SIMATS
School of Engineering

OPERATING SYSTEMS

Computer Science Engineering

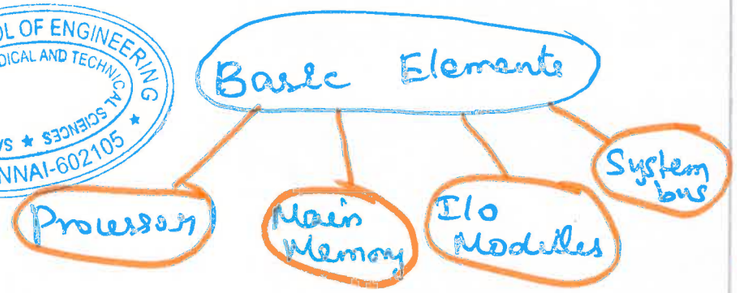
Saveetha Institute of Medical And Technical Sciences,Chennai.

TOPIC 1 : Introduction to Operating Systems

Operating Systems :-

* Intermediatory between the user and computer hardware.

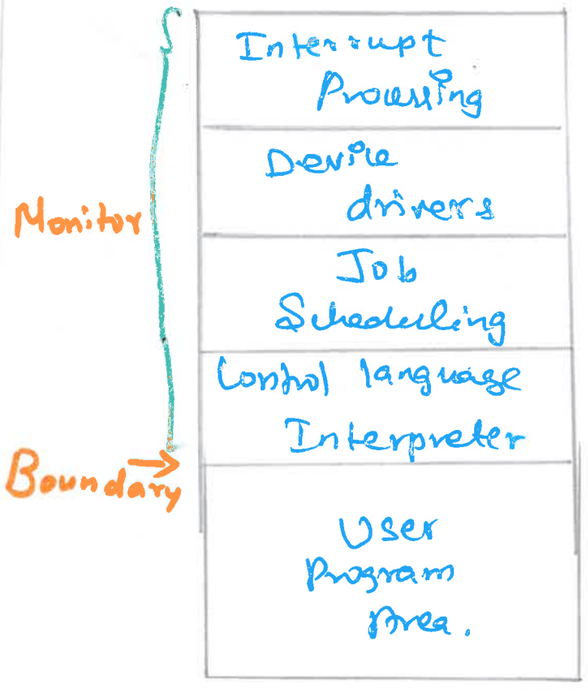
Overview :-



Operating System History :-

* Serial processing. ⇒ Scheduling setup time
* Simple Batch Systems
* Multiprogrammed batch Systems
* Time Sharing Systems.

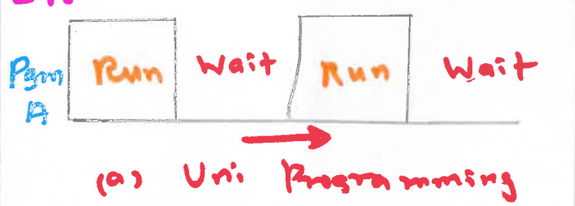
Monitor's Point Of View :-



Processor Point of View :-

* Executing instruction from MM containing Monitor
* Event causes the processor to fetch its next instruction.

Multiprogrammed Batch System :-



(a) Uni Programming

(b) Multi Programming → 2 Programs

Time Sharing :-

* Minimizes Response time
* Commands entered at the terminal.

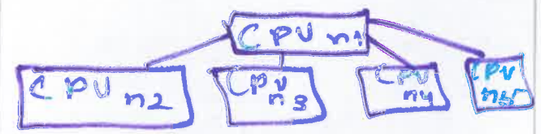
Types of OS :-

* Main frames
* Desktop Systems
* Multiprocessing O.S

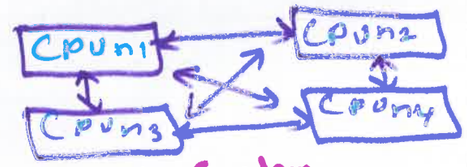
Advantages :-

→ Increased reliability
→ Increased throughput
→ The economy of Scale.

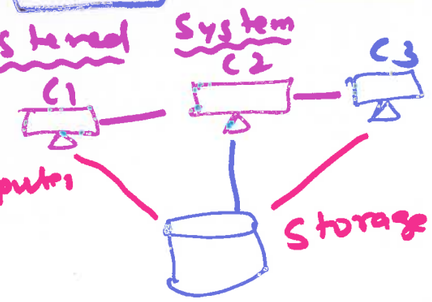
Asymmetric Multi Processor :-



Symmetric Multi Processor

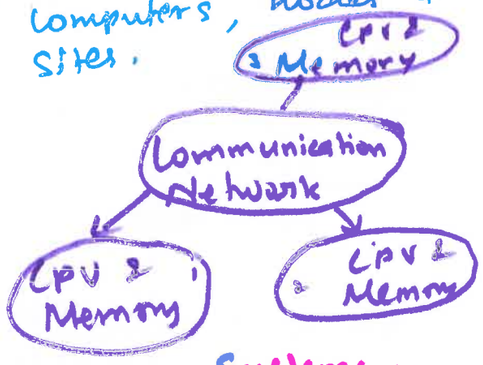


Clustered System :-

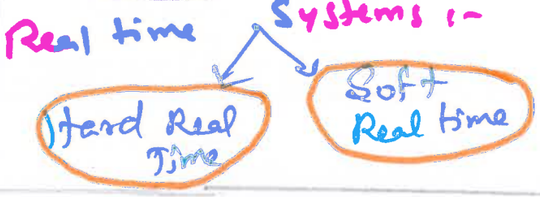


Distributed System :-

* Uses many central processor.
* Connects multiple systems via a single communication Channel.
* Known as loosely coupled Systems.
* Contains numerous nodes & computers, sites.



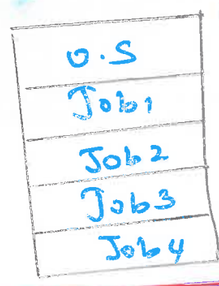
Real time Systems :-

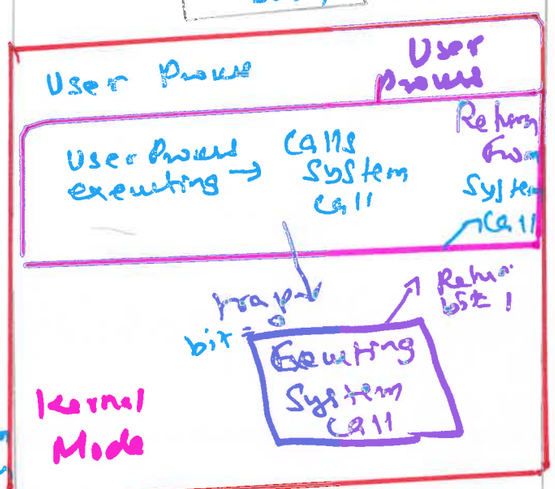


Hand held Systems :-

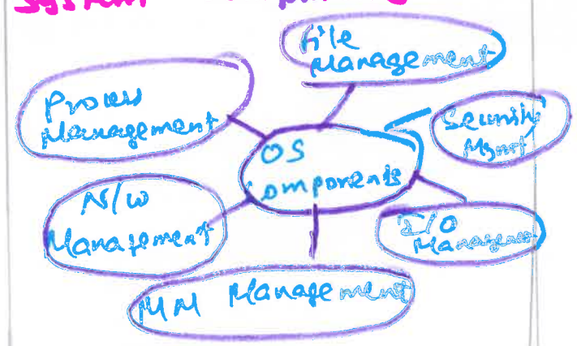
* PDA, Palm - Pilots or Cellular Telephones.

