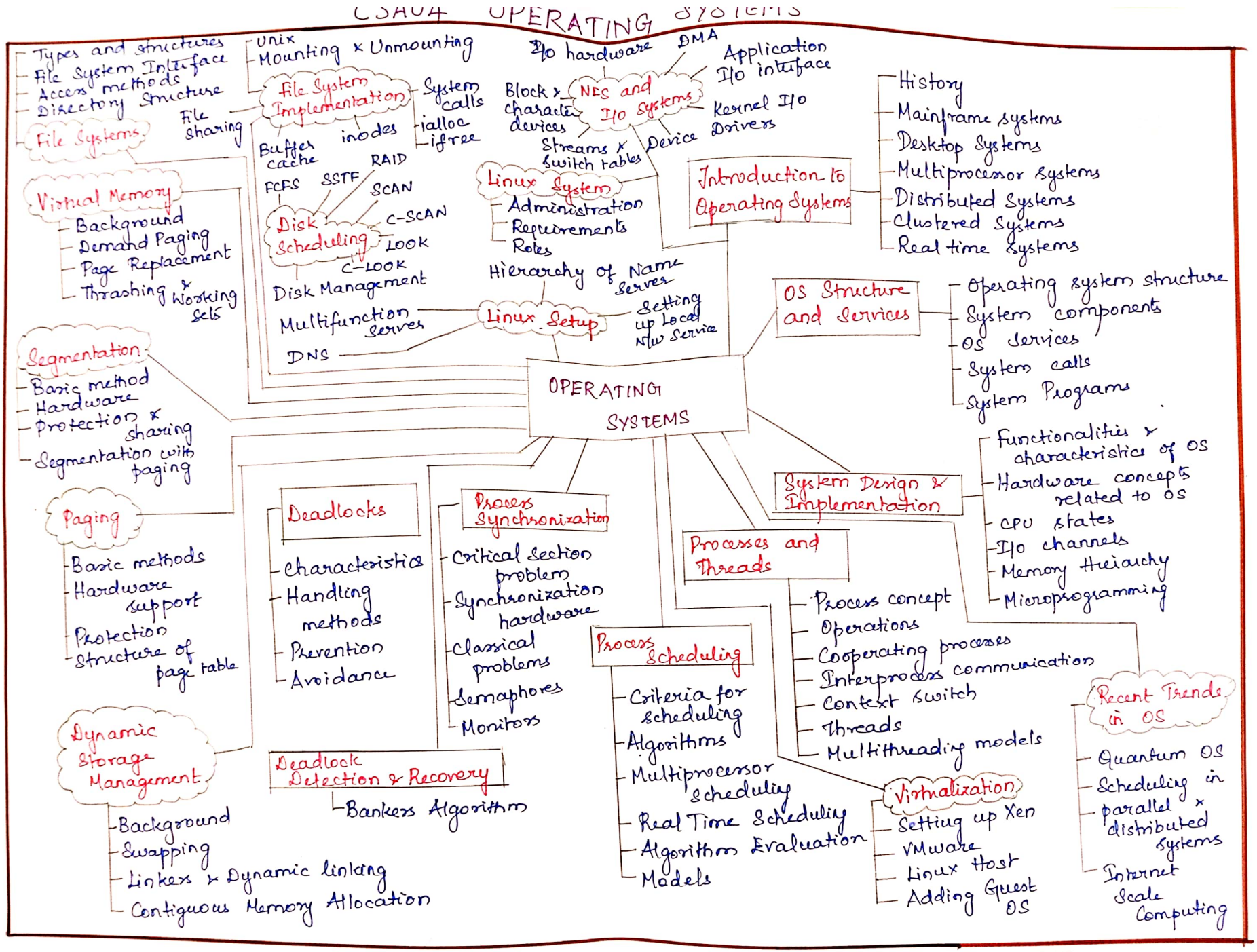
CSA04-OPERATING SYSTEMS

Concept Mapping

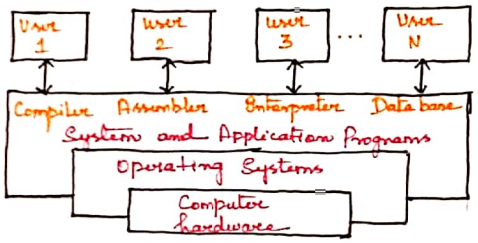

CSA04 OPERATING SYSTEMS
OPERATING SYSTEMS
File Systems
Types and structures
File System Interface
Access methods
Directory Structure
File sharing
File System Implementation
Unix
Mounting & Unmounting
System calls
ialloc
ifree
Buffer cache
inodes
I/O hardware
DMA
Application I/O interface
Kernel I/O Drivers
Device Drivers
Block & character devices
Streams & Switch tables
NFS and I/O Systems
Introduction to Operating Systems
History
Mainframe systems
Desktop Systems
Multiprocessor Systems
Distributed Systems
Clustered Systems
Real time Systems
Linux System
Administration
Requirements
Roles
Virtual Memory
Background
Demand Paging
Page Replacement
Thrashing & Working Sets
FCFS
SSTF
SCAN
C-SCAN
LOOK
C-LOOK
RAID
Disk Scheduling
Disk Management
Multifunction Servers
DNS
Hierarchy of Name Servers
Linux Setup
Setting up Local Nw Servic
OS Structure and Services
Operating system structure
System components
OS services
System calls
System Programs
Segmentation
Basic method
Hardware
Protection & sharing
Segmentation with paging
Paging
Basic methods
Hardware support
Protection
Structure of page table
Deadlocks
Characteristics
Handling methods
Prevention
Avoidance
Process Synchronization
Critical section problem
Synchronization hardware
Classical problems
Semaphores
Monitors
Deadlock Detection & Recovery
Bankers Algorithm
Process Scheduling
Criteria for scheduling
Algorithms
Multiprocessor scheduling
Real Time Scheduling
Algorithm Evaluation
Models
Processes and Threads
Process concept
Operations
Cooperating processes
Interprocess communication
Context switch
Threads
Multithreading models
System Design & Implementation
Functionalities & characteristics of OS
Hardware concepts related to OS
CPU states
I/O channels
Memory Hierarchy
Microprogramming
Dynamic Storage Management
Background
Swapping
Linkers & Dynamic linking
Contiguous Memory Allocation
Virtualization
Setting up Xen
VMware
Linux Host
Adding Guest OS
Recent Trends in OS
Quantum OS
Scheduling in parallel & distributed systems
Internet Scale Computing

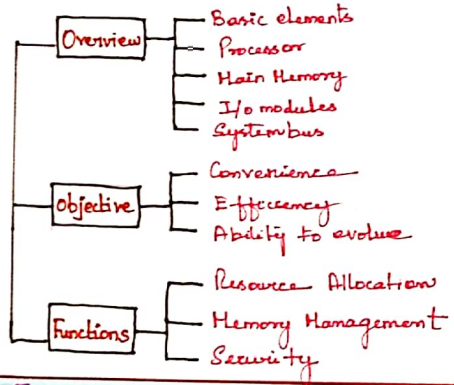
Topic 1 : INTRODUCTION TO OPERATING SYSTEM

OPERATING SYSTEM

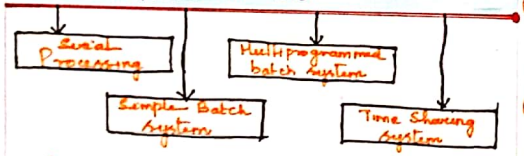
User — intermediate between — Computer hardware

Components of Computer system

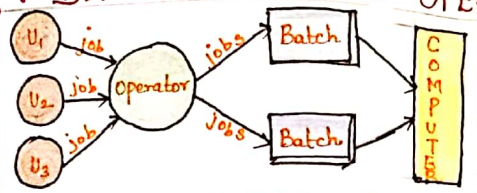




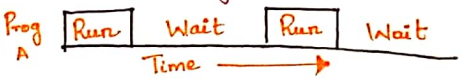
EVOLUTION OF OS



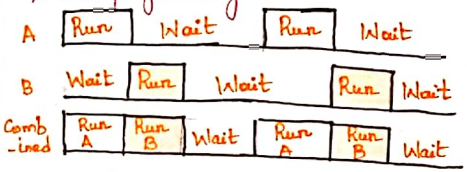
* **Serial Processing :**
  - ↳ No OS.
  - ↳ Run from Console — display lights, toggle switches
  - ↳ 2 problems — Scheduling, Setup time
* **Simple Batch system :**
  - ↳ User submit job to computer Operator
  - ↳ Use of — Monitor's point of view, Processor point of view



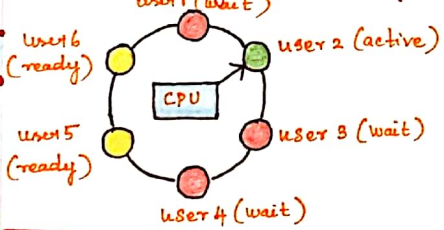
* **Multiprogrammed Batch system**
  - ↳ one/multiple programs loaded into main memory — Uniprogramming, Multiprogramming
  - ↳ Uniprogramming



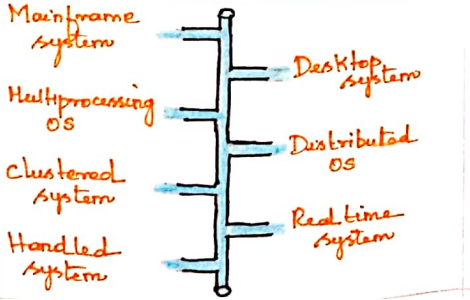
  - ↳ Multiprogramming



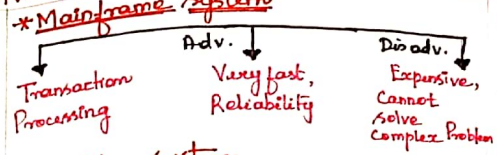
* **Time Sharing system**
  - ↳ Multiple Users share Computers simultaneously (Multitasking)



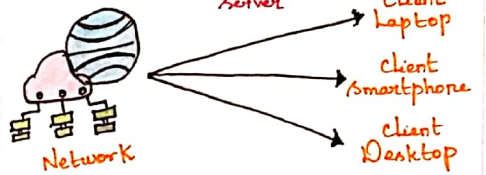
TYPES OF OS



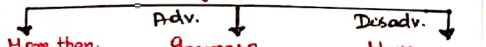
* **Mainframe system**

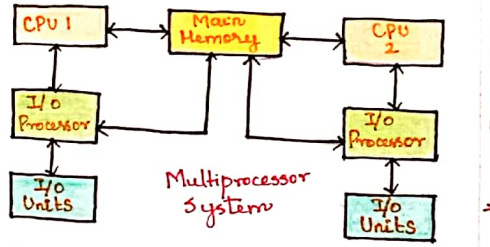


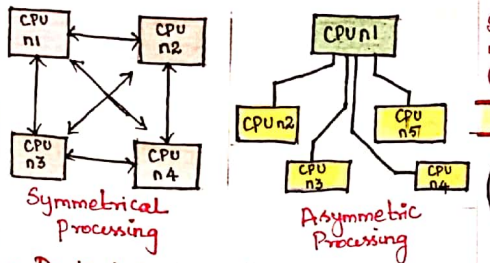
* **Desktop system**



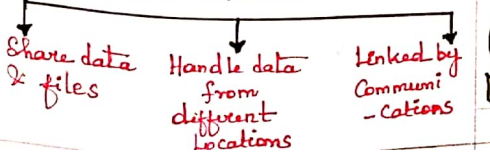
* **Multiprocessing Operating system**

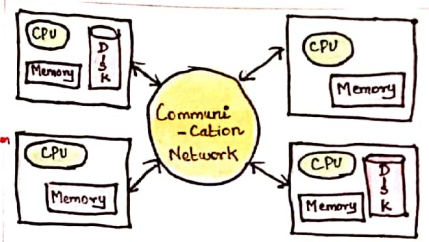




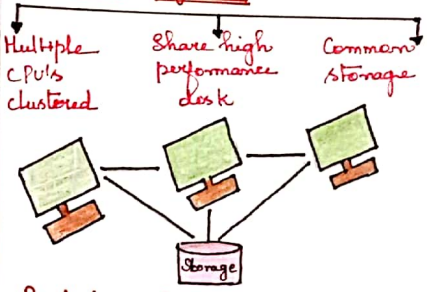


* **Distributed System**



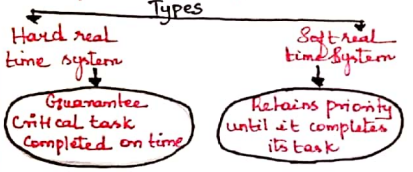


* **clustered System**

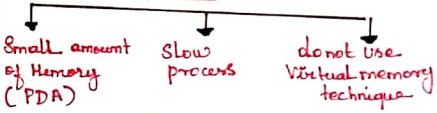


* **Real time System**
  - ↳ time taken by the system to respond to an i/p

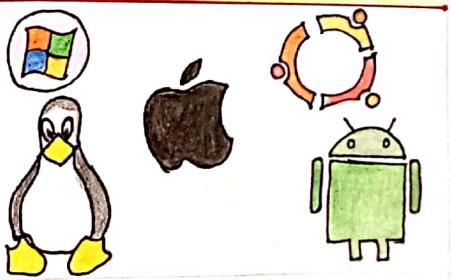
Types



* **Handheld system**



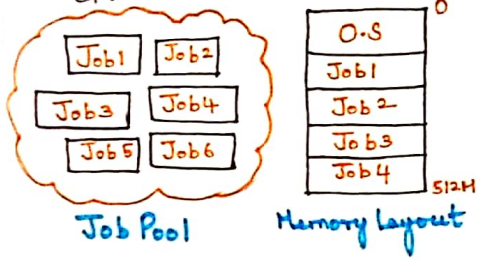
APPLICATIONS



TOPIC 2 : OS STRUCTURES AND SERVICES

OS STRUCTURE

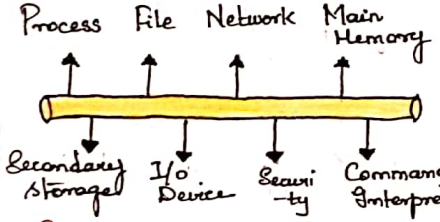
* Vary greatly in their makeup internally
* Commonalities [Multiprogramming, Time Sharing]
* Multiprogramming : increases CPU utilization



* Time Sharing : Many Users share computer simultaneously

SYSTEM COMPONENTS

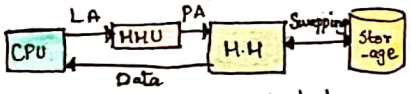
* Large & Complex system partioned into small parts



* Process Management :
  - Manages many process running simultaneously.
  - Execution is sequential
  - Example : Search Engine like Chrome.

