Optimal

## FIFO page Replacement

- \* Simplest Algorithm
- \* Replace old page brought into memory.
- \* Performance not always good.
- \* may suffer from Belady's Anomaly.

## LRU page Replacement

- \* Replace page not been used long period of time
- \* Looking backward in time
- \* Did not suffer from Belady's Anomaly.

## Optimal Page Replacement

- \* Lowest page fault rate
- \* Never suffer from Belady's Anomaly.
- \* Replace page - Not Used for longest period of time
- \*

## FIFO Example

\* consider the reference string

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Number of frames = 3.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
0	0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	0	0	0
1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1		
x	x	x	x	v	x	x	x	x	x	v	v	y	x	v	v	x	x	x	x
f	f	f	f	f	f	f	f	f	f	f	f	h	h	f	f	h	f	f	f

x = page fault  
(F)

H = page hit (H)

Page Fault = 15      Page Hit = 5.

## LRU Example

The reference string

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0
4	1	1	1	3	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
x	x	x	x	v	x	v	x	x	x	v	v	y	v	x	v	x	v	v	v

x = page fault      v = page hit

Page faults = 12      Page hits = 8

## Optimal Page Replacement Example

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
0	0	0	0	0	0	4	4	4	4	0	0	0	0	0	0	0	0	0	0
1	1	1	1	3	3	3	3	3	3	3	3	3	3	3	1	1	1	1	1
x	x	x	x	v	x	v	x	v	v	x	v	v	x	v	v	x	v	v	v

Page Fault = 9  
Page Hit = 11

\* Replace page not used for long period of time in future.

## TUTORIAL QUESTIONS - PAGE REPLACEMENT DAY 4

1. A system uses 3 page frames for storing process pages in main memory. It uses the
  - (i) First In First Out (FIFO) page Replacement policy.
  - (ii) Least Recently Used (LRU) page Replacement policy.
  - (iii) Optimal page Replacement policy

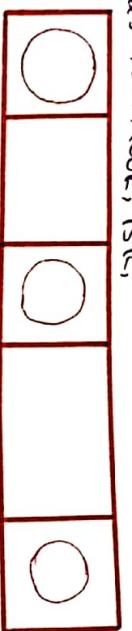
Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page Reference string given below 4, 7, 6, 1, 7, 6, 2, 7, 2. Also calculate the hit ratio and miss ratio.
2. Consider page reference string 1, 3, 0, 3, 6, 3 with 3 page frames. Find the number of page faults using
  - (i) First In First Out (FIFO) page Replacement policy
  - (ii) Least Recently Used (LRU) page Replacement policy
  - (iii) Optimal page Replacement policy.
3. Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 3, 0, 3, 2, 3 with 4 page frames. Find number of page faults using
  - (i) First In First Out (FIFO) page Replacement Policy
  - (ii) Least Recently Used (LRU) page Replacement Policy
  - (iii) Optimal page Replacement policy
4. Consider a main memory with four-page frames and the following sequence of page references 3, 8, 2, 3, 9, 1, 6, 3, 8, 6, 2, 1, 3. Find number of page faults using First In First Out (FIFO) and Least Recently Used (LRU)?
5. A process refers to 5 pages, A, B, C, D, E in the order: A, B, C, D, A, B, E, A, B, C, D, E. If the page replacement algorithm is FIFO, the number of page transfers with an empty internal store of 3 frames is?



## TUTORIAL

**Process Request are given as:-**

25K, 50K, 100K, 75K,



50K  
75K  
150K  
175K  
300K.

Determine the algorithm which can optimally satisfy the Requirements

1. first fit algorithm
2. Best fit algorithm
3. Neither of the two
4. Both of them.

Problem:-1

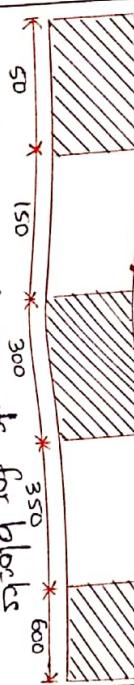
Consider six memory partitions of size 200KB, 400KB, 600KB, 800KB, 1000KB and 1200KB. These partitions need to allocated to four processes of sizes 357KB, 210KB, 468KB and 941KB in that order.

Perform the allocation of processes using:-

1. first fit algorithm
2. Best fit algorithm
3. Worst fit algorithm

consider the following heap (figure) in which blank regions are not in use and hatched regions are in use:-

Main Memory



The sequence of Requests for blocks of size 300, 250, 150, 250, 150 can be satisfied if we use.

1. either first fit or best fit policy (anyone)
2. first fit but not best fit policy
3. Best fit but not first fit policy
4. None of the above.

Problem:-4

Given six memory partitions of 300KB,

200KB, 350KB, 200KB, 150KB, 600KB, 400KB, 600KB, 300KB and 250KB. Need to allocated

in order how would the first-fit, best-fit and worst-fit algorithms place processes of size 115KB, 500KB, 358KB, 200KB and 375KB (in order).

