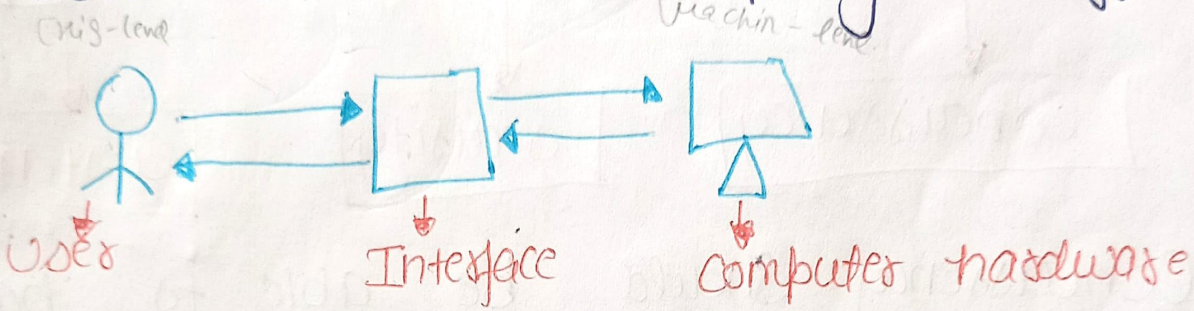
Operating System

- An operating system is a program that acts as an interface b/w the user and computer hardware.
- The purpose of an operating system is to provide a platform on which a user can execute program efficiently.



eg:-> Linux operating system
Window operating system

- It performs all the basic tasks like file management, Memory Management.

Goal of operating system

- Is to make a computer environment more convenient to use
- Secondary goal is to use the resource most efficiently.

Why we use

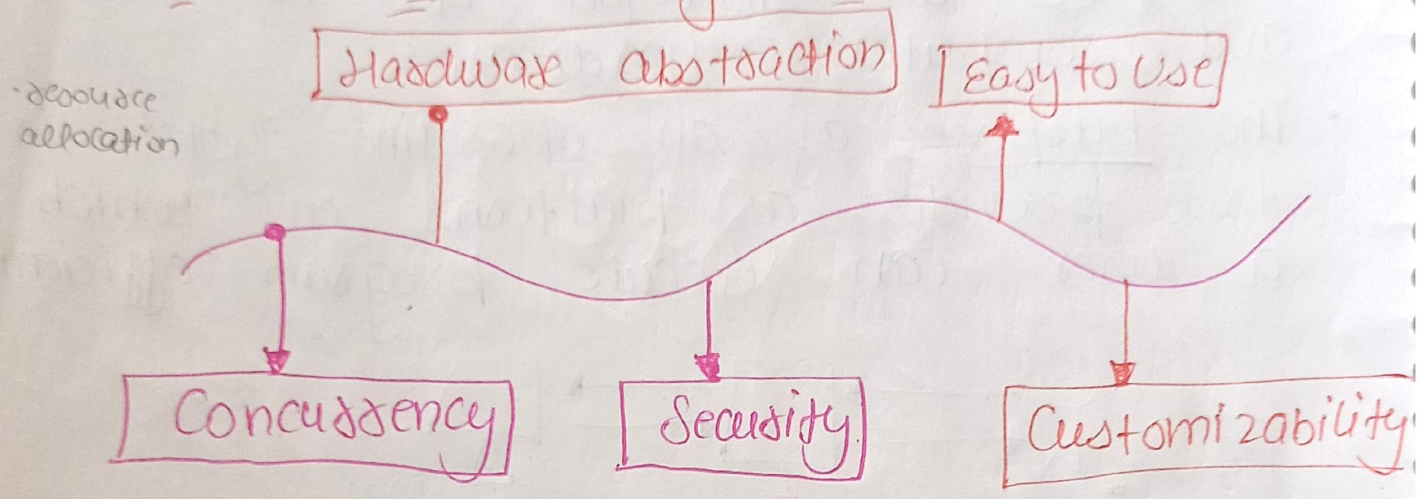
It works as an intermediate b/w System Hardware & End-User.

Copyrighted by Codelcap.io
All rights reserved.