OS Structure :-





System Components



Services :-

* Scheduling
* I/O operations
* Resource allocation
* File Manipulation
* Error detection
* Communication System.

2: SYSTEM CALLS, FUNCTIONS OF OS AND I/O CHANNELS

SYSTEM CALLS:

* Interface between user & kernel modes.

User application
↓
System call interface
Kernel mode
open() implementation return()

Methods to pass parameters to OS
- Simplest
- Block or table in memory
- Stack.

User information from User program → x register → User info in x → OS

Types of system calls
- Process control → Information Management
- File Management
- Device Management → communication & protection

System programs:

* Convenience of execution.

↳ File Management
→ Status Information
→ File Modification
→ Communication.

System Design:

Goals
- User Goals
- System goals

Principles to separate:
- Policy
- Mechanism.

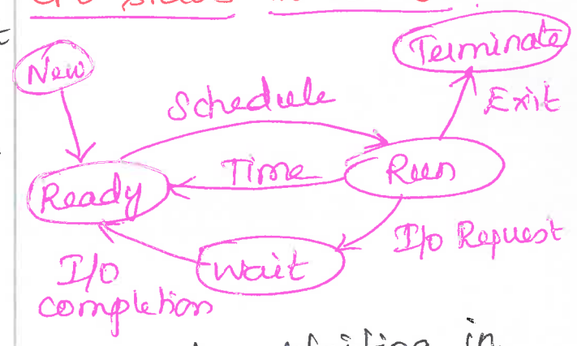
Functionalities of OS:

→ Resource Management
→ Memory Management
→ I/O & Process Management
→ Device & file Management
→ Error Detection
→ Security.

Characteristics of OS

- GUI
- multi user
- multi processing
- multi tasking → Multi threading.

CPU states in an OS:



New state → Waiting in secondary memory for exec.
Ready State → Loaded in main memory.
Run state → Assigned CPU for execution.
Wait state → Requires I/O operation to be completed.
Terminate State → After completion of execution.

Hardware concepts related to OS

* Components working together
* CPU & peripherals.

I/O channels:

* Extension of direct memory access.
* Complete control over I/O operations.

Types of I/O channels.
- Selector channel → * one controller for each device
- Multiplexer channel. → * one controller for multiple devices.

I/O bus:

* Set of wires with a protocol.
* Message transfer
* Daisy chain.

Types
- PCI bus
- SCSI bus
- Expansion bus.

* Addresses for devices for unique identification.

I/O Ports:

Registers
- Status
- Control
- Data in
- Data Out

Memory hierachy:

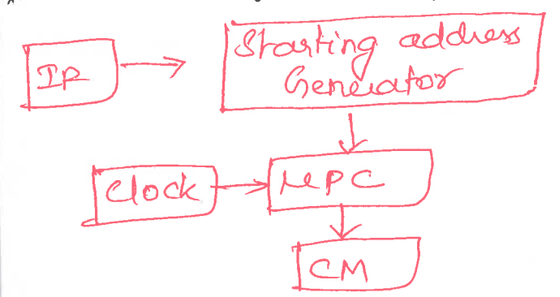
* Speed, cost, size and volatility.

- registers
- cache
- Main Memory
- Solid state Disk
- Magnetic Disk
- Optical Disk
- Magnetic tapes

Volatile ↕ Non volatile

Microprogramming:

* Microcode for microprocessor



Microprogram:

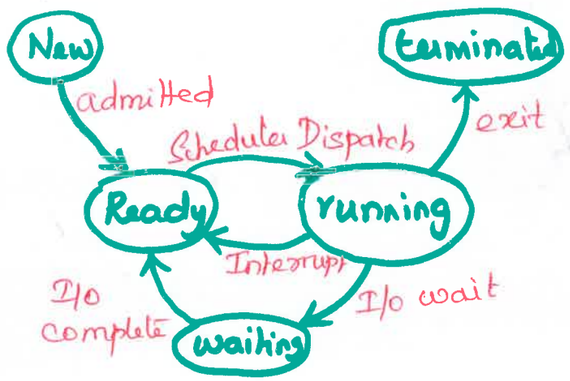
* Sequence of microinstructions
* Microroutines for all instructions in instruction set
* Low level code
* Read control words sequentially from control memory.

3. INTRODUCTION TO PROCESSES

PROCESS :

- Program in execution.
- Need resources — CPU, I/O, files
- Changes state during execution.

PROCESS STATES :



PROCESS CONTROL BLOCK :

- Process representation in the Operating system.
- Contains — process state, process number, program counter, registers, list of open files
- Information associated with the process

PROCESS SCHEDULING :

- Maximise CPU utilization
- Switch CPU among processes
- Have some process running at all times

SCHEDULING QUEUES :

Ready queue
- Processes residing in memory waiting to execute

Device queue
- List of processes waiting for an I/O device

SCHEDULERS :

Long term
- Job scheduler
- Selects processes & loads into memory for execution

Short term
- CPU scheduler
- Allocates CPU to one of the selected processes

Medium term
- To reduce overload (or) degree of multi program

PROCESS CLASSIFICATION :

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.

OPERATIONS ON PROCESSES :

- PROCESS CREATION
- PROCESS TERMINATION

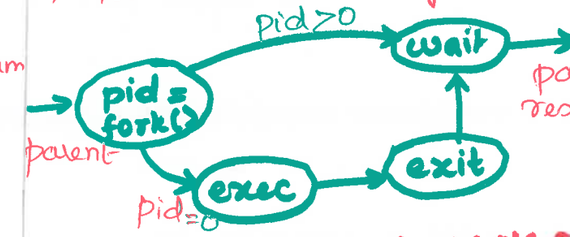
PROCESS CREATION :

- create parent and child processes.
- Process identifier assigned to each process.

Child : can be a duplicate of parent or can have a new program.

PROCESS TERMINATION :

- can be terminated if
  * Child exceeds resources
  * child task not needed
  * Parent is exiting.

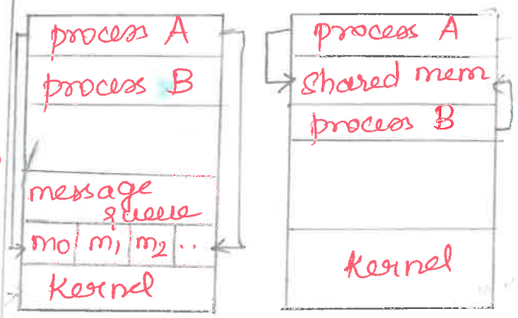


PROCESS CREATION USING FORK ()

INTERPROCESS COMMUNICATION

REASONS :
* Information sharing
* Computation speedup
* Modularity
* Convenience.

MODELS OF IPC :
* Shared memory
* Message passing
  - Allocd process to exchange information.



Communication Models

Shared Memory :
- Region of memory shared by cooperating processes.
- Information read/write from/to the shared region

Types of buffer :
* Unbounded buffer
* Bounded buffer.
  - Fixed buffer size.
  - 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

TOPIC 4 : Process Synchronization

Why Process Synchronization?