- Functions :
  1. process creation and deletion
  2. Suspension, Synchronization :
* File Management
  - Represents programs & data
  - Functions :
    1. file directory Creation, deletion, manipulation
* Network Management
  - Administrating Computer networks
  - Functions : access shared Resources, involves minicomputers, microprocessor
* Main Memory Management
  - Large array of storage
  - Functions : Allocate Memory - Process Request

* Secondary storage Management
  - Store data permanently

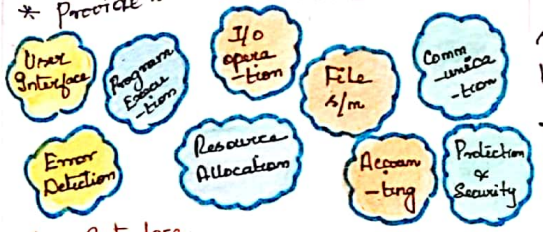


  - Functions : Disk scheduling
* I/O Management
  - Hide specific h/w devices from User

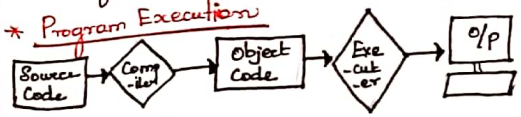
  - Functions : buffer caching system
* Security Management
  - Control the access of programs
  - improve reliability
* Command Interpreter system
  - Interface [User, Rest of system]

OS SERVICES

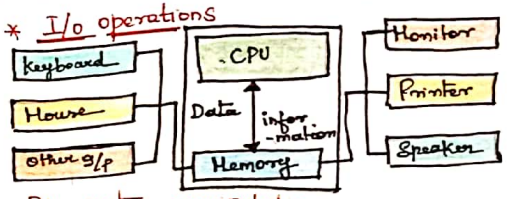
* Provide services [Programs, Users]



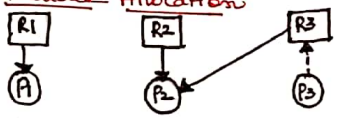
* User Interface
  - Command line Interface [CLI]
  - Graphical User Interface [GUI]
* Program Execution



* I/O operations



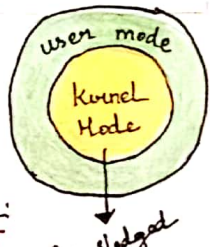
* File system manipulation
  - Read, Write, Create, delete, search
* Communications exchange information
  - Same Computer
  - Computers Over N/w
* Error detection occurs in
  - CPU
  - Memory hardware
  - I/O Devices
  - User Programs
* Resource Allocation



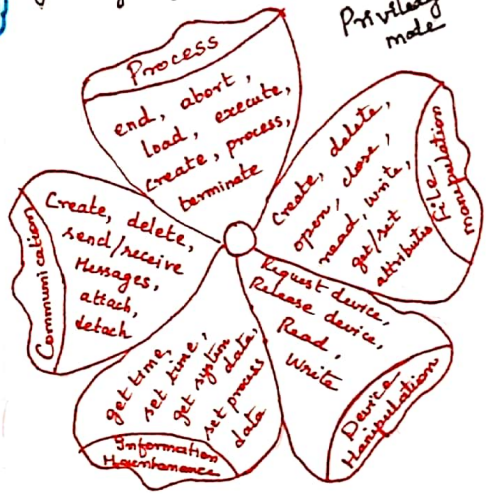
* Accounting : Track of Users use Resources
* Protection & Security :
  - Ensure Resources Controlled
  - User Authentication

SYSTEM CALLS

Interface to the services available by an OS.



Types of System call:



SYSTEM PROGRAM

Convenient Environment
- Program development
- Execution

File Management	Status Information	Programming Language
Create Delete Copy Rename Print Dump List	Date Time Amount of Memory space No. of Users Logging Debugging	Compilers Assemblers Debuggers Interpreters

File Modification	Loading Execution	Communi-cation
Create file Modify file Search file	Absolute Loader Relocatable Loader Linkage Editor Overlay Loaders	Send Message Browse web page emails Log Remotely

System Design & Implementation

- Internal structure of different OS can vary widely
- start the design by defining goals and specifications
- Affected by choice of hardware, type of system

user goals → Convenient, easy to learn reliable, Safe and fast

system goals → Easy to design, Easy to implement, easy to maintain, flexible, reliable, error free & efficient

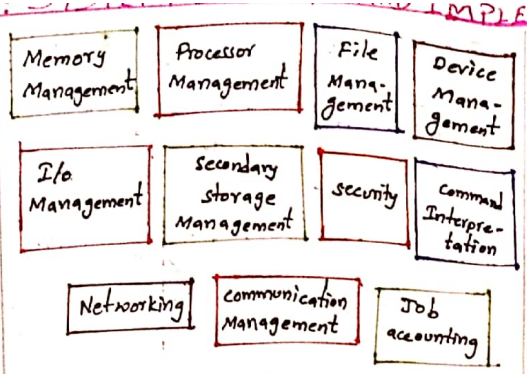
Important principle to seperate
* Policy - what will be done?
* Mechanism - How to do it?

Implementation :

- Early Os in assembly language
- Then system programming languages
- Now C, C++
- Usually a mix of languages
  * lowest level in assembly
  * Main body in C
  * System programs in C, C++, PERL Python, shell script

Functionalities & characteristics of OS

* Operating systems provides essential functions for managing devices connected to a Computer.

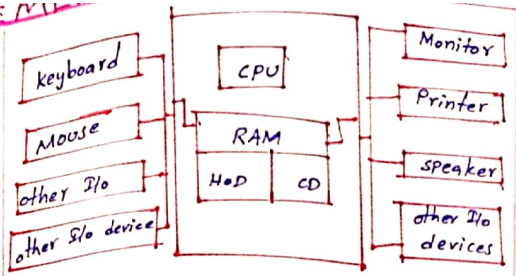


characteristics of Operating System

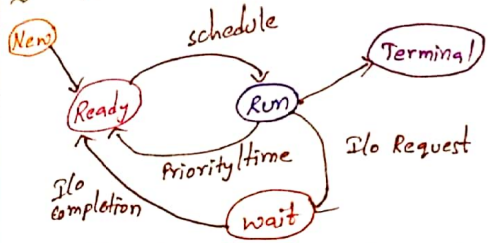
* Memory Management - keeps track of the Primary memory.
* Processor Management - Allocates the cpu and deallocates the processer when it is no longer required.
* Device Management - keeps track of all devices
* File Management - Allocates and deallocates the resources
* Security - Prevents unauthorized access to programs
* Job accounting - keeps track of time and resources
* Control over system performance - Records delays
* Interaction with the operators - Intraction may take place.
* Error detecting Aids - Production of dumps, traces, error massages and error-detecting methods
* Coordination b/w other software and users - coordination and assignment of compilers, interpreters, assemblers and other software

Hardware Concepts :-

* Collection of several components working together.
* Comprises of CPU and peripherals.
* H/w refers to Mechanical device that makes up computer
* H/w are CPU, keyboard, Mouse, harddisk, Printer etc..
* H/w can understand only low level language or Machine language
* Works only on binary codes (1's and 0's)



CPU states in Os :-

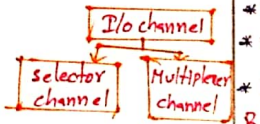


* New state - when a program present in secondary memory
* Ready state - Moves from New state to ready state after it is located in M
* Run state - Ready to Run state after it is assigned the CPU for execution.
* wait state - A process Moves from run to wait state if requires I/o ope
* Terminate state : Moves from run to terminate after completion.

I/O channel

It has ability to execute I/o instructions using special-purpose processor on I/o channel and complete control over I/o operations.

Types of I/o channel

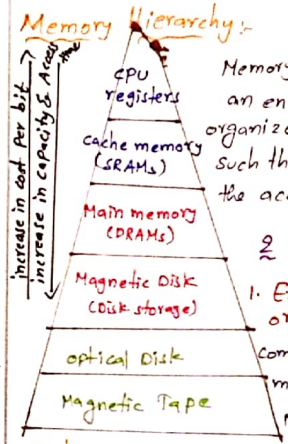


* selector channel - Selector channel controls multiple high-speed devices.
* Multiplexer channel - It can handle multiple devices at the same time. It Can do block transfers for several devices at once.

Two types of multiplexers are used in this channel:

1. Byte Multiplexer - It is used for low-speed devices.
2. Block Multiplexer - It accepts or transmits block of characters.

Memory Hierarchy :-



Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time.

2 types :-

1. External Memory or secondary memory comprising of Main memory, optical disk, Magnetic tape e

* Internal memory or primary memory comprising of Main memory, Cache memory & CPU registers

Micro Programming

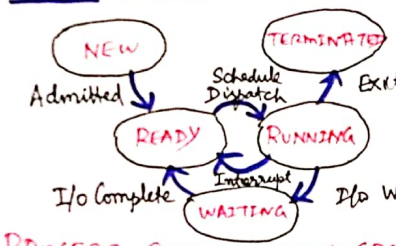
* It is the process of writing Micro code
* Low level code
* Defines about Microprocessor functions
* Machine language instructions
* Translate to several Microcode
* Realization of control unit
* Commands Adds to chips
* reduction of size of control memory
* Binary values stored as word.

Real time os design Method
→ Layer
→ vm approach
→ Kernel

TOPIC 4 : PROCESS and THREADS

PROCESS:
- Program in execution
- Need resources → CPU, I/O, FILES
- 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:-
- READY QUEUE: Process residing in memory waiting to execute
- DEVICE QUEUE: List of processes waiting for an I/O device.

SCHEDULERS:-
THREE TYPES

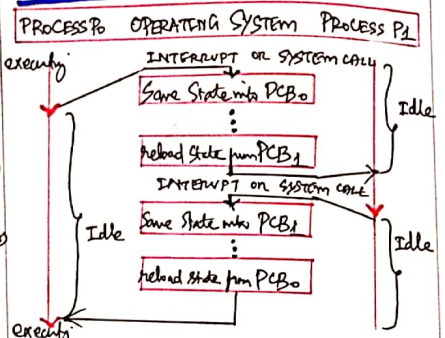
LONG TERM	SHORT TERM	MEDIUM TERM
- Job Scheduler - Select processes and loads into memory for execution	- CPU Scheduler - Allocate CPU to one of the Selected Processes	- To reduce overhead of degree of multiprogram

PROCESS CLASSIFICATION
Two Types

- I/O BOUND: - Spends most of the time doing I/O
- CPU BOUND: - Generates I/O requests infrequently

CONTEXT SWITCH:-
- Save the current context of Process
- Load the context of new process in the PCB.

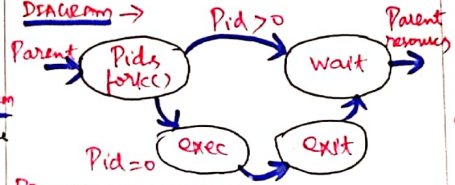
DIAGRAM:



OPERATIONS ON PROCESSES:-
PROCESS CREATION
- Create Parent and Child Processes.
- Process identifier assigned to each process

CHILD:-
- can be a duplicate of Parent or can have a new program

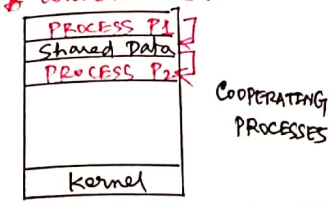
DIAGRAM →



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

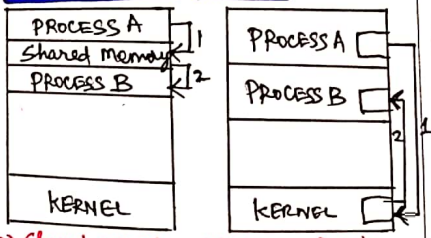
Advantages:-
- * Information Sharing
- * Computation Speed up
- * Modularity
- * Convenience.



INTERPROCESS COMMUNICATION (IPC)
- Communicating with the process (shared / not shared)

MODELS OF IPC:-
- * Shared Memory
- * Message Passing
  - Allow process to exchange information.

COMMUNICATION MODELS:-



a) Shared memory b) Message Passing

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

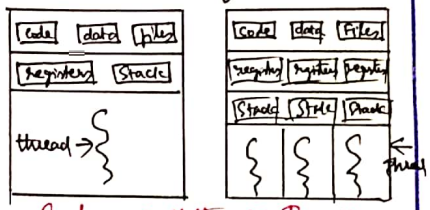
TYPES OF BUFFER
- BOUNDED BUFFER: * Fixed Buffer Size
- UNBOUNDED BUFFER: * No practical limit on the buffer

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 → User threads, kernel threads

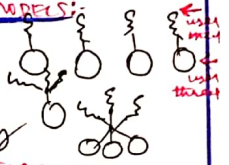
MULTI THREADING:-
- executing multiple threads simultaneously.



* Single & Multithreaded Processes

BENEFITS OF MULTITHREADING:
- * Responsiveness * Resource sharing
- * Economy * Utilization of multiprocessor A/c

MULTITHREADING MODELS:
- * one-to-one
- * many-to-one
- * many-to-many



REAL TIME APPLICATION:-
Multithreaded
- used in the web browser
- used in the Games

Multithreaded → AUDIO, TEXT, VIDEO

TOPIC 5 : PROCESS SCHEDULING

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

OBJECTIVE OF MULTIPROGRAMMING
- Maximize CPU Utilization

SCHEDULING CRITERIA

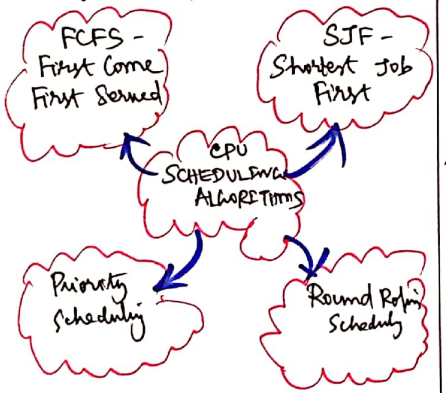
MAXIMIZE
* CPU Utilization
  - CPU remains as busy as possible
  - 100% utilization
* THROUGHPUT
  - no. of processes that finish their execution per unit time

MINIMIZE
* WAITING TIME
  - Process needs to wait in the ready queue
* RESPONSE TIME
  - amount of time the request submitted until the first response
* TURN AROUND TIME
  - the amount of time to execute the specific process

CPU SCHEDULING → 4 Circumstances

Preemptive
1. * Process switches from running state to ready state
2. * Process switches from waiting state to ready state

Non-preemptive
3. * Process switches from running state to waiting state
4. * A process terminates or completes



(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
P_1	24
P_2	3
P_3	3

Gantt chart:-

P_1	P_2	P_3
0	24	27 30

(i) Waiting time of $P_1 = 0$
(ii) Waiting time of $P_2 = 24$
(iii) Waiting time of $P_3 = 27$
AWT = (0+24+27)/3 = 51/3 = 17 ms

TAT (Turn around time)
TAT = BT + WT
(i) TAT of $P_1 = 24+0 = 24$
(ii) TAT of $P_2 = 3+24 = 27$
(iii) TAT of $P_3 = 3+27 = 30$
ATAT = (24+27+30)/3 = 81/3 = 27 ms

(ii) SJF → Shortest Job First
- Process with shortest execution time.
- reduces average waiting time
- Shortest job executed first and shortest turn around time.

Gantt chart

P_2	P_3	P_1
0 3	6	30

(i) WT of $P_1 = 6$
(ii) WT of $P_2 = 0$
(iii) WT of $P_3 = 3$
AWT = (6+0+3)/3 = 9/3 = 3 ms

(i) TAT of $P_1 = 24+6 = 30$
(ii) TAT of $P_2 = 3+0 = 3$
(iii) TAT of $P_3 = 3+3 = 6$
ATAT = (30+3+6)/3 = 39/13 = 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_1	24	2
P_2	3	1
P_3	3	3

Gantt chart:-

P_2	P_1	P_3
0 3	27	30

(i) WT of $P_1 = 3$
(ii) WT of $P_2 = 0$
(iii) WT of $P_3 = 27$
AWT = (3+0+27) = 30/3 = 10 ms

(i) TAT of $P_1 = 24+3 = 27$
(ii) TAT of $P_2 = 3+0 = 3$
(iii) TAT of $P_3 = 3+27 = 30$
ATAT = (27+3+30)/3 = 60/3 = 20 ms

(iv) ROUND ROBIN ALGORITHM:-
- Algorithm comes from Round Robin principle
- mostly used for scheduling algorithm in multitasking
- It is a hybrid model which is clock-driven
- Time Slice should be minimum assigned for specific task to be processed.

TIME SLICE = 5

Gantt chart:-

P_1	P_2	P_3	P_1	P_1	P_1	P_1
0 5	8	11	16	21	26	30

(i) WT of $P_1 = 6$
(ii) WT of $P_2 = 5$
(iii) WT of $P_3 = 8$
AWT = (6+5+8)/3 = 19/3 = 6.3 ms

(i) TAT of $P_1 = 24+6 = 30$
(ii) TAT of $P_2 = 3+5 = 8$
(iii) TAT of $P_3 = 3+8 = 11$
ATAT = 49/3 = 16.3 ms

MULTIPLE-PROCESSOR SCHEDULING OR MULTIPROCESSOR SCHEDULING
* Consists of more than one process
* multiple CPUs share the load so that various processes run simultaneously.
* In multiprocessor scheduling, many processes are identical and running process at any time.