Problem:-5

Assume a computer with a memory size of 512K, initially empty. Requests are received for blocks of memory of 15K, 10K, 53K and 80K. Show how the buddy system would deal with each

Request, showing the memory layout at each stage and the states of lists at the end. After allocating all the processes. what would be the effect of the 15K fragmentation this type of allocation suffers with?

Problem:-6

Given memory partitions of 100K, 50K, 60K, 100K, 200K, 600K, determine the available space list after allocating the space for the stream of requests consisting of the following block sizes 250, 100, 150 K. vice.

(i) First fit  
(ii) Best fit  
(iii) Worst fit.

Problem:-7

Given 5 partitions of 100KB, 500KB, 200KB, 300KB, 200KB (in order)

use first fit, worst fit, best fit, algorithms to place 512KB, 117KB, 112KB, 162KB (in order)

# FILE CONCEPT

## TUTORIAL DRAFTS

- 1) consider a secondary storage system of size 2 TB, with 512-Byte sized blocks.

Assume that the filesystem uses a multilevel inode datastructure to track data blocks of file. The inode has 64 bytes of space available to store pointers to data blocks, including a single indirect block, a double indirect block, and several direct blocks.

What is the maximum size that can be stored in such file system?

A:- Number of data blocks:  $\frac{2^{41}}{2^9} = 2^{32}$   
So 32 bits or 4 bytes are required to store number of a data block.

Number of data block pointers in inode =  $\frac{64}{4} = 16$ , of which 14 are direct blocks. The single indirect block stores pointers to  $\frac{512}{4} = 128$  data blocks.

The double indirect block points to 128 single indirect blocks, which in turn point to 128 data blocks each.

- 3) calculate the disk capacity for a hard disk that supports 2 surfaces, 32 tracks/surface, 64 sectors/track and 128 bytes/sector.  
Ans:- Disk Capacity = surfaces × Tracks / surface × sectors / track × bytes / sector

$$\text{Disk capacity} = 2 \times 32 \times 64 \times 128 \\ = 524288 \text{ bytes}$$

$$\text{Disk capacity} = 2^{20} \text{ bytes}$$

A:-

for one block, the following conditions hold. In the contiguous allocation case, assume that there is no room to grow at beginning but there is room to grow at end. Also assume that block information to be added is stored in memory.

- a. The block is added at beginning  
b. The block is added in middle  
c. The block is added at end.

- d. The block is removed from beginning.  
e. The block is removed from middle.  
f. The block is removed from end.

A:-

	contiguous	linked	indexed
a.	201	1	1
b.	101	52	1
c.	1	3	1
d.	198	1	0
e.	98	52	0
f.	0	100	0

- 5) The fast file file system uses an inode array to organize the files on disk. Each inode consists of a user id (2 bytes), three timestamps (4 bytes each), protection bits (2 bytes), a reference count (2 byte), a file type (2 bytes) and the size (4 bytes). Additionally, the inode contains 13 direct indexes, 1 index to a 1st-level index table, Index to 2nd level table, and 1-index to 3rd index table. The file system also stores first 436 bytes of each file in the inode. Assume a disk sector is 512 bytes, and assume that any auxiliary index table takes up an entire sector, what is maximum size for file in file system?

A:- Correct answer is:  $436 + 13 \times 512 + 1 \times 128 \times 512 + 1 \times 128 \times 128 \times 512 = 1082203060$

- 6) consider a file system that uses inodes to represent files. Disk blocks are 1KB in size, and a pointer to a disk block requires 8 bytes. This file system has 10 direct disk blocks, as well as single and double indirect disk blocks. What is maximum size in GB of a file that can be stored?  
Ans:- Disksize: 12KB  
No. of pointers in ablock would be:  $\frac{128KB}{8KB} = 1536$   
Total file size: 10 direct disk block size:  $10 \times 12KB$   
+ 1 single indirect size:  $1 \times 1536 \times 12KB$   
+ 1 double indirect size:  $1 \times 1536 \times 1536 \times 12KB = 27.6GB$

- 2) consider a file currently consisting of 100 records where each record requires 40 bytes (and the index block, in the case of index allocation) is already in memory. Calculate how many cylinders are required for this file?
- Ans:- Each sector can hold  $\frac{512}{40} = 12.8$  records.  
The file requires  $50,000 / 12.8 = 3906.25$  sectors.  
normally 1 cylinder can hold: 1000 sectors.  
No. of cylinders required is approx =  $3906.25 / 1000 = 3.906$

## Disk Scheduling

## Address Translation

**Day 5**

Disk Scheduling :-

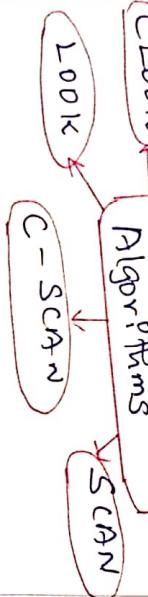
- \* To schedule 10 requests arriving for disk.