- Data consistency
- Cooperating process
- Race condition
  - Wrong o/p if more than one process execution on same memory

General Structure of process Pi

```
do
{
    entry section
    Critical Section
    exit section
    remainder section
} while (true)
```

Critical Section problem

processes : { $P_0, P_1, P_2 \dots P_{n-1}$ }

↓

Critical Section

↓

Share Common variables

Solution to Critical Section

- Mutual Section
- progress
- Bounded Waiting

Approaches for critical Section

- Preemptive kernel
- Non preemptive kernel
  - → only 1 process active at a time

Software based solution to critical section

↓

peterson's solution

↓

```
do {
    flag[i] = true;
    turn = j;
    while (flag[j] && turn == j);
    Critical Section.
    flag[i] = false
    remainder Section
} while (true)
```

where P_i & P_j are two processes.

peterson solution

- mutual exclusion preserved
- progress requirement satisfied
- bounded waiting requirement is met.

Synchronization Hardware

Locking (non preemptive kernel)

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

Mutual exclusion implementation test() and set()

2)

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

Software tools to handle Critical Section

↓

Mutex lock.

```
do {
    acquire lock
    Critical Section.
    release lock.
    remainder section
} while (true)
```

↓

Spin locks (useful for short times)

Semaphores : S

↓

Integer variable

↓

involves 2 operations

↓

1) wait() 2) signal()

*

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

*

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

```
Wait(Semphore *S)
    S->value--;
    if (S->value < 0) {
        add process to S->list
        block()
    }
}
```

TOPIC 5 : Classical Problems of Synchronization

1) Bounded-Buffer problem (or)

Producer - Consumer Problem.

Consumer producing empty buffer for the producer.

↓

2 counting buffers.

full & empty

↓

```
do {
    wait(full);
    wait(mutex);
    // remove an item from buffer to the next cons
    signal(mutex);
    signal(empty);
} while(True);
```

2) Dining Philosopher Problem

Philosper(P) thinking & eating

↓

P dont eat while thinking

↓

When P is hungry he tries to pick up Chopstic

↓ Picks up

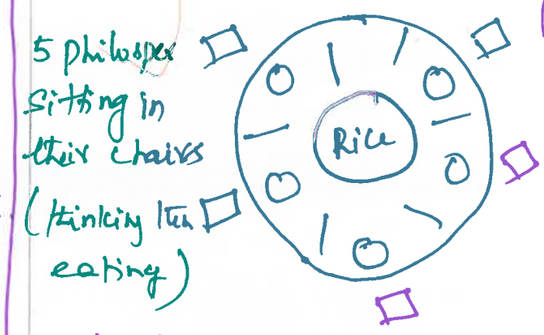
If both chopsticks are available

↓ odd no. of P

Picks up left first then right

↓ Even no. of P

Picks up right first then left.




```
do {
    wait chopstick[i]
    wait chopstick[(i+1)%5]
    signal(chopstick[i]);
    signal(chopstick[(i+1)%5]);
} while(true);
```

3) Readers - Writers Problem

Set of resources

↓ Shared

$P_0, P_1 \ldots P_{n-1}$

↓ Write

If write process is ready only one writer allowed to write

↓

While writing no reading

↓

While reading no writing

↓

Readers only read no write.

```
do {
    wait(rw-mutex);
    /* Writing
    signal(rw-mutex);
} while(true)
```

* Write Process *

```
do {
    wait(mutex)
    read.count++;
    if(read.count == 1)
        wait(rw.mutex);
    signal(mutex);
```

* Reading process *

```
    wait(mutex);
    read.count--;
    if(read.count == 0)
        signal(rw.mutex);
    signal(mutex);
} while(true);
```

4) Sleeping Barber problem.

Barber Shop

↓ one barber

available

* One barber chair, n customer chairs

↓

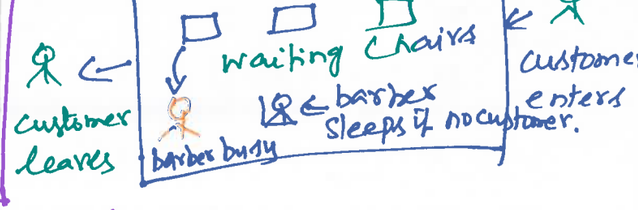
When barber is busy with 1 customer

↓

rest wait in n-1 chairs.

↓ If no customer.

barber sleeps.



Monitors :

↓

to achieve mutal exclusion

eg wait() & notify() in Java.

↓

Only one process P is active at a time.

```
Monitor demo{
    Variables;
    condition variables;
    Procedure P1{ ... }
    Procedure P2{ ... }
}
```

TOPIC 6

Deadlocks

System Model

⇓

Resource Types → R_i → CPU cycles, Memory Space, I/O devices

↓

W_i Instances

Process → Request → Resource
Process → Use → Resource
Process → Release → Resource

Deadlock

Mutual Exclusion
No Preemption
Hold and Wait
circular Wait

Prevent Concurrent processes

Resource Allocation Graph

Vertices V
- Process (P)
- Resource (R) → Instances

Edges E
- Request edge: $P_i \rightarrow R_i$
- Assignment edge: $R_i \rightarrow P_i$

Facts:

Graph → No cycles → No deadlock

Graph → Cycles
- One instance → deadlock
- Several instances → Possibility of deadlocks

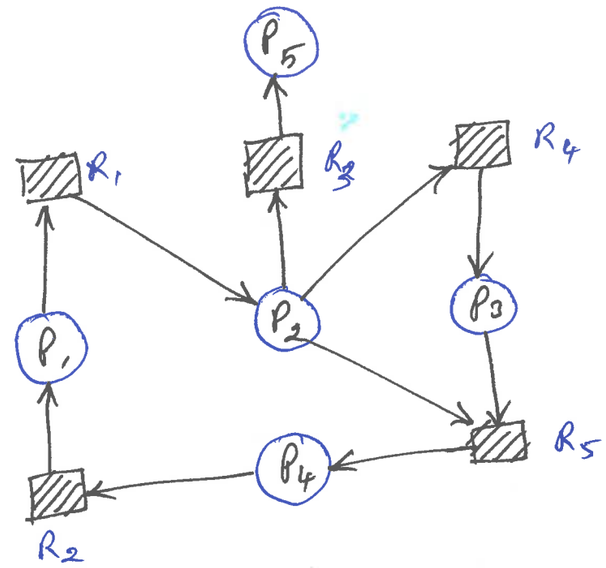
Methods:

* Prevention
  - Mutual Exclusion
  - Hold and Wait
  - No Preemption
  - circular Wait
* Avoidance
  - Need Prior information
  - declare maximum need
  - examine circular-wait

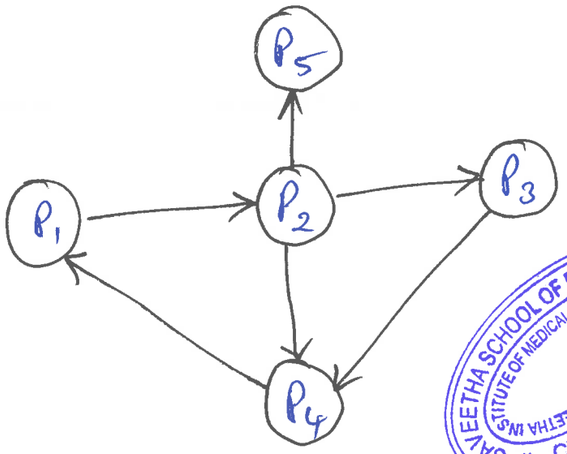
Unsafe → Possibility of deadlock
Deadlock
Safe → No deadlocks

Avoidance
- Single Instance ⇓ Resource Allocation graph
- multiple instance ⇓ Banker's Algorithm

Resource Allocation Graph



wait-for-Graph



⇒ Nodes are Process

⇒ $P_i \rightarrow P_j$ if P_i waiting for P_j

⇒ Periodically invoke an algorithm

⇒ Search for the cycle in the graph.