REAL TIME SCHEDULING:
* It carry real time tasks
* Tasks performed immediately with a certain degree of urgency
* classified into → Hard Real time tasks, Soft Real time tasks

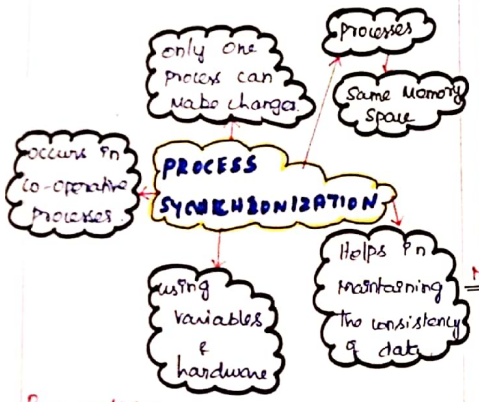
ALGORITHM EVALUATION:
- To compare all four scheduling algorithm, SJF → Shortest Job first scheduling algorithm produces best results.

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 (thread life) of control
- classified into → user level threads, kernel level threads (consume of the)
- 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)

TOPIC 6 : PROCESS SYNCHRONIZATION.

PROCESS SYNCHRONIZATION

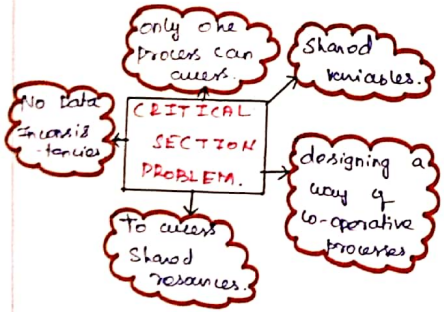


Race condition

↳ More than one process access the same Resource.
↳ possibility of wrong output.

CRITICAL SECTION PROBLEM

↳ Code Segment that can be accessed by only one process.



Solutions to the critical section problem.

- Mutual exclusion.
- progress
- Bounded waiting.

```
do{
    entry section      → entry of a process
        critical section.
    exit section       → allows the process that are waiting in entry section.
        Remainder section
} while (true);
```

Mutual exclusion :-

↳ If one process executing in critical section.
↓
No other processes allowed to execute in critical section.

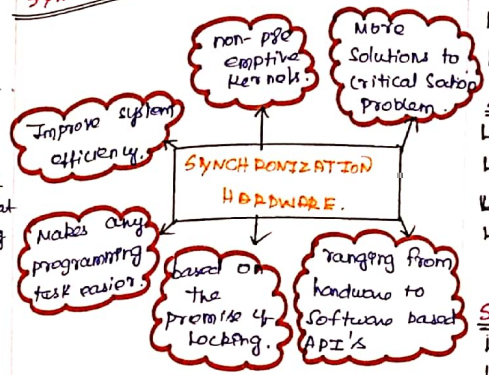
progress

↳ No process in critical section.
↳ other process waiting outside.
↳ only those process that are not in remainder section.
↓
Can decide which will enter in critical section.

Bounded waiting.

↳ bound must exist on the number of times that the process allowed.
↳ enter the critical sections.
↳ after a process has made a request
↓
for getting in to critical section
↓
there is a limit for the no. of. processes.
requests granted.
↓
Limit reached
↓
critical section.

SYNCHRONIZATION HARDWARE :-



TEST AND SET

```
do{
while (test and set (lock))
    //do nothing.
    /* critical section */
    lock = false
    /* remainder section */
} while (true);
```

COMPARE AND SWAP

```
do{
while (compare_and_swap (lock, 0, 1) != 0)
    // do nothing.
    /* critical section */
    lock = 0;
    /* remainder section */
} while (true);
```

CLASSICAL PROBLEMS OF SYNCHRONIZATION.

1. **Bounded buffer problem.**
   ↳ Producer- consumer problem.
   ↳ 2 buffers
      ↳ full, empty
      ↳ Producer produces the product
      ↳ Consumer consumes the product.
2. **Dining philosopher problem :-**
   ↳ philosopher thinking & eating.
   ↳ Don't eat while thinking.
   ↳ hungry → tries to pickup the Chopstick.
3. **Reader Writer problem :-**
   ↳ Set of resources shared.
   ↳ P0, P1, ... Pn-1
   ↓write.
   ↳ while reading no writting.
   ↳ while writting no reading.
4. **sleeping barber problem! -**
   ↳ Barber problem (one barber available)
   ↳ Barber is busy with one customer.
   ↳ rest wait in n-1 chairs.
   ↳ one chair of barber, n customer chair.

SEMAPHORES.

↳ Integer variable.
↳ Involves two variable.
↳ wait () & signal.

↳ wait ().

```
wait (S){
    while(S<=0)
        // busy wait
        S--;
}
```

↳ signal ()

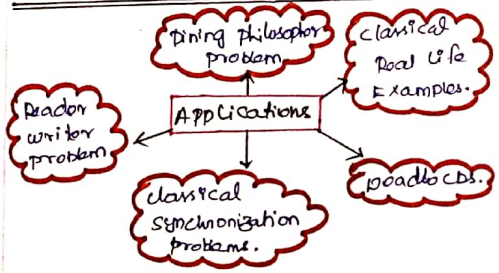
```
signal(S)
{
    S++;
}
```

MONITORS.

```
↳ Monitor monitor name
{
    /* shared variable declaration */
    function P1 (.....) {
        ...
    }
    :
    function Pn (.....) {
        ...
    }
}
```

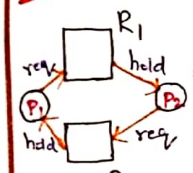
↳ to achieve Mutual exclusion.
↳ only one process at a time.

REAL-TIME APPLICATIONS.

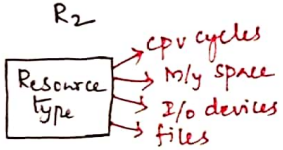


Deadlocks

TOPIC 7

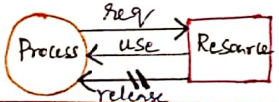


⇒ Cannot change their state of a process ⇒ Deadlock

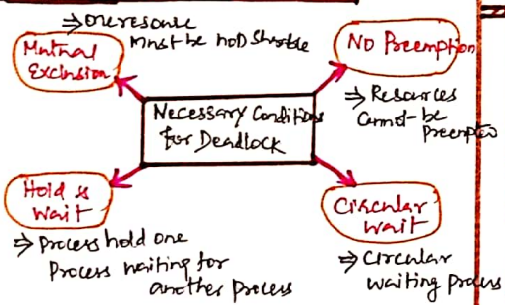


System model

Under normal mode of operation



Deadlock Characterization



Resource Allocation graph

⇒ Represented by directed graph

G = {V, E}

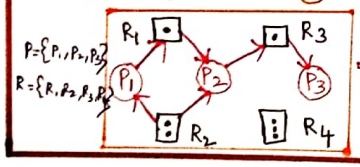
V → Set of process P = {P1 ... Pn}

Set of Resources R = {R1 ... Rn}

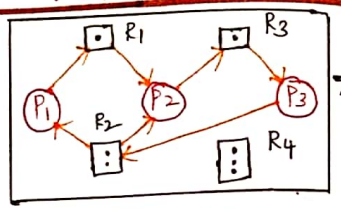
E → Set of Edges E = {Pi → Rn, Rn → Pn}

Assignment edge

Request edge



⇒ Resource Allocation graph



⇒ Graph with a deadlock

cycle → P1 → R1 → P2 → R3 → P3 → R2 → P1

P2 → R3 → P3 → R2 → P2

Graph → No cycles ⇒ No deadlock

Cycles

One instance ⇓ Deadlock

Several instance ⇓ Possibility of Deadlocks

Methods for Handling Deadlock

- Deadlock Prevention
- Deadlock Avoidance
- Deadlock Detection
- Deadlock Recovery

Deadlock Prevention

⇒ one of the condition cannot hold

Mutual Exclusion | Hold & wait | No Preemption | Circular wait

Deadlock Avoidance

- Need Prior Information
- declare Maximum Need
- Examine Circular wait

⇒ Possibility of deadlock

⇒ No deadlock

Avoidance

Single instance ⇓ Resource Allocation graph

Multiple instance ⇓ Banker's Algorithm

Safe state ⇒ only if there exist a Safe Sequence

Banker's Algorithm

* Available - Vector of length 'm'
  available [j] = k
* Max - n x m matrix
  max [i,j] = k
* Allocation - n x m matrix
  Allocation [i,j] = k
* Need - n x m matrix
  Need [i,j] = k

Need [i,j] = max [i,j] − Allocation [i,j]

Safety Algorithm :-

1. Let Work$_m$ and finish$_n$
   Initialize Work = Available
   Finish(i) = false
   for i = 0 ... n-1
2. Find an i such that both
   a. Finish [i] = false // if no such i exist goto step (4)
   b. Need$_i$ ≤ work
3. work = work + Allocation$_i$
   Finish(i) = true
   go to step (2)
4. If finish(i) = true for all i, the the system is in a safe state

Resource Request Algorithm

⇒ determining whether requests can be safely granted or not

Request$_i$ → request vector for process P_i

Request$_i$[j] = k ⇒ P_i wants k instance of resource type R_j

1. If Request$_i$ ≤ Need$_i$ goto step 2
   otherwise raise an error condition
2. If Request$_i$ ≤ Available, goto step 3
   otherwise P_i must wait
3. Allocated the requested resource to P_i
   Available = Available − Request$_i$
   Allocation$_i$ = Allocation$_i$ + Request$_i$
   Need$_i$ = Need$_i$ − Request$_i$
   If Safe ⇒ allocate resources to P_i
   If unSafe ⇒ P_i must wait and restore old resource allocation state

Claim Edge $P_i \rightarrow R_j$

⇒ Process P_i may request resource R_j in the future

⇒ represented by dashed line ⇢

Realtime Examples

⇒ Traffic deadlock

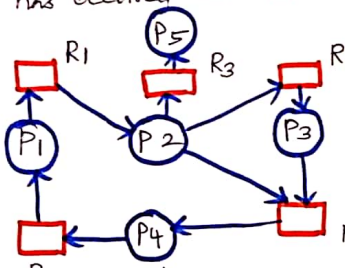
⇒ Banking Application

⇒ Computing environment

TOPIC 8

Deadlock Detection

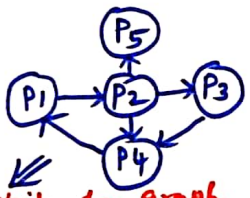
→ An algorithm that examines the state of the system to determine whether a deadlock has occured or not



⇐ Resource allocation graph

$P = \{P_1, P_2, P_3, P_4, P_5\}$

$R = \{R_1, R_2, R_3, R_4, R_5\}$



⇒ by removing the resource nodes and collapsing the appropriate edges

Wait-for graph

⇒ 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
- Single instance → Resource allocation graph → wait-for graph → Detect deadlock if wait-for graph contains cycle
- Multiple instance → Deadlock detection Algorithm

Deadlock detection:

* Available – Vector of length 'm'
  available [j] = k
* Allocation – n x m matrix
  Allocation [i, j] = k
* Request – n x m matrix
  Request [i, j] = k

1. Initialize
   i) Work = Available
   ii) for i = 0, ... n-1
   if Allocation$_i$ ≠ 0
   finish[i] = false
   otherwise
   finish[i] = true
2. Find an i such that
   a) finish[i] = false
   b) Request$_i$ ≤ work
   if no such i exist, go to step 4.
3. Work = Work + Allocation$_i$;
   Finish[i] = true
   go to step 2.
4. If finish[i] = false for some i, 0≤i<n
   then the system is in a deadlocked state
   if finish[i] = false → P_i is deadlocked

Recovery from Deadlock

- Abort one or more processes to break the circular wait → (Process termination)
- Preempt some resources from one or more deadlocked processes → (Resource Preemption)

Process termination

- Abort all deadlocked processes
- Abort one process at a time until deadlock cycle is eliminated

How to abort based on
- Priority
- Computation time
- Resources need / used → Select minimum Cost
- interactive / batch

Resource Preemption

→ eliminate deadlocks using resource preemption

3 issues need to be addressed
- Selecting a victim – which process & resources are to be preempted?
- Rollback – if preempt a resource, what should done with that process?
- Starvation – How do ensure that Starvation will not occur?

Selecting a Victim → determine the order of Preemption to minimize Cost

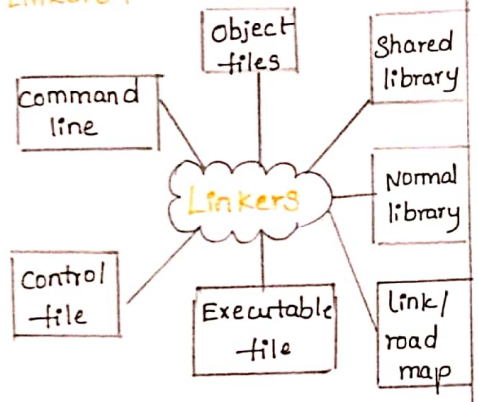
Rollback → roll back the process to Safe State and restart from that State

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

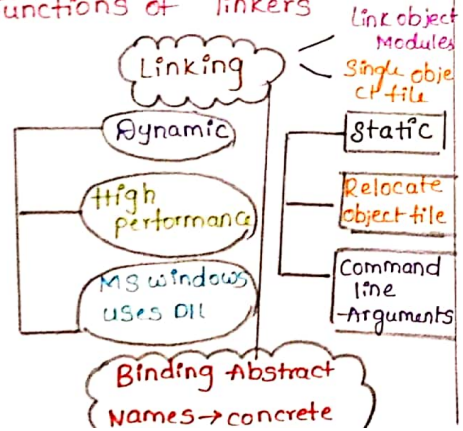
Linkers :-



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.

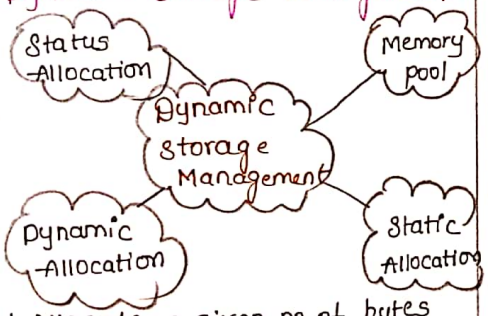
Dynamic linking :-

→ way to implement Dynamic linking - Jump table.
→ contains small function called when program items.
→ link library determies 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 :-

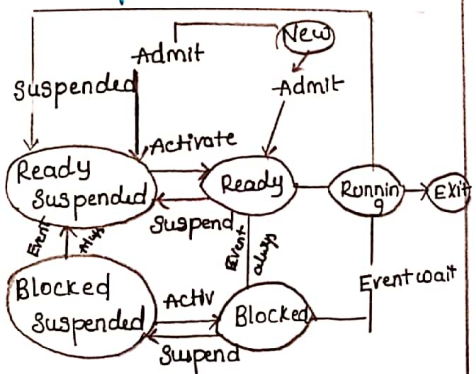
→ used when memory Association and freezing are partially predictable

→ status based organization keeps as 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.