Seek Time :-  
→ Time taken to locate head over disk.

Algorithms:

FCFS

SSTF



- \* First come first serve.
- \* Request served in orders.

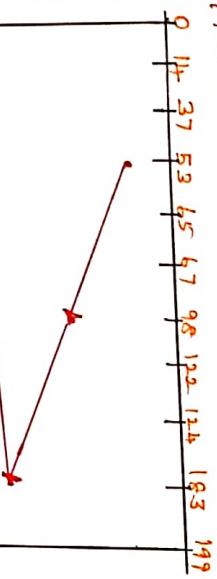
Ex: Order of Request → 98, 183, 37, 122, 14, 124,

65, 67  
65' Pointer : 53.

Total Head Movement = 936.

SCAN:

One end → Other end



Calculate Absolute Distance

Request in Ascending Order

Head

Total Head Movement = 936.

= (199 - 53) + (199 - 0) + (37 - 0)

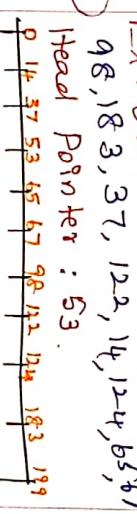
= 382 cylinders.

= 640 cylinders.  
Total Head Movement = 640

SSTF :

- \* Shortest seek time serve
- \* Closed request serve first.

Ex: Order of Request → 98, 183, 37, 122, 14, 124, 65, 67  
Head Pointer : 53.



$$= (53 - 0) + (183 - 0) \\ = 236 \text{ cylinders.}$$

Total Head Movement = 236.

C-SCAN :-

→ More uniform wait time than SCAN.

Example: Order of Request

98, 183, 27, 122, 14, 124, 65, 67

Head Pointer : 53.

(d) (3, 400)

= 500 < 100 (false)

= Illegal Reference, trap to O.S.



$$= (199 - 53) + (199 - 0) + (37 - 0)$$

Total Head Movement = 382

(e) (4, 112) < 96 (false)

= Illegal Reference, trap to O.S.

= (98 - 53) + (183 - 37) + (122 - 14) + (124 - 65) + (67 - 65)

Example:- Order of Request  
Translation:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

(a) What are the physical address for the logical Address?  
a) (0, 430) b) (4, 10) c) (2, 500)  
d) (3, 400) e) (4, 112)

Sol:  
(a) (0, 430) = 430 + 600 (true)  
= 219 + 430 (offset)  
= 649  
⇒ physical Address = 649.

(b) (1, 10) = 10 + 14 (true)  
= 2300 + 10  
= 2310  
⇒ physical Address = 2310

(c) (2, 500) = 500 < 100 (false)

Illegal Reference,

trap to O.S.

(d) (4, 112) = 400 < 96 (true)

= 400 + 1327

= 1727

⇒ physical Address = 1727

## TUTORIAL - DISK-SCHEDULING PROBLEMS

- Suppose the order of request is - (80, 140, 43, 140, 24, 16, 190) And current position of Read / write head is '50'. Find the total seek time for "SSTF algorithm."

- Suppose the request to be addressed are (80, 140, 43, 140, 24, 16, 190). And the Read / write arm is at 50, and it is also given that the disk arm should move "towards the larger value". calculate the seek time for "SCAN algorithm."
- Suppose the requests to be addressed are (80, 140, 43, 140, 24, 16, 190). And the Read / write arm is at 50, and it is also given that the disk arm should move "towards the larger value". compute the seek time for "CSCAN algorithm."
- Suppose the request to be addressed are (80, 140, 43, 140, 24, 16, 190). And the Read / write arm is at 50, and it is also given that the disk arm should move "towards the larger value". calculate the seek time for "LOOK algorithm."
- Suppose the request to be addressed are (80, 140, 43, 140, 24, 16, 190). And the Read / write arm is at 50, and it is also given that the disk arm should move "towards the larger value". calculate the seek time for "LOOK algorithm."
- Suppose the request to be addressed are (80, 140, 43, 140, 24, 16, 190). And the Read / write arm is at 50, and it is also given that the disk arm should move "towards the larger value". calculate the seek time for "C-LOOK algorithm."
- Suppose a disk drive has 5000 cylinders, numbered 0 to 4999. the drive currently services a seek time for "C-LOOK algorithm".
- Suppose at cylinder 143, and the previous request was at cylinder 125, the queue of pending request in FIFO order is 86, 140, 93, 144, 948, 1509, 1022, 1450, 130. starting from the current position, what is the total distance (in cylinders) that the disk arm moves to satisfy all pending requests, for each of the following algorithms ? , FCFS , SSTF , SCAN , C-SCAN , LOOK , C-LOOK