⇒ Requires an order of n^2 operations

Topic (7)

Banker's Algorithm

Data Structures:

* 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, 1, ..., n-1

2. Find an i such that both:

   a) Finish[i] = false

   b) $Need_i \leq Work$

   If no such i exists, go to Step (4)

3. Work = Work + $Allocation_i$

   Finish[i] = true Go to Step (2)

4. If Finish[i] == true for all i, then the system is in 'Safe State'.

Resource Request Algorithm:

1. If $Request_i \leq Need_i$ go to step 2.

   Otherwise raise error condition

2. If $Request_i \leq$ Available, go to Step 3

   Otherwise P_i must wait

3. Allocate requested resources 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 & restore old resource allocation state

Deadlock Detection:

1. Initialize a) Work = Available

   b) For i = 1, 2, ... n

   if $Allocation_i \neq 0$

   FINISH[i] = false

   otherwise

   Finish[i] = true

2. Find an index i such that both:

   a) Finish[i] = false

   b) $Request_i \leq$ Work

   If no such i exists, go to (4)

3. Work = Work + $Allocation_i$

   Finish[i] = true, Go to Step (2)

4. If Finish[i] == false, for some i, $1 \leq i \leq n$ then 'Deadlock State'

   If Finish[i] == false, then

   P_i is deadlocked

Recovery from deadlock:

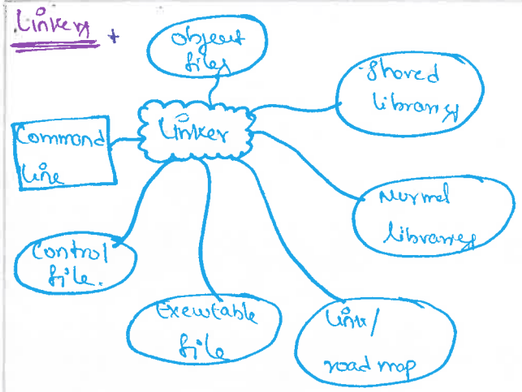
- Process Termination
  - Abort all deadlocked Processes
  - Abort one process at a time, till deadlock cycle is eliminated.
    - Abort → Priority, Computation time, Resources used / Need
- Resource Preemption
  - Selecting a Victim
  - Rollback
  - Starvation

Deadlock Avoidance:

- Single instance → Use Resource allocation graph
- Multiple instances → Use Banker's Algorithm

Topic 8

Linkers +



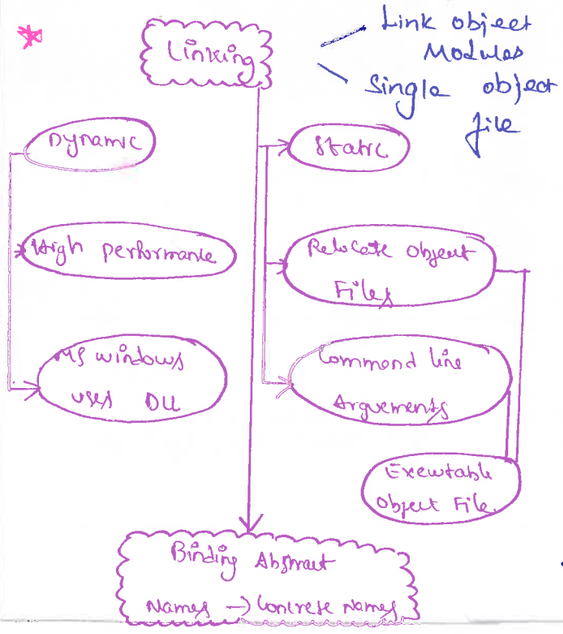
Linker — Program in system
— program into single object file

- Takes one or more object files
- Combine these files into executable files
- output is complete / self sufficient

Functions of Linker

- Collect all pieces of program
- memory organization to fit pieces
- get more memory to run program.

* Linking — Link object Modules
— Single object file



Dynamic Linking

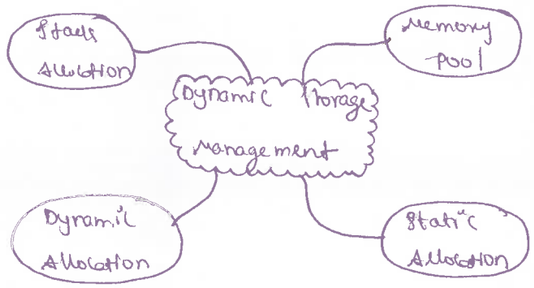
⇒ way to implement Dynamic linking - Jump table

- Contains small function Called when program starts
- link library determines dynamic libraries

Advantages

- memory requirements reduced
- DLL loaded into memory once
- more than one application use single file
- Application support and maintainence cost low

Dynamic storage management +



- Two operations in Dynamic Storage mergement
  1. Allocate a given number of bytes
  2. Free a previously allocated block.

Static Allocation

- used when memory Allocation and freeing are partially predictable
- Stack based organization keeps all free space in one pble.
- Stack used in Tree Traversal, Expression Evaluation

Heap Allocation

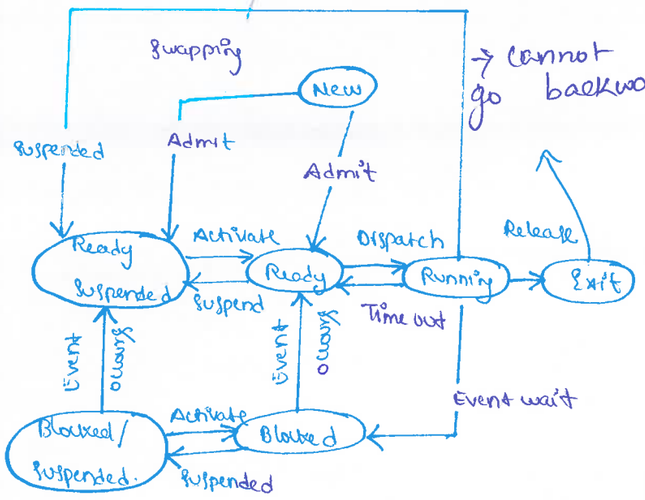
- used when allocation and release are unpredictable
- memory Contains Free Areas and Allocated Areas
- uses free list to keep track of storage

Storage Reclaimation (problems)

- Dangling pointers
- memory leaks

Swapping :-

Swapping b/w (RAM) Main Memory and Secondary memory (HDD)

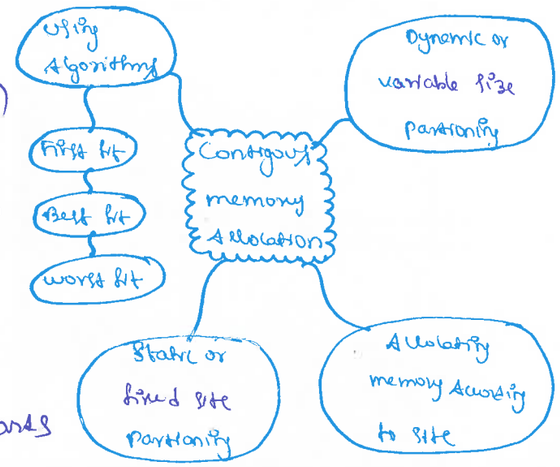


→ cannot go backwards

- 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 back to memory for Execution * HDD ⟷ RAM

- Swapping Exceeds the usual physical memory of System
- Increases degree of multiprogramming in system

Contiguous memory Allocation +



- In variable partioning OS keeps table to know which memory Available which occupied
- Algorithms

First fit - Start search at begining where previous search ended

Best fit - Search for entire list

worst fit - Search entire list until it is sorted

- Allocate the memory According to size
- In Static partioning main memory pre-divided into fixed size partions.