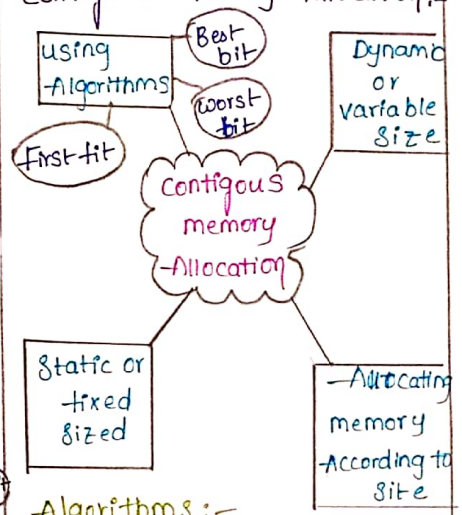
Swapping :-



→ A process need to be in memory to Execute
→ process can be swapped out temporarily out of memory
* From RAM ↔ HDD
→ It can be bought base to memory for Execution
HDD ↔ RAM

→ Swapping Execute the real physical memory of system.
→ Increase degree of multiprogramming in system

Contigous Memory Allocation :-



Algorithms :-

First fit - Start search at begning where previous search Ended
Best fit - Search for entire list
Worst fit - Search Entire list until it is sorted
→ Allocate the memory According to site
→ In static portroning main memory pre-divided into fixed size partitions.

Topic 10 : Paging

Paging :-

- memory management scheme
- Process to be non contiguous
- Paging
- eliminates contiguous allocation
- avoids external fragmentation
- need for compaction

Basic Methods :-

- **Pages** — Breaking logical memory into blocks of same size
- **Frames** — Breaking physical memory into fixed sized blocks

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

- **Address**
  - Logical address - CPU
    - Logical address space
  - Physical address - memory unit
    - Physical address space
- **Memory Management Unit**
  - hardware unit
  - mapping from virtual to physical address

Hardware Support :-

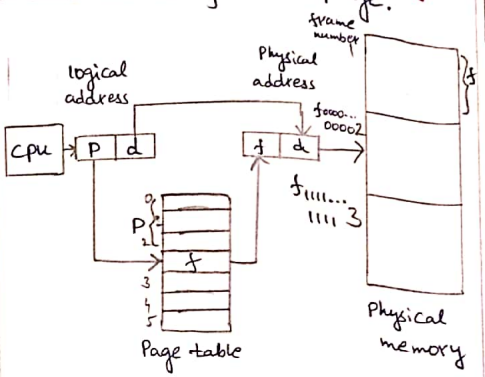
Logical address :

Page number (p)	Page offset (d)

Page Table :

* Page number used as index

* Base address of each page.



Paging Model of logical and Physical Memory :-

Page 0	
Page 1	
Page 2	
Page 3	

logical memory

0	1
1	4
2	3
3	7

Page table

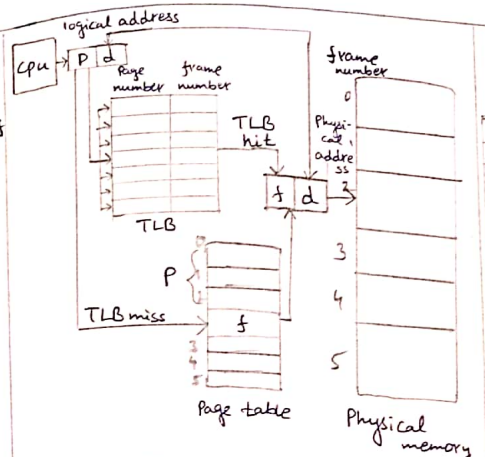
0	
1	Page 0
2	
3	Page 2
4	Page 1
5	
6	
7	Page 3

Physical memory

- **Hardware**
  - Registers
    - high speed logic to make paging
  - Translation look-aside buffer
    - fastlookup hardware cache, associative, key
    - Translation look-aside buffer miss
    - Page number is not in the TLB

Paging Hardware with the TLB :-

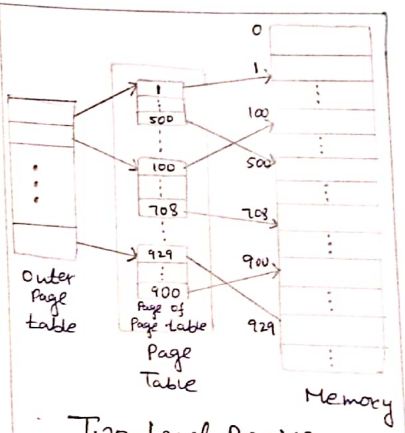
* Particular Page number is found in the TLB



- **Memory Protection**
  - accomplished Protection bits
  - Bits are kept in page table
- **Bits**
  - Valid bit — Page in process's logical address space
  - Invalid bit — Page is not in process's logical address space
- **Structure of Page Table**
  - Hierarchial Paging
    - two-level Paging algorithm
  - Hashed Page tables
    - hash value in virtual Page number
  - Inverted Page Tables
    - It has one entry

Page number		Page offset
P_1	P_2	d
10	10	12

*Translation works from the outer Page table



Two-level Paging

- **Advantages**
  - Virtual memory
  - time Sharing
  - Doesn't affect fragmentation
  - sharing of common code
- **Disadvantages**
  - Suffers Page breaks
  - Difficult to maintain Page
- **Application**
  - Windows based Paging Software
    - filter messages by cap code
    - message text
  - Create multiple watchdog alarms

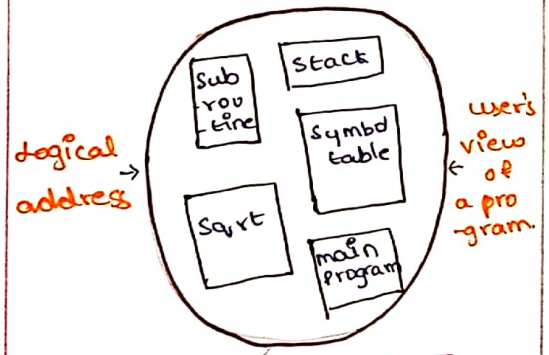
Topic 11:- Segmentation

Segmentation :-

Segmentation
- memory management scheme
- segment

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

Segment:
- logical address
- Name & length.



Logical address
- consists of a two tuple
- <segment-Number, off set>.

<segment-Number, off set>.

Hard ware
- Segment table
- 2-dimensional address
  - 1-dimensional sequence of bytes
- physical memory

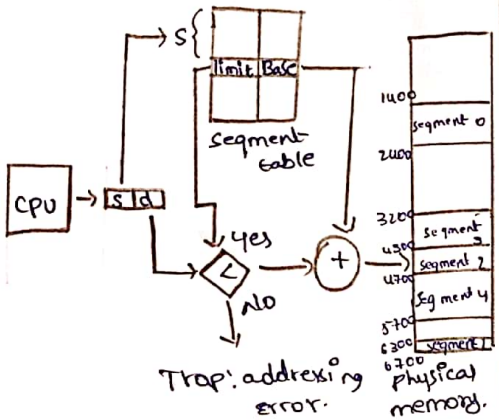
Segment table.

- 2 dimensional Logical address. into 1-dimensi -onal physical address
- Base addres
- limit

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

limit :- specifies the length of the segment.

Segment base
- physical address
- segment resides in memory



Segmentation hardware

Segment Number (s) :- No. of Bits required to represent the Segment.

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

Advantages of segmentation

- No internal fragmentation
- Consumes less space in Comparison to page table

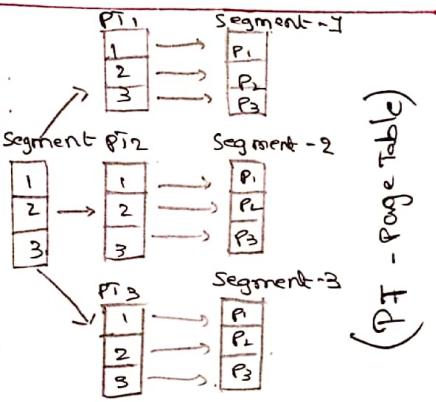
Disadvantages of Segmentation.

- Process are loaded & re -moved from the memory
- External fragmentat -ion.

Segmentation with paging :-

* Main Memory is divided into variable size segments
* segments are further divided
* Fixed sized

Segmentation with paging
- Segment No
  - appropriate Segment Number
- page No
  - exact page within the segment
- page off set
  - used as an off set with in the page.



Fragmentation.

- reduce wasted of memory
  - Small free hole
- holes can not be assigned
- achieve a degree of multi -programing

Fragmentation

- Internal Fragmentation
  - Memory block are allocated to the process more than their requested size
- External Fragmentation
  - free memory blocks, we can not assign it to process beca -use blocks are contiguous

Applications :-
- Stacks used by each thread
- C-compiler
  - The code
  - Global variables
  - Heap
- C library

TOPIC 12: VIRTUAL MEMORY

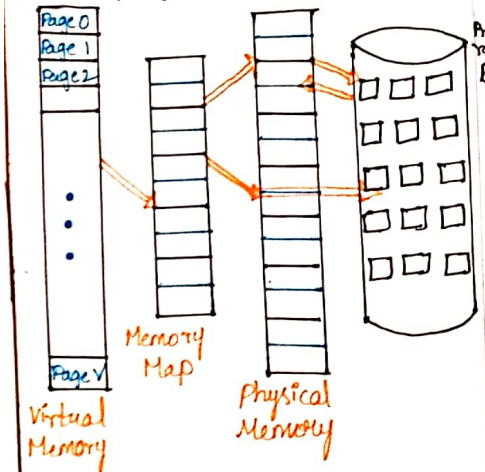
Virtual Memory:-

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:- Seperation of user logical memory from physical memory.

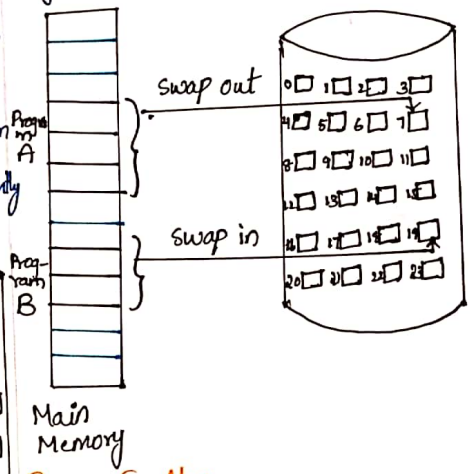
* Only part of the program needs to be in memory for execution.
* More Efficient process creation
* More program running Concurrently



Demand Paging:-

* Process of loading Pages.
* Pages are demanded.
* It works by Virtual Memory
* Primary (or) and Secondary (HD) Storages.
* Pages loads on demand
* Process removed and loaded
* Valid and Invalid Pages.

→ Lazy 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
* OS retrival 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:- 7 0 1 2 0 3 0 3 0 → Pages

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

*Page-Fault
* Page-Hit

→ LRU :- [least Recently used]

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

→ optimal : [later case]

Ex:- 7 0 1 2 0 3 0 7 → Pages

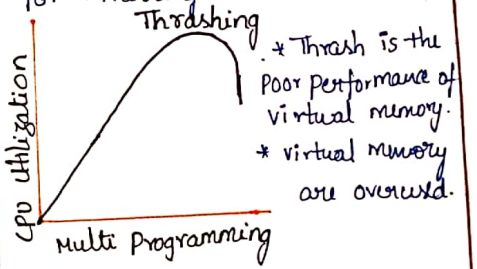
7	7	7	7	7	7	7	7	7
	0	0	0	0	0	0	0	0
		1	2	2	3	3	3	3
*	*	*	*	*	*	*	*	*

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.

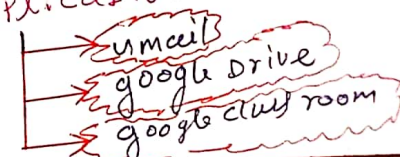


* Thrash is the Poor performance of virtual memory.
* virtual memory are overused.

DISK Devices:-

* Magnetic Disk - Data on magnetized media
* RAID - Redundant array of Independent Di
* CDROM - Compact disk Read only Memory.
* HDD - Hard disk drive.
* SDSD - Solid State drive.

Application:



Topic 13 FILE SYSTEMS

File Concepts

Files – Named collection of related information

File → Program → Source form
File → Program → Object form
File → data
File → Text file
File → Source file
File → executable file

File Attributes

Name | Identifier | Type | Location | Size | Protection

File Operations

Creation | Writing | Reading | Repositioning | Truncating | 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, doc
- library → lib, dll
- print/view → gif, pdf, jpg
- archive → rar, zip
- multimedia → mpeg, 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/Sequential | Direct | Indexed

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()
  → read(n)
  → write(n)
- Relative block number

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

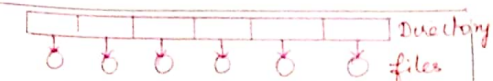
- Most visible aspect of OS.
- Online storage and access to data and programs.
- File System → Collection of files → Store related data
  File System → Directory structure → organize → provide information
- File protection – access and control by multiple users.

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

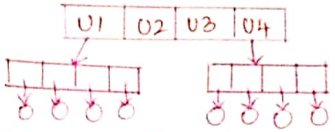


Limitation – when number of files increase.

Two level Directory

- Separate directory for each user

Levels → User file directory – Each user
→ Master file directory – overall system

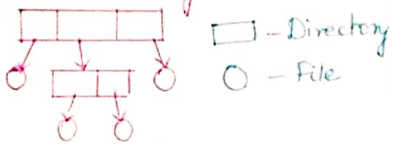


Limitation – Isolates users
- Difficult to cooperate

Tree Structured Directory

- Extend directory structure to a tree of arbitrary height.

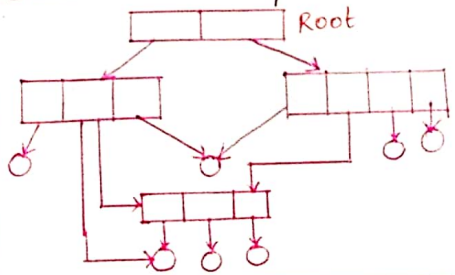
Directory – Subdirectory – files



- 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



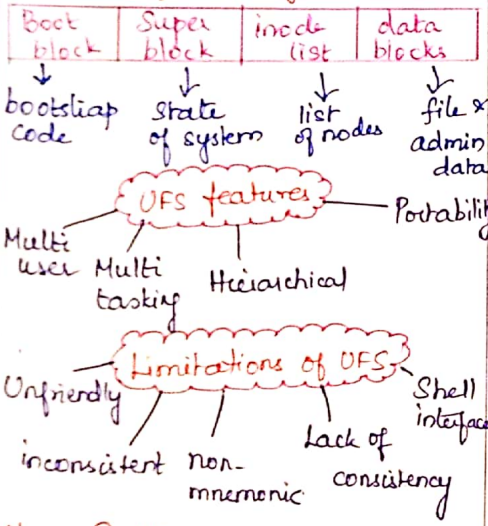
FILE SYSTEM

UNIX file system (UFS)

- Organize information.

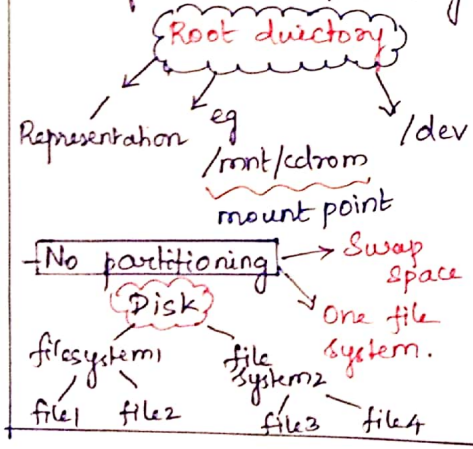
File types
- Ordinary files — data — text
- Directories — organize files
- special files — Device I/O
- Pipes — Temporary storage
- Sockets — Process communication
- Symbolic links — reference to other files

Unix file system layout:



Unix Partitioning:

- all files in one hierarchy



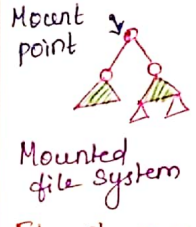
Mounting and Unmounting:

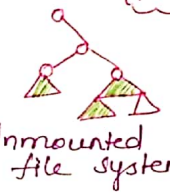
- File mounted before usage.