9. OPERATING SYSTEM

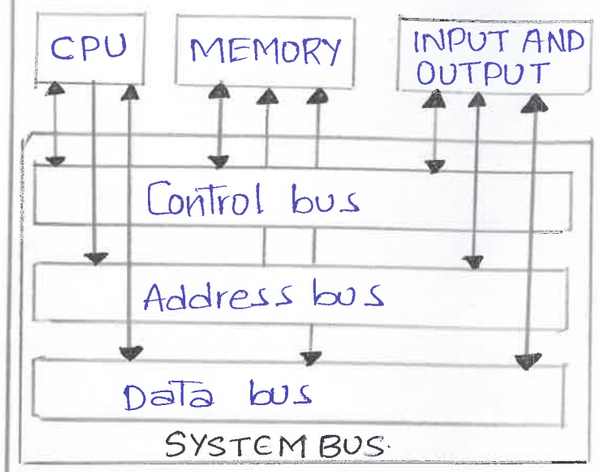
I/O CHANNEL

→ Line of Communication
→ channel in Computing device
→ channel between input and output
→ I/O Bus and Memory to the CPU
→ I/O Computer pheripherals.

BUS

→ A bus is a subsystem that is used to Connect Computer Components and Transfer data between them.
→ Devices Connected to a Computer
→ Also called as Address bus, data bus, Control bus.

EX: A Bus Carries data between a CPU and the system Memory via Mother board.
→ Maintain a schedule of "picking up" data and "dropping it off" at a regular interval.



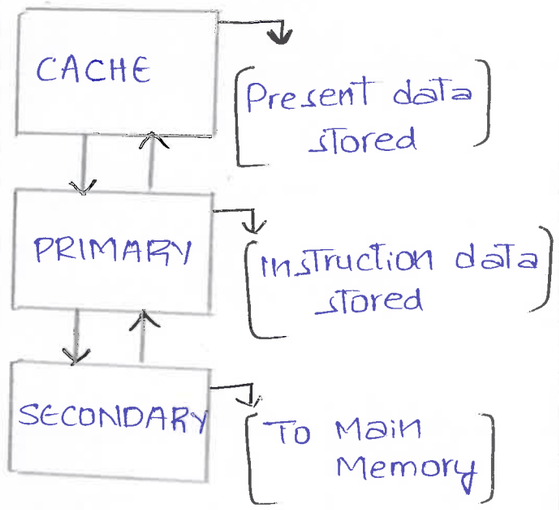
TYPES OF BUSES

1. Address Bus
→ carries Memory Address From the processor to Components
2. Data Bus
→ carries data blw Processor and other Components
3. Control Bus
→ carries Control signals from Processor to other Components

MEMORY

→ Central to the operation of a Modern Computer system.
→ Consists of large array of bytes
→ Consists of its own Address
→ Three types of Memories.

1. CACHE MEMORY
2. PRIMARY MEMORY
3. SECONDARY MEMORY



EXPLANATION

1. CACHE
→ Acts as a buffer blw RAM & CPU
→ It is in high speed
→ It is located in processor
→ High speed Comparing with primary & secondary storage
→ Executing data can be stored in cache.
→ processor Access programs and directly.
→ It is used to reduce Waiting Time.

2. PRIMARY
→ It is an segment of Computer Memory
→ Currently need programs to be shared.
→ Enhance the Efficiency of system.

SECONDARY

→ All permanent and persistent storages
→ Example is ROM [Read only Memory]
→ It overcomes the limitations
→ It can be Fixed and removable.

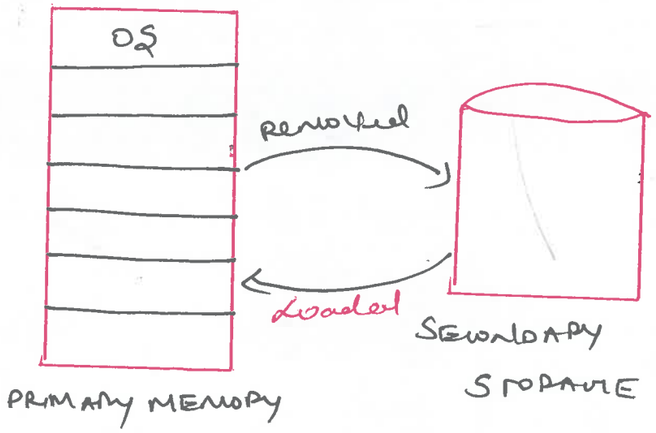
MICROPROGRAMMING

→ It is the process of writing Micro code
→ Low level Code
→ Defines about Microprocessor functions
→ Machine language instructions
→ Translate to several micro-code
→ Realization of Control unit
→ Commands Adds to chips
→ reduction of size of Control Memory
→ Binary values stored as word.

OPERATING SYSTEM (CSA0463) Topic 10

DEMAND PAGING:

* Process of Loading pages
* Pages are demanded
* It works by virtual memory
* Primary (OS) and Secondary (HD) Storages.
* Pages Loads on demand
* Process removed and Loaded
* Valid and Invalid Pages.



PAGE FAULT:

* Needed / Required file is not in primary memory.
* Secondary memory to primary memory.
* OS retrival the file from the memory.

PAGE REPLACEMENT ALGORITHM:

* Decided which memory page to be two Operations
* "Swap in" & "Swap out"
* Page Hit & Page fault
* Three Techniques are there.
  - → FIFO [Simplest page Replacement]
  - → LRU [pages are replaced]
  - → OPTIMAL [Longest duration]

→ **FIFO**: [FIRST IN FIRST OUT]

* Replace the longest time

Ex: 7 0 1 2 0 3 0 → pages

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

* - PAGE FAULT
* - PAGE HIT

→ **LRU**: [Least Recently Used]

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

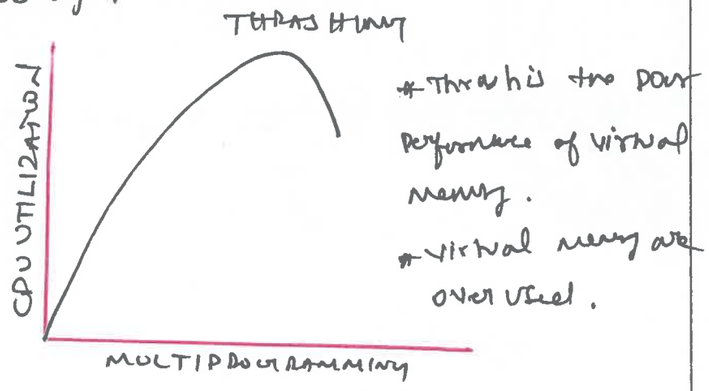
→ **OPTIMAL**: [LATER CASE]

Ex: 7 0 1 2 0 3 0 7 → Pages.