Commands used

Mount
- Mounts a storage device file system

Unmount
- Remove entry from mount tab file
- Deletes file from mount point





File Sharing:

Multiple Users

Owner
- change attributes
- grant access
- User identifiers

Group
- share access to a file
- Group identifiers

Remote file Systems

- Client server model
- Distributed Information Systems

UNIX Semantics

- Writes to open file visible immediately.
- Sharing the pointer of current location to the file

Buffer Cache:

file system → Buffer cache → Kernel I/O prefetching → Kernel I/O clustering → Disk Driver → Disk

Buffer cache
- catch Metadata
- flush dirty buffers
- Keeps a copy of parts of disk
- prevent disk transfers
- call sync 30s once

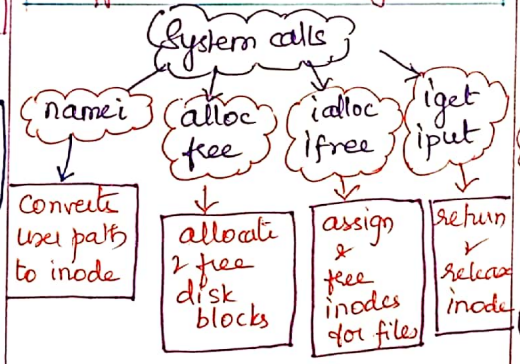
inodes:

- Data Structure
- file system object
- Stores attributes
- file serial number

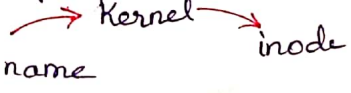
Lower level file system algorithms

namei iget iput bmap	alloc free	ialloc ifree

Buffer allocation algorithms

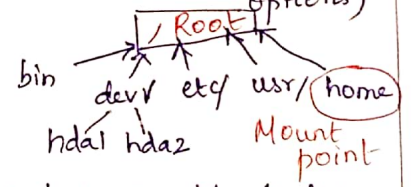


- Access files by system calls
- Specify file by character string



Mounting file systems - Example

mount("/dev/hda2" "/home", options)



File system on /dev/hda2

File System implementation

Type	Mode
User Id	Group Id
Size	nblocks
nlinks	Flags
Direct blocks	→ data → data → data
Single Indirect	
Double Indirect	
Triple Indirect	

Generic File System Layer

Buffer Cache

Raw Block I/O

Device Driver	Device Driver

Common Internet file System (Application of Unix)

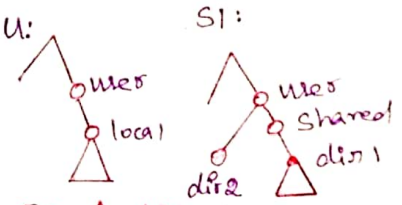
- Standard for computers to view a distant file
- Supported by Unix, Linux, Windows
- Internet systems based file sharing

* Ability to access multiple files at once.
* File sharing, file locking, file caching — characteristics

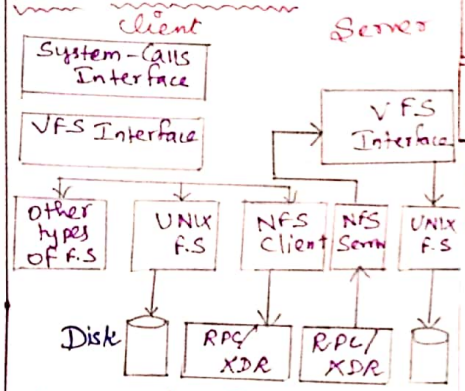
TOPIC 15: Network File Systems, I/O Systems

NFS:

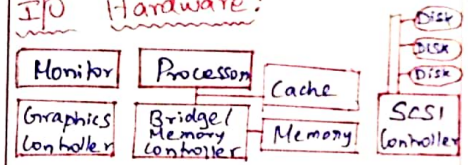
* Share files between users on different distributed file System



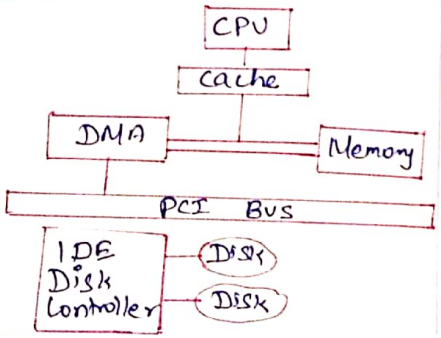
NFS Architecture:



I/O Hardware:



DMA:

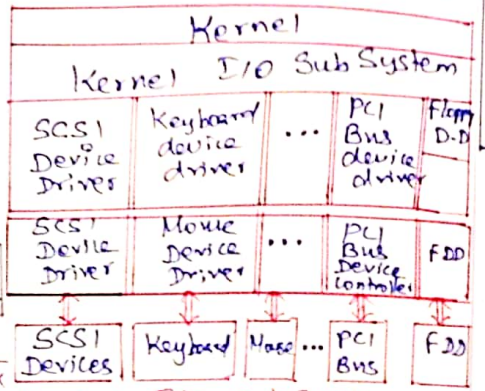


Application I/O Interface:

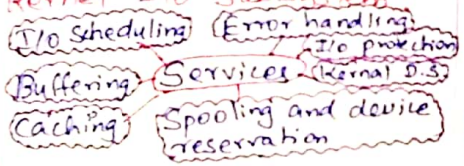
* Interface between application programs and I/O Devices.

Abstraction → Encapsulation → S/W layering

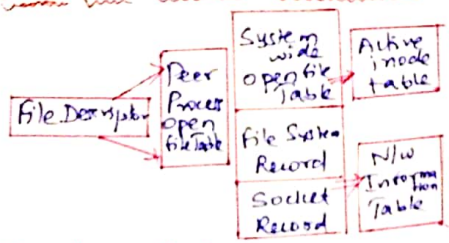
* Hide the differences among device controller → I/O.



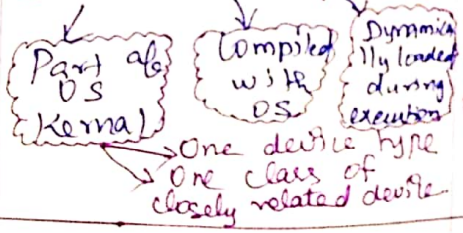
Kernel I/O SubSystem:



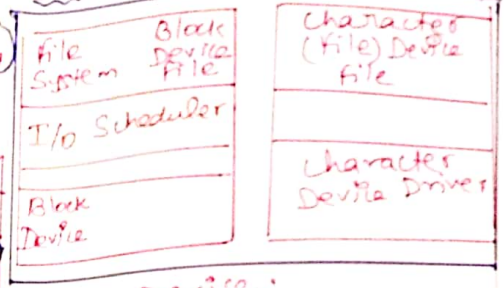
UNIX IO Kernel Structure:



Device Drivers:

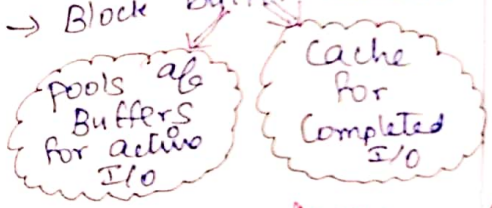


Device Driver Block Structure:



Block Device:

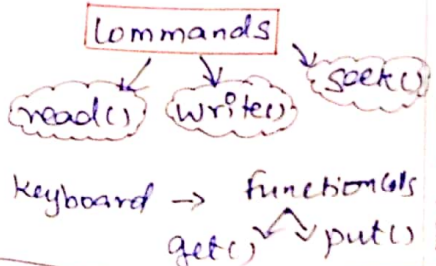
→ Main Interface to all disk device.
→ Block buffer cache.



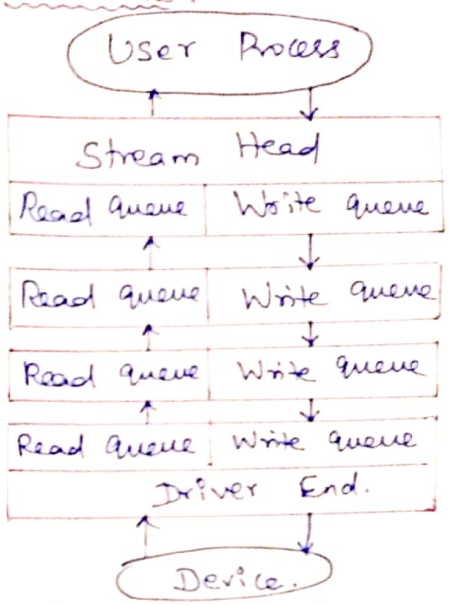
Character Devices:-

- Register set of functions
- Implements drivers
- Performs various file I/O operations

Block and character Devices:-



Streams:-

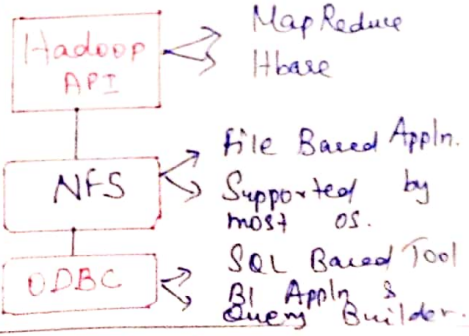


Stream Structure

Character and Block Device Switch tables:

→ Arrays of operational pointers.
→ Sub array of Jump pointers.

Real Time Implementation:



Topic 16: Disk Scheduling

Disk Scheduling Algorithms:

- Effectively utilizing disk drives.
- Fast access time and disk bandwidth.
- Minimizes seek time and distance.

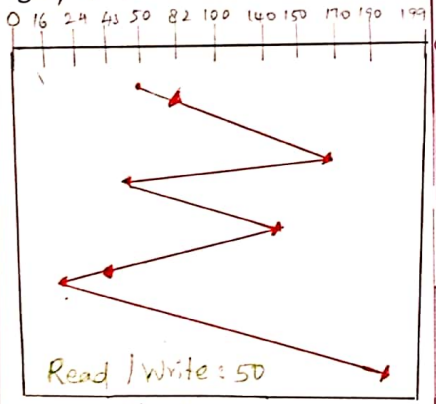
Algorithms:

FCFS, SSTF, SCAN, CSCAN, LOOK, C-LOOK

FCFS: First come first serve

* Simplest Algorithm
* Request ordered in disk queue.

Ex: Order of Request (82, 170, 43, 140, 24, 16, 190)

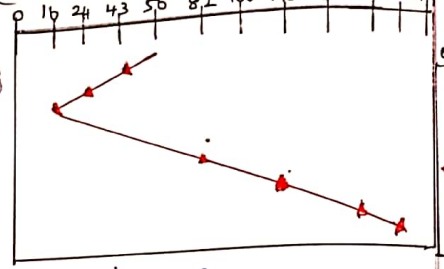


Seek time:
= (82-50) + (170-82) + (170-43) + (140-43) + (140-24) + (24-16) + (190-16)
⇒ Total head Movement = 642 cylinders

SSTF: Shortest Seek time first.

* Request near to disk execute first.

Ex: Order of Request (82, 170, 43, 140, 24, 16, 190)

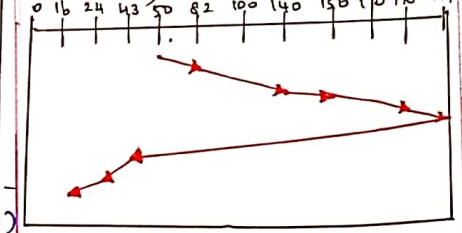


Seek time:
= (50-43) + (43-24) + (24-16) + (82-16) + (140-82) + (170-140) + (190-170)
⇒ Total head Movement = 208 cylinders

SCAN:

* Starts at one end of the disk.
* Reverse after reaching

Ex: 82, 170, 43, 140, 24, 16, 190, 150

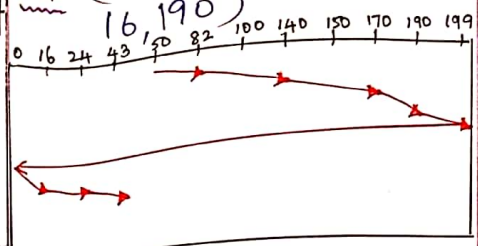


Seek time:
= (199-50) + (199-16)
= 332 cylinders

CSCAN:

* Reverse of SCAN
→ Scan the path again after scanned first time

Ex: (82, 170, 140, 43, 24, 16, 190)

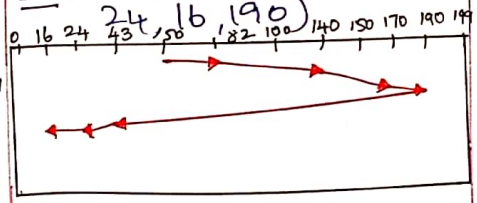


Seek time:
= (199-50) + (199-0)
= (43-0)
= 391 cylinders

LOOK:

→ Similar to SCAN disk
→ Prevent extra time

Ex: (82, 170, 43, 140, 24, 16, 190)



Seek time:
= (190-50) + (190-16)
= 140 + 174
= 314 cylinders

Total head Movement = 314 cylinders.

Disk Management:

Disk formatting → Dividing disk into sectors
Boot Block → Initial Boot strap program.
Bad Block → Sector Sparing, Sector Slipping

RAID:

- Level 0 → Non redundant Stripping
- Level 1 → Disk Mirroring
- Level 2 → Memory Style error correcting code.
- Level 3 → Bit Interleaved parity organization
- Level 4 → Block Interleaved parity organization
- Level 5 → Block Interleaved distributed Parity.
- Level 6 → P + Q Redundancy Scheme.
- Level 0 + 1 → Combination of level 0 and 1
  - 0 ⇒ Performance
  - 1 ⇒ Reliability.

Application of Disk Scheduling in Multimedia:

* DMS → Deadline Modification Scan Algorithm.
* Automatically decide Suitable Task group for Seek Optimization.
* Increase throughput to support more task.

TOPIC 17: LINUX SYSTEM

History

- Modern, free OS based UNIX standards
- Design Goal of UNIX compatibility
- Reliable on common PC Hardware
- Varying Linux Distributions

Kernels → Odd version number - DEVELOPMENT KERNEL
Even version number - PRODUCTION KERNEL

Applications

Management tools

FILE SYSTEM HIERARCHY STANDARD :-

Maintained by Linux to ensure compatability

Standard set of package

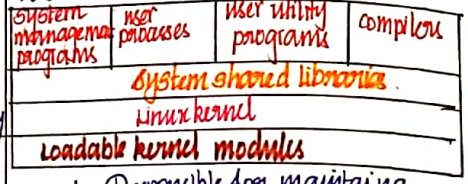
LINUX DISTRIBUTIONS

- RedHat, Debain
- Canonical & SuSE
- General Public License (GPL)
- Installation, Management utilities & Ready to Install packages.

DESIGN PRINCIPLES:

* Multitasking, Multiuser
* Traditional UNIX semantics
* Goals are Speed, Efficiency
* Designed to complaint with POSIX documents.

COMPONENTS OF LINUX SYSTEMS



Kernel :- Responsible for maintaing abstractions

System libraries :- Standard set of functions

System utilities :- Perform individual specialized management tasks.

COMPONENTS TO LINUX MODULE SUPPORT

- Module-Management System
- Module Loader & Unloader
- Driver-registration System
- Conflict-Resolution mechanism

REGISTRATION TABLES

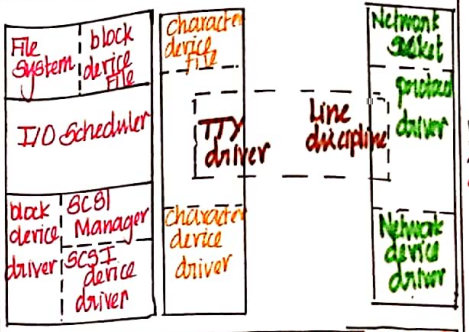
- Device Drivers
- File Systems
- Network protocols
- Binary Format

Linux virtual File system (VFS)

- Inode Object - Represent an Individual file
- File object - Represent an open file
- Superblock object - Represent an entire file system
- Dentry object - Represent Individual entry.

BLOCK DEVICES

- Acts as pool of buffers for active I/O
- Serves as a cache for completed I/O



DEVICE-DRIVER BLOCK STRUCTURE

Character device
- Set of functions to implement I/O operations
- No preprocessing
- Line Discipline is an Interepter
- Pipe mechanism allows child process to communicate.

NETWORK STRUCTURE

- Socket Interface
- Protocol Drivers
- Network Device Drivers

Packets also Pass Firewall management for filtering

LINUX SYSTEM ADMINISTRATION

→ User, Groups & Access Permissions:

r - Read a file
w - Write a file
x - Execute a file.

THREE DEFINED CATEGORIES OF USERS :- OWNER, GROUP & OTHERS

CHMOD COMMAND
- Symbolic Mode
- Absolute Mode

LINUX 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 Systems & Extensive knowledge like Python, PHP or Ruby.
- Experience Handling Mails
- Experience dealing with virtual Machines and containers
- Requires Analytical and Structural way of thinking.

ROLES OF ADMINISTRATOR :

- Defining user/access rights & manage
- Identify any problems & find solutions
- Restore smooth operations
- Functionality of Everyday Applications
- Manages Network Servers
- Implement Hardware/Software update
- Provide Technical Assistance
- Maintain System Security.

BEST LINUX DISTRIBUTIONS

- Ubuntu
- Debain
- CentOS
- Fedora
- RedHat

CASE STUDY - LINUX SETUP

TOPIC 18 : SETTING UP LINUX MULTIFUNCTION SERVER

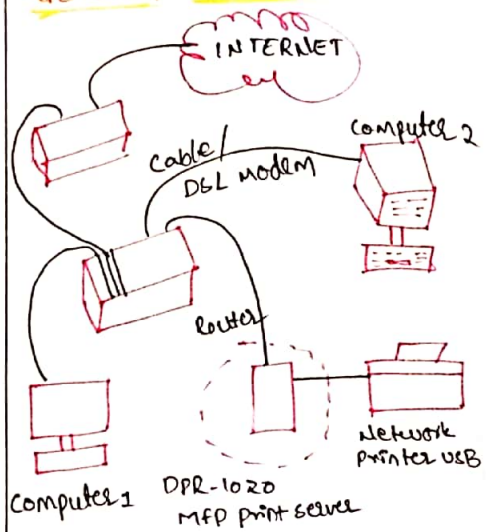
SETTING UP A LINUX MULTIFUNCTION SERVER HARDWARE INSTALLATION

- 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 pannel.
- print servers self-test will proceed.

POWER ON SELF-TEST

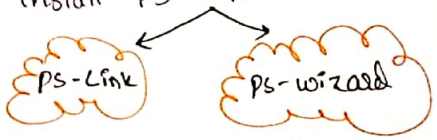
- DPR-1020 powered ON, automatically perform self-test.
- During the time of LED test may causes error.
- Immediately follow the LED Tests.

GETTING STARTED



AUTO RUN INSTALLATION :-

- Insert CD into CD-Rom drive
- initiate auto-run program
- install PS softwall.



- configuel MFP server
  - * change IP address
  - * support the Multi function (print / scan / copy / fax)
  - * Easily add a printer to your computer
- view Quick installation guide
- view manual.
- install acrubal reader
- Exit - close the Auto-run program

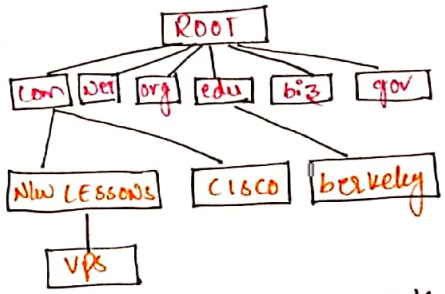
DNS → Domain Naming system

- DNS is the hierarchical and decentralized naming system.
- DNS server converts URLS and domain names into IP
- Transfer user typing into medicine
- find a web page.

Hierarchy of Name servers

Name server [Directory of IP address]

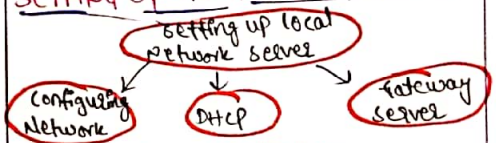
- helps IP addresses into domain names.



- Server component of the DNS.
- * Two principle name spaces:-
  - → Translation (resolution) of human-memoriable domain names.
- host names into the corresponding numeric IP addresses.

* ROOT NAME SERVER:-
  - → Root zone of the DNS.
* TLD NAME SERVER:- (TOP-LEVEL DOMAIN)
  - → Highest level of domain names in the root zone of the DNS.
* AUTHORITATIVE NAME SERVER:-
  - → provides actual answer to your DNN queries such as
  - * mail server IP addresses
  - * recursive - IP addresses
* CHACHING NAME SERVER:-
  - → is a server handles requests from clients.

SETTING UP LOCAL NETWORK SERVICES



CONFIGURING NETWORK:-

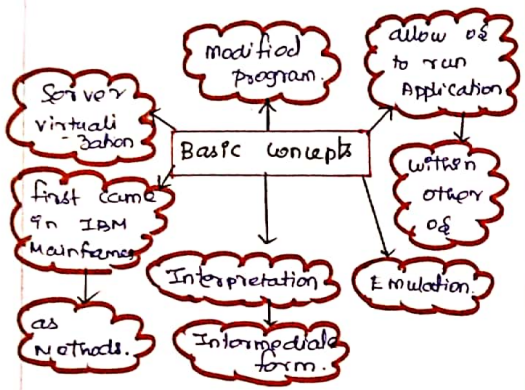
- DHCP is used during the debian installation, then configure the server with a static IP addresses
  - ex:- Debian

DHCP → Dynamic HOST CONFIGURATION PROTOCOL

- it is used to solve a number of problems associated with local environment.

TOPIC 14 : VIRTUALIZATION

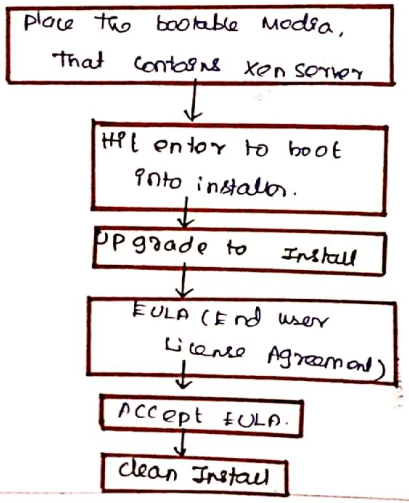
VIRTUALIZATION

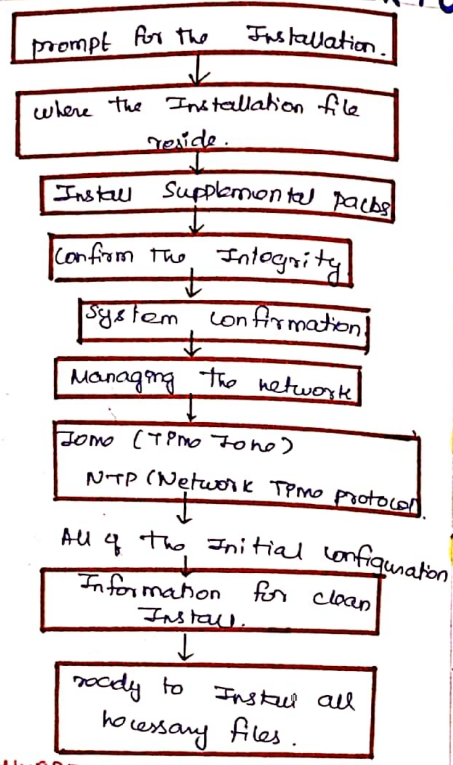


BASIC CONCEPTS

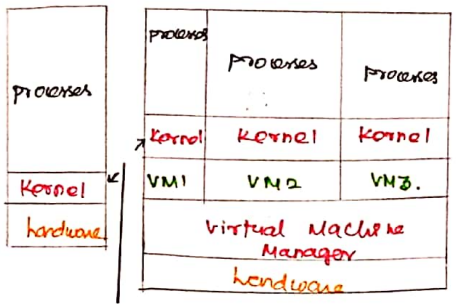
- Modified program. (form) than normal os.
- different users can operate it's end use different applications.
- virtual hosting environment.
- providing security to the hardware
- separate Applications into containers.

SETTING UP XEN





VMWARE

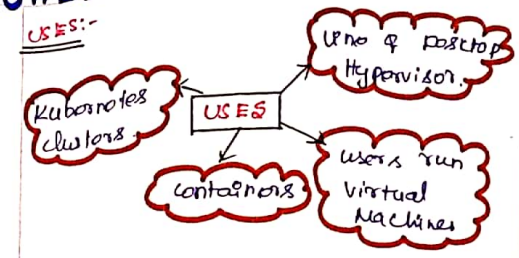


Programming Interface,

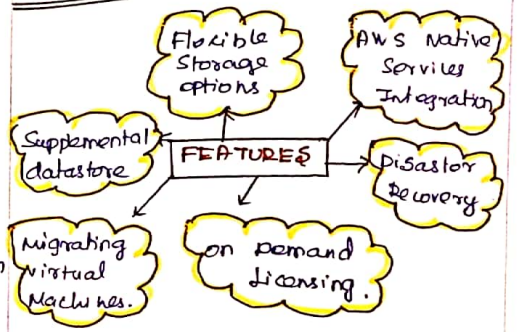
VM WARE.

- run each applications on it's own os.
- Make better use of physical servers on cpu.
- More efficient uses of space and energy.

USES:-



FEATURES:-



TYPES:-

- two different types.
  - process VMs → gives us complete System platform.
  - System VMs → virtual environment of os.

LINUX HOST

- finding IP address of domain name
- performs reverse lookups.
- finding domain name associated with IP address.

Command:-

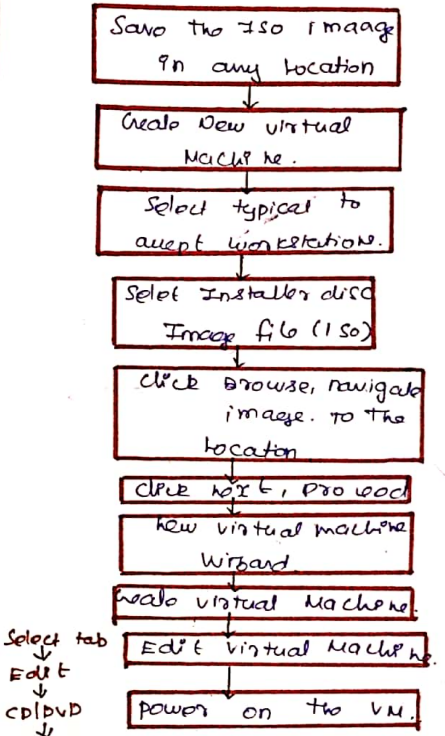
- user for DNS lookup operations.
- Identify the IP address of a particular domain name.
- find more Specific details.

SYNTAX:-

host [-a dbir TMV] [-c class] [-end0is] [-w time] [-R number] [-m flag] hostname [server].

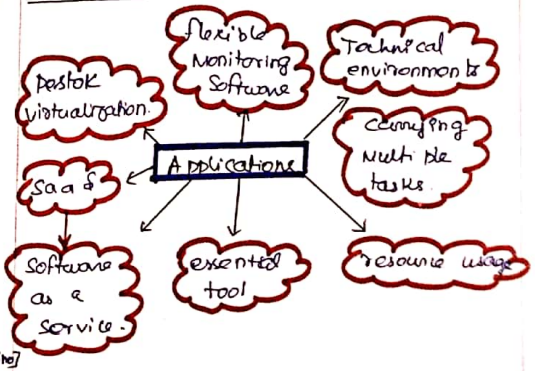
ADDING GUEST OS.

- To Install the OS from an Iso Image.



Select tab → Edit → CD|DVD → Right side → power on → Iso Image file → Browse → OK.

REAL TIME APPLICATIONS.



TOPIC 20 : ADVANCES IN OPERATING SYSTEMS

MOTIVATION :-

* Traditional Operating Systems ran on stand-alone computers with single processors
* High demand and popularity
* Gained Importance

Policies — "WHAT SHOULD BE DONE"
Mechanisms — "HOW IT SHOULD BE DONE"

POPULAR OPERATING SYSTEMS :-

1. Linux
2. BlackBerry
3. MS DOS
4. MacOS
5. ubuntu
6. Apple
7. Microsoft Windows
8. Android
9. EssayCorp

Concurrency in OS
- User Interface
- Resource Management
- Dedicated
- Opensource Licensed

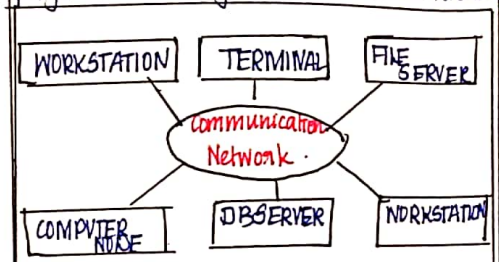
Design Approach
- LAYERED APPROACH
- KERNEL APPROACH
- VIRTUAL MACHINE APPROACH

TYPES :

Advanced Operating System
- Architecture driven
  - Distributed System
  - Multi-processor System
- Application Driven
  - Database Systems
  - Real time Systems

DISTRIBUTED OPERATING SYSTEMS

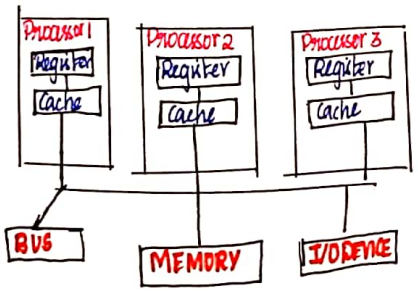
→ Set of processors that shares set of physical memory block over interconnection



→ These are connected through communication Network.

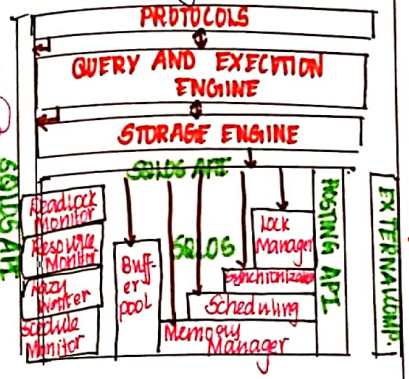
MULTIPROCESSORS SYSTEMS :-

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



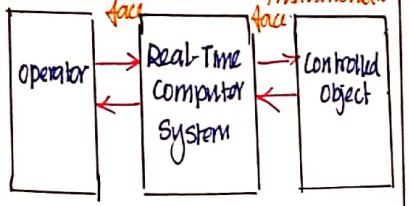
DATABASE SYSTEMS :-

→ Transaction, Operation to store, retrieve and Manipulate data efficiently.



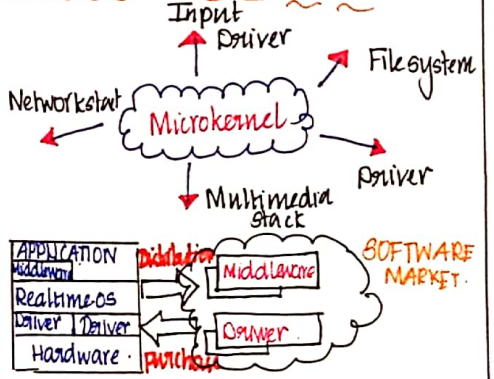
REAL-TIME OPERATING SYSTEMS :-

→ Major issue is design in scheduling Jobs & designing languages and primitives.