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

THRASHING & WORKING SET:

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



* Through the poor performance of virtual memory.
* Virtual memory are over used.

DISK DEVICES:

* MAGNETIC DISK - Data on magnetized medium
* RAID - Redundant array of Independent disk
* CDROM - Compact disk Read only memory
* HDD - Hard disk drive
* SSD - Solid State drive.

TOPIC - 11 : FILE SYSTEMS: SPACE ALLOCATION, INTERFACE, OPERATIONS and IMPLEMENTATION IN OPERATING SYSTEMS

A collection of directories and files

FILE STRUCTURE

- Text
- Object
- Source

FILE ATTRIBUTES

- Name
- Protection
- Identifier
- Types
- Size
- Location
- Time, date & Security

PROPERTIES

- Stored on disk
- Associated
- Complex Structures

FILE ACCESS METHOD

- Sequential Access
- Direct random Access
- Index Sequential Access

SPACE ALLOCATION

* CONTIGUOUS
  * Address space
  * OS disk address
  * External Fragmentation
* LINKED
  * List of links
  * Pointer
  * Sequential
* INDEXED
  * Index blocks
  * Pointer for specific blocks
  * Individual index blocks

FILE SYSTEM INTERFACE

- Collection of files
- Directory Structure

Directory

- FILE1
- FILE2
- FILE3

FILE DIRECTORIES

- Name
- Type
- Size
- Location
- Position
- Protection
- Usage

FILE SYSTEM IMPLEMENTATION

- File system organization
- File system using data structure

APPLICATION PROGRAMS

LOGICAL FILE SYSTEM

FILE ORGANIZATION MODULE

BASIC FILE SYSTEM

I/O CONTROL

DEVICES

FILE SYSTEM USING DATA STRUCTURE

- ONDISK
  * Boot control block
  * Volume control block
  * Directory structure
  * Per-file FCB
- IN-MEMORY
  * Mount Table
  * Directory structure cache
  * System wide open file Table
  * Per-Process open file Table

- Inode object
- File object

VIRTUAL FILE-SYSTEMS

- Super block
- Dentry object

OPERATIONS

- Open
- Write
- Re-position
- Delete
- Truncate
- Append
- Read
- Close

Topic : 12

UNIX - File System (UFS)

- organize and store large information
- Central component of OS

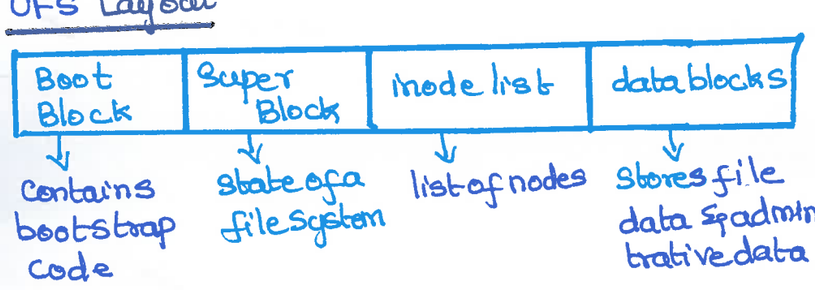
classification

- Ordinary files
- Directories
- Special files
- Pipes
- Sockets
- Symbolic links

Unix file System Types

- ext 3 (formerly ext2)
- reiserfs
- Vfat
- ntfs (now read & write)

UFS Layout



Features of UFS

- Multi-User
- Multi-tasking
- Hierarchical file System
- Portability
- Tools for program development

Limitations

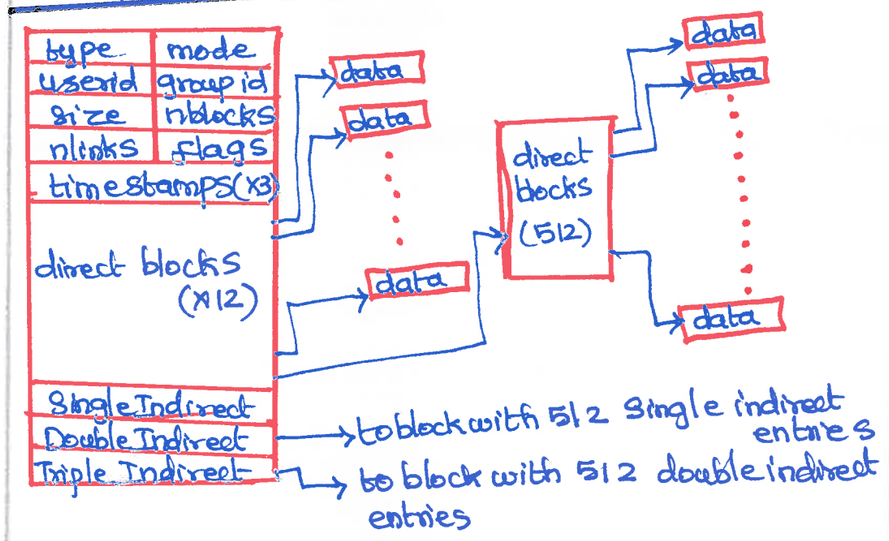
- Unfriendly
- inconsistent
- non-mnemonic user interface
- Shell interface is treacherous
- designed for slow computers
- Lacks in consistency

Layered Architecture

File Operations

- fd = open (Pathname, mode)
- bytes = read (fd, buffer, nbytes)
- count = write (fd, buffer, nbytes)
- reply = seek (fd, offset, whence)
- reply = close (fd)

FS Implementation

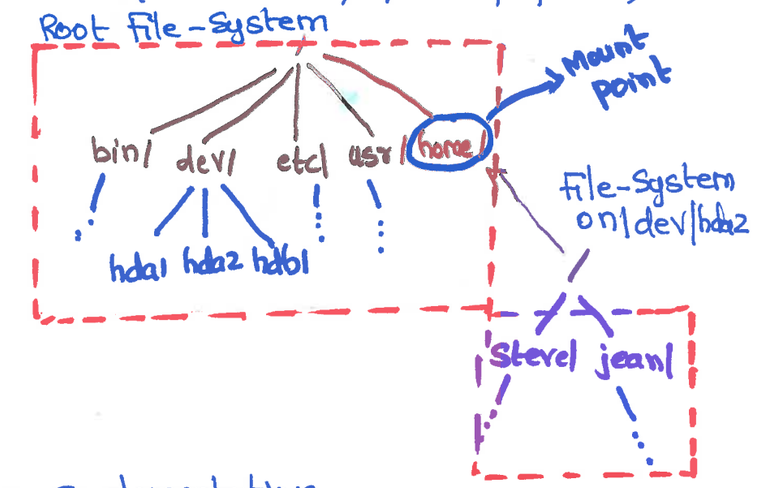


I-nodes (index nodes)