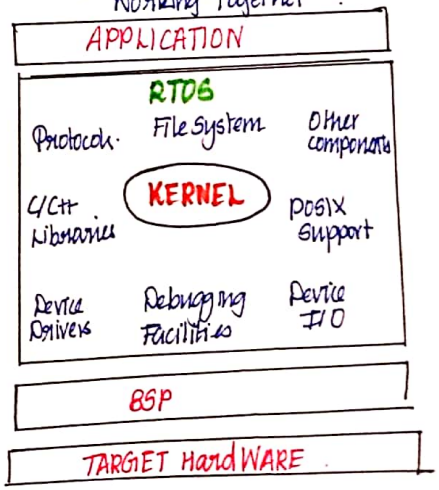
Types of Realtime OS
- Hard Realtime OS
- Soft Real time OS
- Firm Real time OS

CHANGING TRENDS IN OS.



THE RIGHT REALTIME OPERATING SYSTEM.

- Requirements Review
- Availability Portability
- Support Functions
- Evaluate Alternatives
- Being Future Proof.
- Existing Internal Experience
- Support Partnership & Working Together.



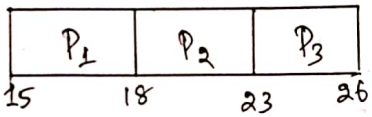
1) FIRST COME FIRST SERVED

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

Process Name	Arrival Time	Burst Time (ms)
P_1	15	3
P_2	17	5
P_3	19	8

Solution:-

Gantt Chart



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

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

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

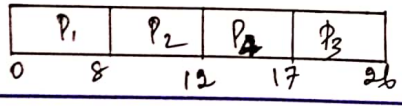
$= \frac{3+6+7}{3} = 5.33$ ms

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_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Solution:-



Non-preemptive SJF:-

Waiting Time = Starting Time − Arrival Time

Average Waiting Time

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

$= (0+7+15+9)/4 = 7.75$ ms

Average Turnaround Time

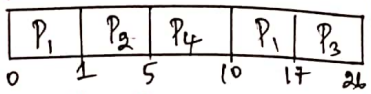
$= \frac{(8+0)+(4+7)+(9+15)+(5+9)}{4}$

$= 14.25$ ms

Turn around time = Burst time + waiting time

Preemptive SJF:-

Gantt Chart



Average waiting Time

$= \frac{(10-0-1)+(1-1)+(17-2)+(5-3)}{4}$

$= (9+0+15+2)/4 = 6.5$ ms

Average Turnaround Time

$= \frac{(8+9)+(4+0)+(9+15)+(5+2)}{4}$

$= (17+4+24+7)/4 = 13$ ms

Preemptive algorithm - process can be interrupted even before completion

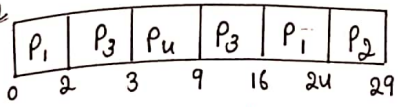
Non-preemptive process not interrupted until completion.

3) PRIORITY SCHEDULING:-

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

Process Name	Arrival time	Burst time	priority
P_1	0	10	6
P_2	1	5	9
P_3	2	8	4
P_4	3	2	2

Sol



Lowest order preemptive Scheduling

Average waiting time

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

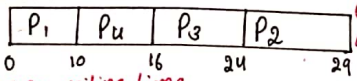
$= \frac{(14+23+6+0)}{4} = 10.75$ ms

Average turnaround time

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

Lowest order non-preemptive scheduling

Solution:-



Gantt chart

Average waiting time

$= ((0-0)+(24-1)+(16-2)+(10-3))/4 = 13.5$ ms

Average turnaround time

$= (10+0)+(5+23)+(8+14)+(6+7)/4$

$= 18.25$ ms

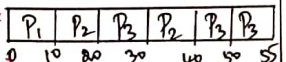
4) ROUND ROBIN SCHEDULING

process	Arrival time	Burst time
P_1	0	10
P_2	1	20
P_3	2	25

Avg. waiting time

$= 0+(30-1-10)+(40-2-10)$ / 3

$= 15.67$ ms



Avg. Turn Around time

$= (10+39+53)/3$

$= 34$ ms

* preemptive
* Each process provided fixed time to execute

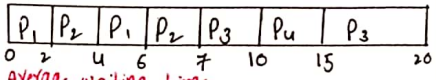
5) Multilevel Queue scheduling

* Process permanently assigned to a Queue

process	Arrival time	Burst time	Queue Number
P_1	0	4	1
P_2	0	3	1
P_3	0	8	2
P_4	10	5	1

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

Solution:-



Average waiting time

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

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

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

$WTP_4 = 10-10 \Rightarrow 0$

Average $= \frac{2+4+12+0}{4} = \frac{6+12}{4} = 4.5$ 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 - 2 sec

  process interrupted 4 times & complete on Queue 5
* Algorithm Allows process to move between Queue

Scheduling criteria

- Maximise
  - CPU utilization
  - Throughput
- Minimise
  - Turnaround time
  - waiting time
  - Response time

1) 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 algorithms and find Completion Time (CT), Turn around Time (TAT), Waiting Time (WT) and Response Time (RT)

1) FCFS (First Come First Serve)
2) Shortest Job First
3) Round Robin with Quantum time 4
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 algorithms and find Completion time (CT), Turn around time (TAT), waiting Time (WT) and Response Time (RT)

1) FCFS (First Come first Serve)
2) SJF (Shortest Job First)
3) Round Robin with Quantum time 4
4) Priority Scheduling

3) Consider the following Processes with given Burst Time and Arrival Time

Process ID	Arrival 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 listed above which are produces least average waiting Time.

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

Process Queue	Burst Time	Arrival Time
P_1	6	2
P_2	2	5
P_3	8	1
P_4	3	0
P_5	4	4

Calculate Average waiting Time of the above processes if SJF Non-preemptive scheduling is followed.

5) Consider the following processes with Burst time (BT) and Arrival time (AT).

Process Queue	Burst Time	Arrival Time
P_1	6	2
P_2	2	5
P_3	8	1
P_4	3	0
P_5	4	4

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

Deadlock avoidance using Banker's Algorithm

DAY 2

Consider a system with 5 processes P_0 through P_4 and 3 Resource types A, B, C.

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

find out if the system is in a safe state for the following sequence $\langle P_1, P_3, P_4, P_0, P_2 \rangle$

→ Calculate Need Matrix

$$Need[i,j] = Max[i,j] - Allocation[i,j]$$

Process	Allocation A B C	Max A B C	Need A B C
P_0	0 1 0	7 5 3	7 4 3
P_1	2 0 0	3 2 2	1 2 2
P_2	3 0 2	9 0 2	6 0 0
P_3	2 1 1	2 2 2	0 1 1
P_4	0 0 2	4 3 3	4 3 1

Initially work = Available = 3 3 2

Finish(i) = false for all $i = 0, \dots 4$

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

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

Check the sequence $\langle P_1, P_3, P_4, P_0, P_2 \rangle$

for P_1 Check if $need_{P_1} \le work$

Step 2: $(1\ 2\ 2) \le 3\ 3\ 2 \Rightarrow$ True

Step 3: work = work + Allocation$_{P_1}$

= 3 3 2 + 2 0 0

= 5 3 2

finish(P_1) = True got step 2

for P_3 check if finish(P_3) = false and

Step 2: $Need_{P_3} \le work$

$0\ 1\ 1 \le 5\ 3\ 2 \Rightarrow$ True

Step 3: work = work + Allocation$_{P_3}$

= 5 3 2 + 2 1 1

= 7 4 3

finish(P_3) = True got step 2

for P_4

Step 2: check if finish(P_4) = false and

$Need_{P_4} \le work$

finish(P_4) = false & $4\ 3\ 1 \le 7\ 4\ 3 \Rightarrow$ True

Step 3: work = work + Allocation$_{P_4}$

= 7 4 3 + 0 0 2

= 7 4 5

finish(P_4) = True got step 2

for P_0

Step 2: if finish(P_0) = false & $Need_{P_0} \le work$

finish(P_0) = false & $4\ 3\ 1 \le 7\ 4\ 5 \Rightarrow$ True

Step 3: work = work + Allocation$_{P_0}$

= 7 4 5 + 0 1 0

= 7 5 5

finish(P_0) = True go to Step 2

for P_2

Step 2: finish(P_2) = false & $need_{P_2} \le work$

finish(P_2) = false & $6\ 0\ 0 < 7\ 5\ 5 \Rightarrow$ True

Step 3: work = work + Allocation$_{P_2}$

= 7 5 5 + 3 0 2

= 10 5 7

finish(P_2) = True

A B C

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

Resource Request Algorithm

→ need to check the request can be granted or not

find out if P_1 request (1, 0, 2) be granted or not

$Request_1$ (A B C) (1, 0, 2)

Step 1: $request_1 \le Need_1$, goto step 2

else error condition

$1\ 0\ 2 \le 1\ 2\ 2 \Rightarrow$ True goto step 2

Step 2: $request_1 \le Available$, goto step 3

else P_1 must wait

$1\ 0\ 2 \le 3\ 3\ 2 \Rightarrow$ True goto step 3

Available = Available − request$_1$

= 3 3 2 − 1 0 2

= 2 3 0

Allocation$_1$ = Allocation$_1$ + request$_1$

= 2 0 0 + 1 0 2

= 3 0 2

Need$_1$ = Need$_1$ − request$_1$

= 1 2 2 − 1 0 2

= 0 2 0

Now update the Allocation & Need matrix of Previous

Process	Allocation A B C	Max A B C	Need A B C
P_0	0 1 0	7 5 3	7 4 3
P_1	3 0 2	3 2 2	0 2 0
P_2	3 0 2	9 0 2	6 0 0
P_3	2 1 1	2 2 2	0 1 1
P_4	0 0 2	4 3 3	4 3 1

work = Available = 2 3 0

Now have to check the sequence $\langle P_1, P_3, P_4, P_0, P_2 \rangle$ using Safety algorithm

if finish(i) = True for all $i = 0 \dots 4$ process in the system, then the system process P_1 request can be granted immediately

P_1 request 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 alloc 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 process will finish LAST?

process	Alloc			Request		
	X	Y	Z	X	Y	Z
P_0	1	2	1	1	0	3
P_1	2	0	1	0	1	2
P_2	2	2	1	1	2	0

1. P_0
2. P_1
3. P_2
4. None of the above since the system is in a deadlock

Problem - 02

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_0, P_1 and P_2. 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_0	0	0	1	8	4	3
P_1	3	2	0	6	2	0
P_2	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_0 requests 0 units of X, 0 units of Y and 2 units of Z

REQ2 : P_1 requests 2 units of X, 0 units of Y and 0 units of Z Which of the following is True?

1. Only REQ1 can be permitted
2. Only REQ2 can be permitted
3. Both REQ1 and REQ2 can be permitted
4. Neither REQ1 nor REQ2 can be permitted

Problem - 03

Consider the following Snapshot of a system. Answer the following questions using the Banker's Algorithm

Process	Allocation			Max		
	X	Y	Z	X	Y	Z
P_0	0	0	1	8	4	3
P_1	3	2	0	6	2	0
P_2	2	1	1	3	3	3

i) What is the content of the matrix Need?
ii) Is the system in a safe state?
iii) If a request from process P_1 arrives for (0,4,2,0) can the request be granted immediately?

Problem - 04

Process	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

Solve using Banker's Algorithm
1. What is the content of need matrix?
2. Check if the system is in a safe state?
3. Determine the total sum of each type of resource?

PAGING & SEGMENTATION

PAGING:-

1) Consider a paging system with the page table stored in Memory.

a) If a memory reference takes 50 nanoseconds, 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

Solution:-

a) Given

Time for Memory reference takes place = 50 nS.

Time for paged Memory reference.

= Time for accessing table + time for accessing the target data in Memory.

= 50 nS + 50 nS

= 100 nS.

b) Assume that finding a page table entry in the TLB takes 2 nS, if the entry is present.

The effective Memory reference time

$time_{avg} = 0.75 * (50nS + 20ns) + 0.25 * (2ns + 50ns + 50ns)$

$= 64.5\ ns$.

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

Solution.

Answer size = 2^n = 1KB = 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.

Important Formulas:-
* size of page table = No. of entries in page table x page table entry size

a) 3085

Decimal = 3085

Binary = 000000110000001101

Page # = 011 = 3

page offset = 000000 1101 = 13.

b) 42095

Decimal = 42095.

Binary = 1010010000 1101111

Page # = 101001 = 41

Page offset = 0000 1101111 = 11.

c) 215201

Decimal = 215201, Binary = 110100100010100001

page # = 11010010 = 210

Page offset = 0010100001 = 161.

d) 650000

Decimal = 650000

Binary = 10011110101100010000

Page # = 1001111010 = 634

Page offset = 1100010000 = 784.

SEGMENTATION.

1) Consider the following segment table

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

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.

Important Formulas.
* Physical address space = Size of main memory
* Frame size = page size
* Size of main memory = Total no. of frames x page size
* page size = 2^x
  x → no of bits in page offset

Solution:-

Segment — Logical Address — Offset

a) 0, 430.

physical address = Base + logical Address offset

= 219 + 430 (600 > 430 legal)

= 649. (649 > 600 illegal ref)

b) 1, 10

physical address = 2300 + 10 (14 > 10 legal)

= 2310 (2310 > 14 legal ref)

c) 2, 500

Physical address = 90 + 500

(100 > 500) ✗ illegal ref.

d) 3, 400

Physical address = 1327 + 400 (580 > 400 legal)

= 1727.

e) 4, 112.

Illegal, since the size of segment 4 is 96, and the offset of logical address is 112.

∴ it is Illegal.

2) Consider a logical address space of 256 pages with 4KB 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 (LS size) = 2^m.

Logical address space (LS size)

= # of pages * page size.

= 256 x 4KB

= 256 x 4096 (4 x 1024)

= 1048576.

= 2^{20}

∴ $2^m = 2^{20}$

∴ m = 20 //.

b) Let x be the no. of physical Addresses

physical address space (PS size) = 2

Physical address space (PS size)

= # of frame x frame size

= 64 x 4KB

= 64 x 4096 (4 x 1024)

= $2^6 \times 2^{12}$

= 2^{18}

∴ $2^x = 2^{18}$

x = 18 //.

∴ Number of required bits in Physical address = x = 18 bit

TUTORIAL DAY 2

PROBLEMS:-

Important formulas:-

For main memory:-

* Physical address = Size of Main Memory.

* Size of Main memory = Total number of frames × Page Size.

* If number of frames in main memory = 2^x. then the number of bits in frame number = x bits.

* If page size = 2^y bytes, then the number of bits in page offset = x bits.

* If the Size of Main Memory = 2^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 = 2^y bytes," then the number of bits in virtual address space = x bits.

NOTE:-

* "n bits → 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.

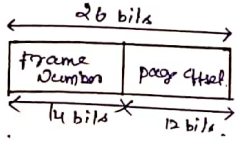
3) Calculate the number of bits required in the Address for memory having size of 16 GB. Assume the Memory is 4-byte addressable.

4) Calculate the size of Memory of M's 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 Mega bytes).

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?

1. 16 Mb
2. 8 Mb
3. 2 Mb
4. 24 Mb.



7) In a virtual Memory system, size of virtual address is 32 bit, the size of physical address is 30 bit, page size 4 Kbyte and size of each page table is (entry) 32 bit. The main Memory is byte addressable. find the Maximum no. of bits can be used to storing Protection and other information in table entry?

SEGMENTATION:-

PROBLEM:-

1) 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, 430
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 address spaces contain 2^{16} 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 space } = process size

* Physical Address space } = Main memory size

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

PAGE REPLACEMENT ALGORITHMS

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 bought into memory.
* Performance not always good.
* may suffer from Belady's Anamoly.

LRU page Replacement

* Replace page not been Used long period of time
* Looking backward in Time
* Did not suffer from Belady's Anomoly.

Optimal Page Replacement

* Lowest page fault rate
* Never suffer from Belady's Anamoly.
* 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	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
		1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
x	x	x	x	✓	x	x	x	x	x	x	✓	✓	y	x	✓	✓	x	x	x
F	F	F	F	H	F	F	F	F	F	F	H	H	F	F	H	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	3	3	3	3	3	3	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
x	x	x	x	✓	x	✓	x	x	x	x	✓	✓	y	✓	x	✓	x	✓	✓

x = page fault ✓ = Page Hit

Page faults = 12 Page hits = 8

* Replace page not used for long period of time.

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	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
x	x	x	x	✓	x	✓	y	✓	✓	x	✓	✓	y	✓	✓	✓	x	✓	✓

Page Hit = 11 page Fault = 9

* 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 procesing 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 five - 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 refer 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 ?

Memory Allocation Strategies, Buddy Systems

Memory Allocation Strategies, Buddy Systems :-

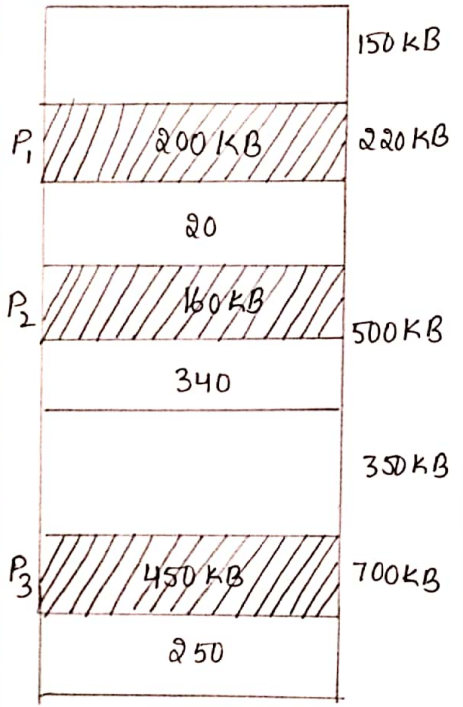
Memory Partition :- 150 KB, 220 KB, 500 KB, 350 KB, 700 KB.

Process :- 200KB, 160KB

First Fit :-

* Allocates first hole that is big enough.
* Scans memory from begining.
* Faster.

Process :- sol



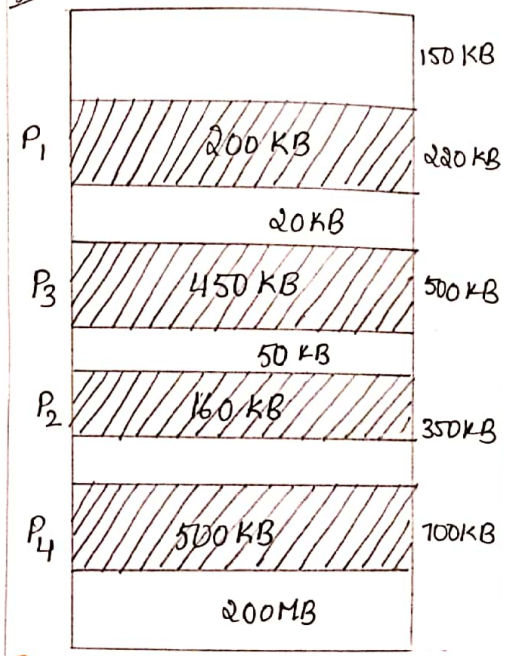
* P→500KB Must Wait

Best Fit :-

* Allocates smallest block (hole).
* Chooses the block that is closest.
* Searches Entire list.

Memory Partition :- 150 KB, 220 KB, 500 KB, 350 KB, 700 KB.

sol :-



Process :- 200KB, 160 KB, 450 KB, 500 KB.

→ Best Algorithm which makes most efficient use as memory.

* starts from Ascending order 160 KB, 200 KB, 450 KB 500 KB.

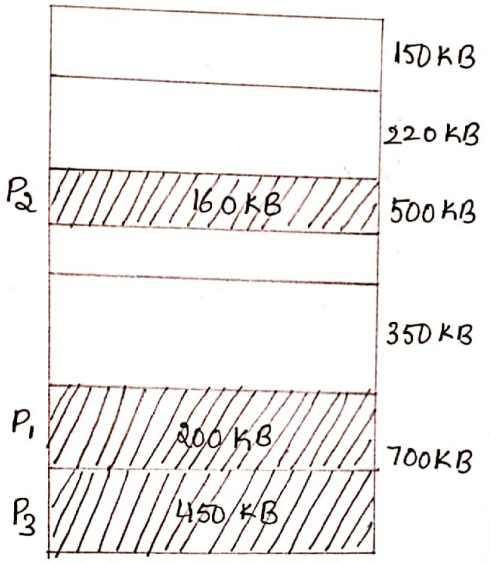
Worst Fit :-

* Allocates larger hole.
* Searches entire list
* Memory space wasted.

Memory Partition :-

150 KB, 220 KB, 500 KB, 350 KB, 700 KB.

Process :- 200 KB, 160 KB, 450 KB, 500 KB.



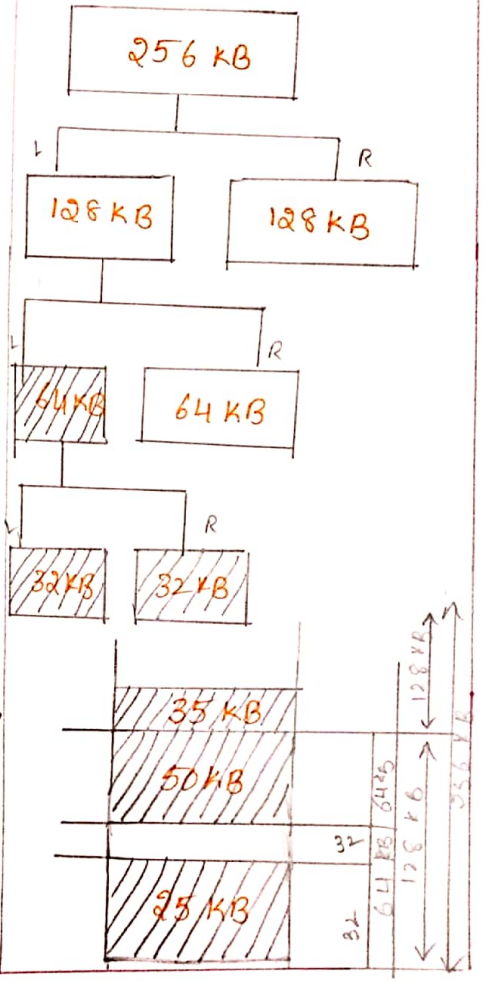
* Highest memory space 700KB, 500 KB, 350 KB, 220KB, 150 KB
* Remaining Process has to Wait.

Buddy System

* Memory → single, contiguous, Free block.
* Request by process → Memory divided into two half blocks.

256 KB.

i) 25 KB ii) 50 KB iii) 30 KB.

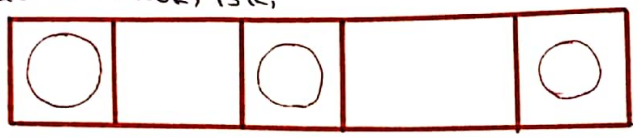


TUTORIAL

Process Request are given as:-

Problem :- 1

25k, 50k, 100k, 75k,



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.

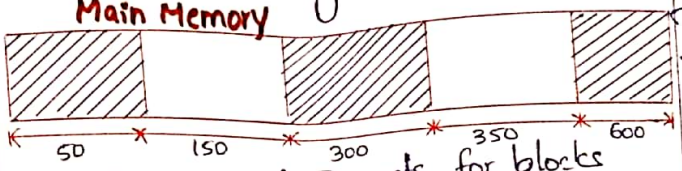
Problem :- 2

Consider six Memory Partitions of size 200kB, 400kB, 600kB, 500kB, 300kB and 250kB. These Partitions Need to allocated to four Processes of sizes 357kB, 210kB, 468kB and 491kB in that order. Perform the allocation of Processes using:-

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

Problem :- 3

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



The sequence of Requests for blocks of size 300, 25, 125, 50 can be satisfied if we use.

1. either first fit or Best fit Policy (anyone)
2. first fit out 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, 600kB, 350kB, 200kB, 750kB, and 125kB (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, 125k, 53k and 210k. 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 Process terminating and Returning its Memory? what type of allocation fragmentation this type of allocation suffers with?

Problem :- 6

Given memory Partitions of 100k, 50k, 600k, 450k, 200k, 600k, determine the available space list after allocating the space for the stream of Requests consisting of the following block sizes 25, 100, 250, 200, 100, 150 rise.

(i) first fit
(ii) Best fit
(iii) worst fit.

Problem :- 7

Given 5 Partitions of 100kb, 500kb, 200kb, 300kb, 600kb (in order) Use first fit, worst fit, best fit, algorithms to place 212kb, 417kb, 112kb, 426kb (in order)

FILE CONCEPT TUTORIAL DAYS

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.

2) Consider a file currently consisting of 100 blocks. Assume that the file control block (And the index block, in the case of indexed Allocation) is already in memory. Calculate how many disks I/O operations are required for contiguous, Linked, and indexed (single-level) Allocation strategies, if for one block, the following conditions hold. In the contiguous allocation case, Assume that there is no room to grow at begining 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

Ans:

	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

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 x Tracks/surface x sectors/track x bytes/sector

Disk capacity = 2x32x64x128
= 524288 bytes

Disk capacity = 524 KB.

4) To store a file with 50,000 fixed length data records where each record requires 40 bytes 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 = 1008 sectors.

No. of cylinders required is approx = 3906.25 / 1008 = 3.86

Total no. of cylinders required = 3.86 cylinders

5) The fast File file system uses an inode array to organize the disk files on disk. Each inode consists of a user id (2 Bytes), Three time stamps (4 byte 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, 1 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*512 + 1*128*512 + 1*128*128*512 + 1*128*128*128*512 = 1082203060

6) Consider a file system that uses inodes to represent files. Disk blocks are 12KB 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?

A:- Disksize: 12KB Pointer size: 8B

no. of pointers in a block would be: $\frac{12KB}{8KB} = 1536$

Total file size: 10 direct disk block size: 10*12KB + 1 single indirect size: 1*1536*12KB + 1 doubly indirect size: 1*1536*1536*12KB = 27GB

Disk Scheduling, Address Translation. DAY 5

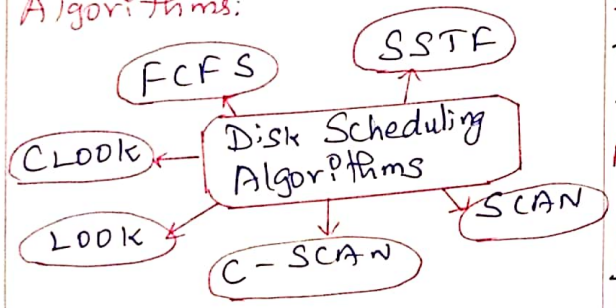
Disk Scheduling :-

* To schedule I/O requests arriving for disk.

Seek Time :-

→ Time taken to locate the disk.

Algorithms:

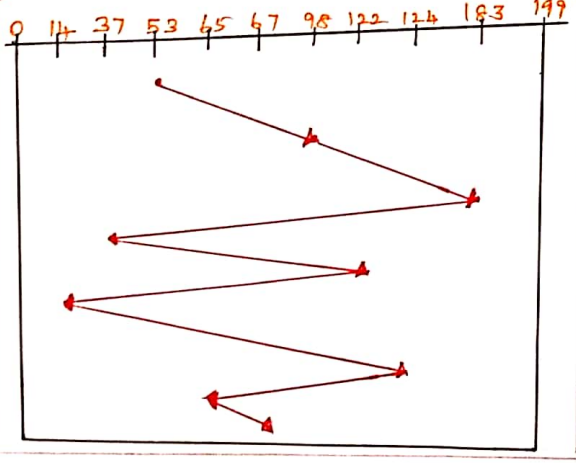


FCFS :

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

Ex: (Order of Request) →
98, 183, 37, 122, 14, 124, 65, 67

Head Pointer : 53.



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

$= 640$ cylinders.

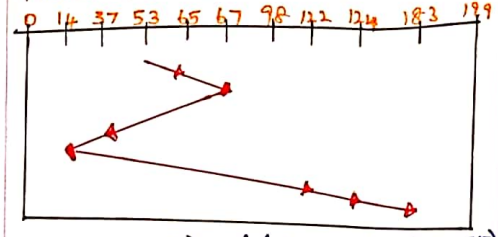
Total Head Movement = 640

SSTF :

* Shortest seek time Serve first
* Closed request serve first.

Ex: Order of Request →
98, 183, 37, 122, 14, 124, 65, 67

Head Pointer : 53.

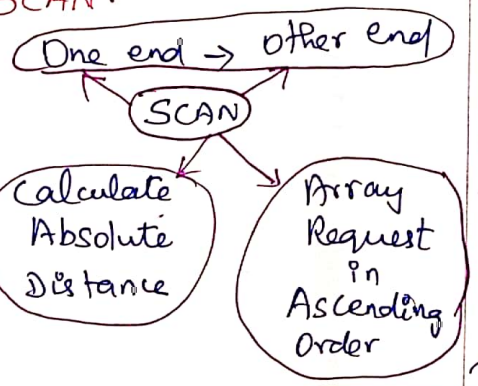


$= (53-65) + (65-67) + (67-35) + (35-14) + (122-14) + (122-124) + (183-124)$

$= 236$ cylinders.

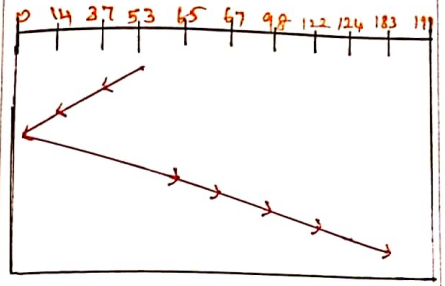
Total Head Movement = 236.

SCAN:



Example :- Order of Request
98, 183, 37, 122, 14, 124, 65, 67

Head Pointer : 53.



$= (53-0) + (183-0)$

$= 236$ cylinders.

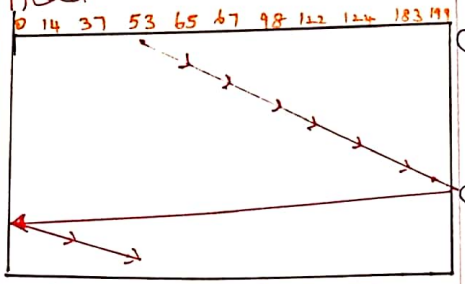
Total head Movement = 236.

CSCAN :-

→ More uniform wait time than SCAN.

Example: Order of Request
98, 183, 27, 122, 14, 124, 65, 67

Head Pointer : 53.



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

$= 382$ cylinders.

Total head Movement = 382

Logical to physical Address Translation :

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

Q: What are the physical addresses for the logical Addresses?
a) (0, 430) b) (1, 10) c) (2, 500)
d) (3, 400) e) (4, 112)

Sol:

(a) (0, 430) (True)
$= 430 < 600$
$= 219 + 430$ (Base + 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) (3, 400)
$= 400 < 580$ (True)
$= 400 + 1327$
$= 1727$
⇒ physical Address = 1727

(e) (4, 112)
$= 112 < 96$ (false)
= Illegal Reference, Trap to O.S.

TUTORIAL – DISK-SCHEDULING PROBLEMS

1. Suppose the order of request is - (82, 170, 43, 140, 24, 16, 190) And current position of Read/write head is '50'. . Find the total seek time for "SSTF algorithm."

2. Suppose the request to be addressed are (82, 170, 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."

3. Suppose the requests to be addressed are (82, 170, 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."

4. Suppose the request to be addressed are (82, 170, 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."

5. Suppose the request to be addressed are (82, 170, 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 "CLOOK algorithm."

6. Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. the drive currently services a request at cylinder 143, and the previous request was at cylinder 125, the queue of pending request in FIFO order is 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 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 (i) FCFS (ii) SSFT (iii) SCAN (iv) C-SCAN (v) LOOK (vi) C-LOOK