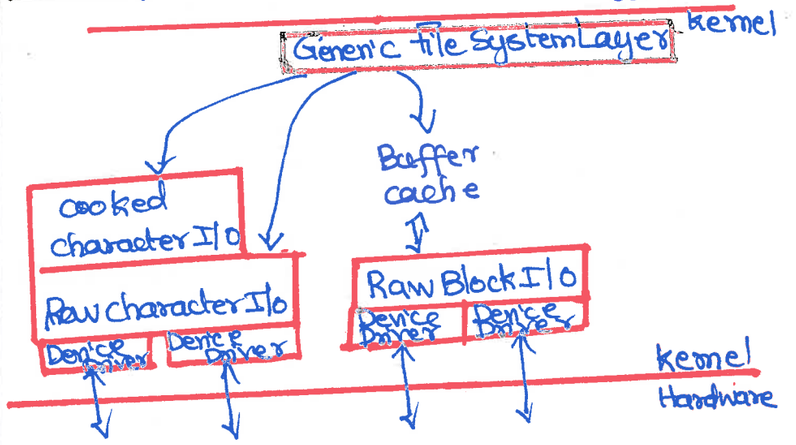
- data structure
- describes a file-system object
- stores attributes & disk block
- attributes contains the metadata
- inode phrases file serial number

Mounting file Systems

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



IO Implementation

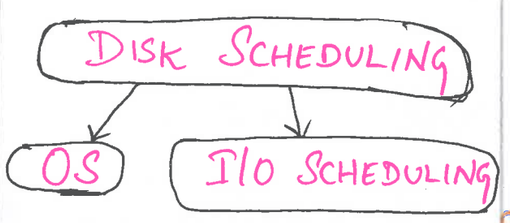


The Buffer cache

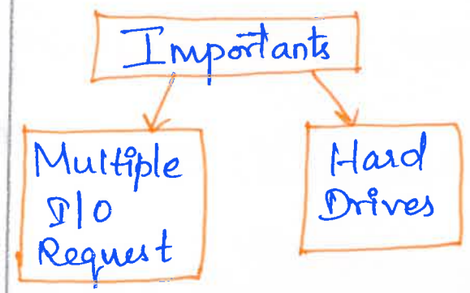
- keep copy of some parts of disk in memory for speed
- prevents disk transfers (85%)
- call sync 30 secs once
- to flush dirty buffers
- catch metadata
- Disks blocks
- Kernel to read from buffer cache.

MODULE 13

DISK SCHEDULING

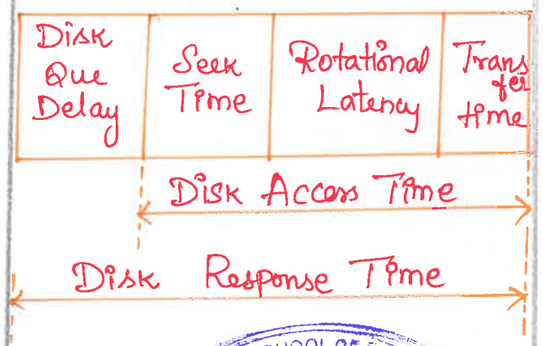


IMPORTANT OF DISK SCHEDULING:

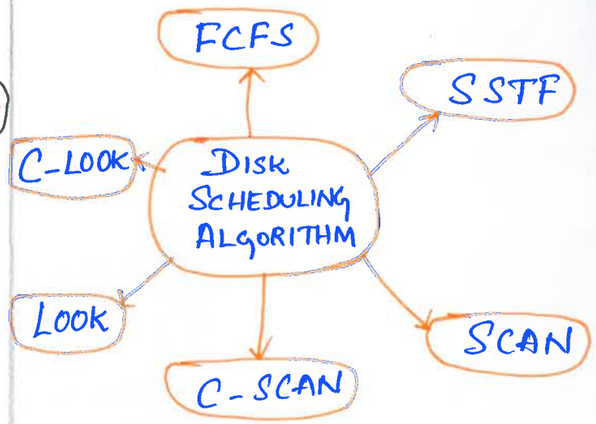


TERMS:

* Seek Time
* Transfer Time
* Rotational Latency
* Disk access time.



ALGORITHMS:



FCFS:

* First Come first served.
* Requests served in order.

EXAMPLE:

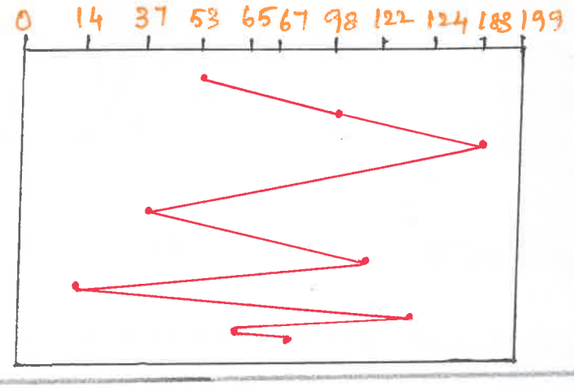
Order of request: 98, 183, 37, 122, 14, 124, 65, 67.

Head pointer: 53

→ Head initially at Cylinder 53.

→ It will move from 53 to 98.

→ Then 183, 37, 122, 14, 124, 65, 67.



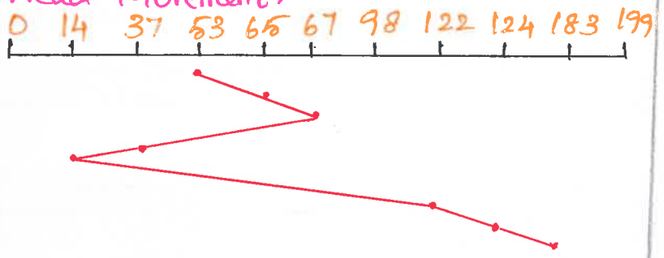
$= (98-53) + (183-98) + (183-37) + (122-37) + (122-14) + (124-14) + (124-65) + (67-65) = \boxed{640}$

Total head movement

SSTF:

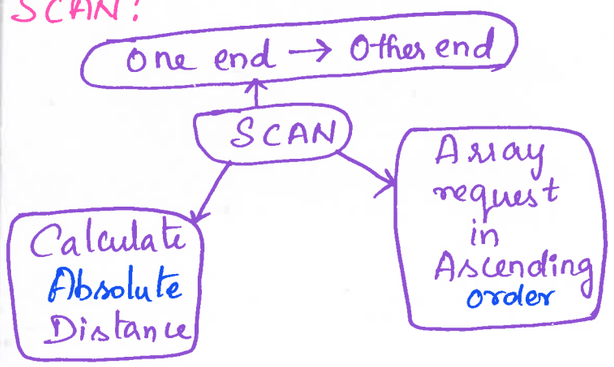
* Short Seek time Serve first.
* Closest request serve first
* Let request array with head.

Head Movement:



$= (53-65) + (65-67) + (67-35) + (35-14) + (122-14) + (122-124) + (183-124) = \boxed{236}$

SCAN:



LOOK:

* Improved version of SCAN
* Calculate absolute distance
* If no request reverse the direction.

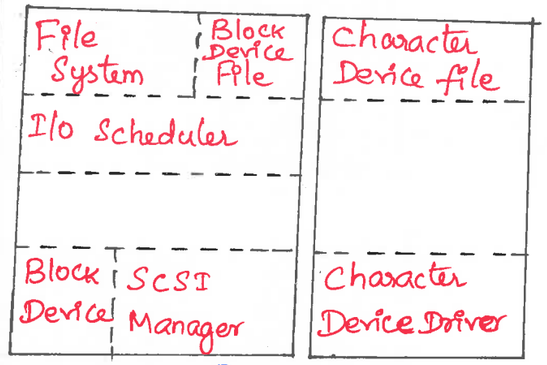
C-SCAN:

* Circulator Elevator Modified version
* Head Services only in right.

C-LOOK:

* Circular look
* Head requests in one direction

DEVICE DRIVER:



* Block Buffer Cache
* Act as pool of buffers.

Character Devices:

* Perform I/O in a byte
* Additional Interfaces
* Read & write entry points.

I/O Control:

* Devices have characteristics that can be configured.
* Open ended & frame buffers.

Stream Drivers:

* Programming model
* Data Asynchronously.

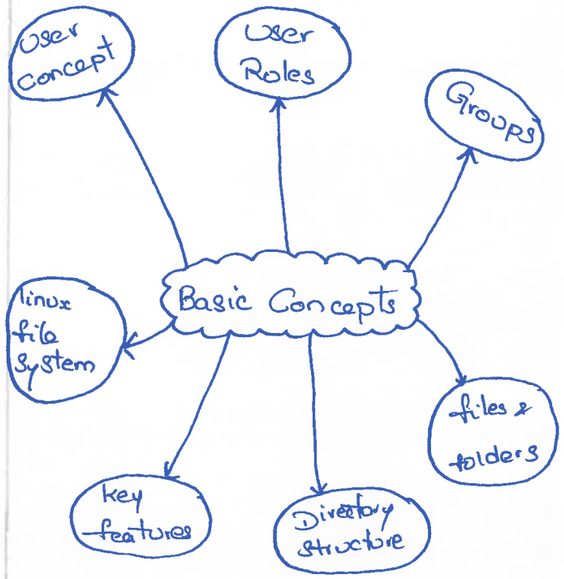
Block Devices Driver:

* Non Volatile Storage
* Supports file system

Linux System

Topic 14

Basic concepts :-



→ key features contains
- Specifying paths
- Partitions, drives/devices and directories

→ This is case sensitive (file extension)

Components - Diagram :-
→ kernel
→ System libraries
→ System utilities.

System Requirements :-
→ web Server → mail Server
→ DNS Server → FTP Server
→ mail delivery agents
→ webalizer for web site statistics.

Responsibilities in Linux
→ Performance monitoring
→ Tuning and management
→ administration & documentation

Setting up Local Network Services

Configuring network :-
→ DHCP is used during the debin installation, then configure the Server with a static IP address
Eg:- Debian

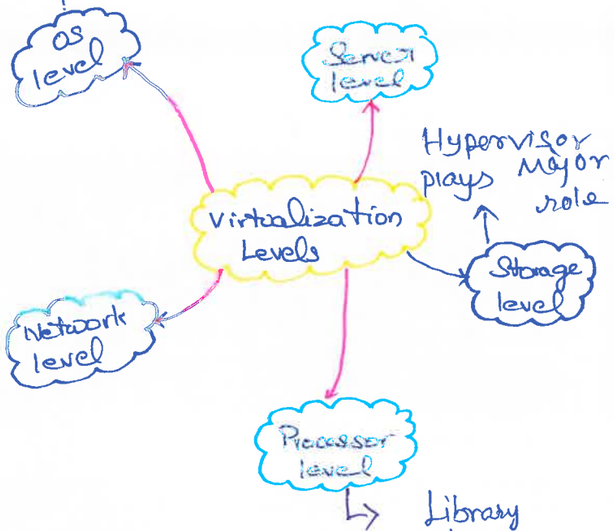
DHCP :- [Dynamic Host Configuration Protocol]
→ It is used to solve a number of Problems associated with local network environment.

DNS :- [Domain naming system]
→ DNS is the hierarchical and decentralized naming system used to identify computers.
→ DNS server converts URLs and domain names into IP
→ Transfers user typing into machine
→ find a web page

Virtualization

*Guest OS could be installed using
i) ISO image
ii) DVD ROM
- Setting up Xen
- VMware on linux host
- Adding Guest Os

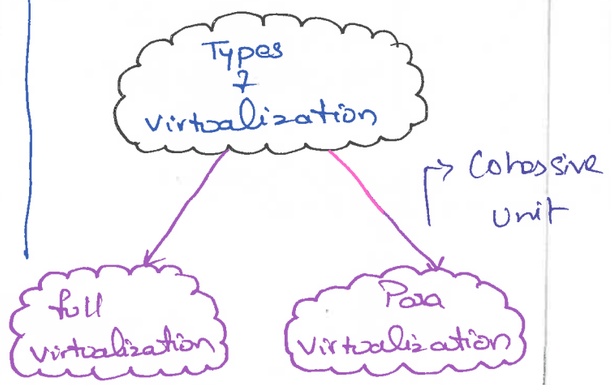
→ It is the creation of the virtual version.
→ Allows one computer to do multiple computers.



VMM :-
[Virtual machine monitor]
→ It is a middle ware
↳ It is a tool for VM Monitoring

Benefits :-
→ replaces wasteful array.
→ It simplifies administration
→ maintains the documents
→ reduce complexity
→ reverse the trend of Sever sprawl.

Example Hypervisor Software



full Virtualization :-
→ does not modify Guest Os
→ VMware workstation applies full virtualization.

Para Virtualization :-
→ modifies the Guest Os
→ Supported by Xen, Denali
→ Reduces overhead.

XEN- Hypervisor :-
→ It is a hardware virtualization technique allowing multiple Os called guests to run on a host machine.

TOPIC 15 : SETTING UP LOCAL NETWORK SERVICES, VIRTUALIZATION BASIC CONCEPTS, SETTING UP XEN, VMWARE ON LINUX HOST AND ADDING GUEST OS

SETTING UP LOCAL NETWORK SERVICES

LINUX :-

- Popular in Computer Networking
- Telecommunication Industry
- Source code can be downloaded from FTP / HTTP Sites on Internet

TCP / IP

- Transmission Control Protocol / Internetworking Protocol
- IP address Composed of 4 octets separated by Decimal points

Eg :- 192.168.7.127

NETWORK AND BROADCAST ADDRESS :-

Eg: 192.168.1.0 (N/w Address) 192.168.1.255 (Broadcast Address)

IP Address

- Static
  - * IP address assigned manually
- Dynamic
  - * IP address assigned automatically
  - * Use DHCP

VIRTUALIZATION

- Use cloud Technology
- NO pre-installation
- Virtual Disk remotely stored on Server

OS VIRTUALIZATION

- LINUX
- WINDOWS

SETTING UP XEN

1. INSTALL XENSERVER, DOWNLOAD XENCENTER
2. START (OR) IMPORT VMS TO VM SERVER CHOOSE TEMPLATE
3. CONFIGURE ISO LIBRARY VIA REMOTE SHARE
4. AFTER CONFIGURATION CREATE VM IN XEN SERVER
5. OTHER STEPS ARE SIMPLE & STRAIGHT FORWARD

FINISH SETUP: INSTALL XENSERVER TOOLS TO ENABLE COPY & PASTE RESOLUTION ADJUSTMENT

VM Ware on Linux Host and Adding Guest OS :-

VM Ware

- Leading Provider of virtualization products
- Install on Linux or windows Server

SETTING UP VMware Server on Linux :-

1. Download VM ware Server rpm for Linux http://www.vmware.com
2. Setting up Access to Server
3. Log into Linux Server
   ssh linux 01
   cd/ media / usbdisk / Source / Vmware / server linux // 0.3 SM
4. Install the RPM
   rpm - ivh vmware - Server - 1.0.3 - 44356
   Configure VM ware

INSTALL VM ware Console on Windows

1. Download Vmware Console for windows from http://www.vmware.com
2. Windows workstation install Server
3. Configure the Console In connect to Host Dialog Box. Enter the Host name, User name and Password