Scanned with OKEN Scanner

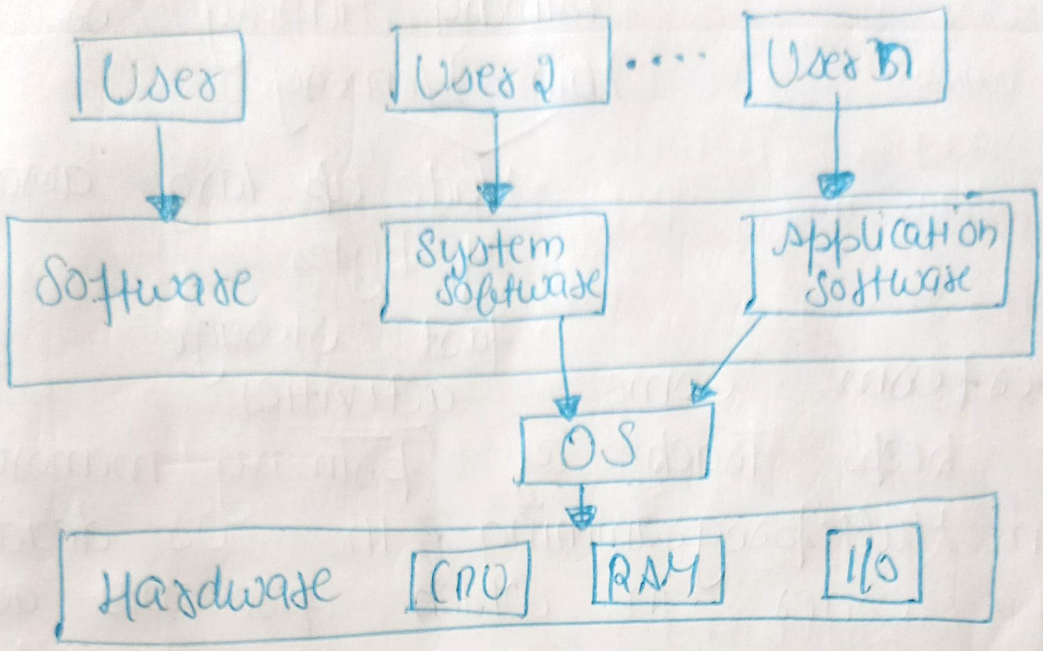
- Control all Computer resource
- provide resource for user program
- provide an interfaces to the user.
- hides the complexity of Software.

Characteristics of OS



- Concurrency: Should be able to handle multiple tasks or processes at the same time.
  - improve system efficiency
- Security: like protection against unauthorized access. also provide features like backup to ensure data.
- Hardware abstraction: user-friendly, provide a simple interface for user.
  [hide species
- Customizability: enabling user to tailor the system to their needs and preferences.
  [Modify system setting
- Scalability [small to large device]
- Virtualization. [Create virtual Memory in CPU]

Components of OS | Architecture



Copyrighted by Codecap.io
All rights reserved.

- User: (people, machine, other)
- Hardware: provide Basic computing resource.
- Kernel: active part of OS
  - central component of OS, that manages operation of computer and hardware.
- Shell: called command interpreter.
  - set of program used to interact with the application program.

Function of Operating System

1) Memory Management
2) Processor Management
3) Device Management
4) File Management
5) User interface
6) Booting the Computer.
7) Security

Need
- Manage computer resources
- establish a user interface.
- execute and provide service

Scanned with OKEN Scanner

- **Memory Management:** The OS manage the Primary Memory or Main Memory

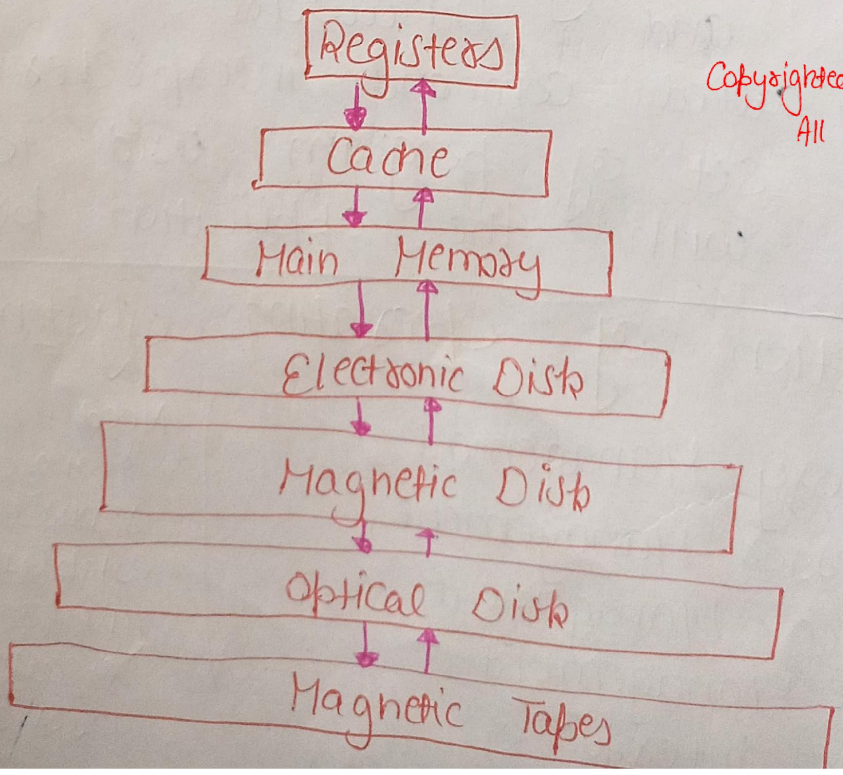
for a program executed, it should be first loaded in Main Memory.

- Made up large array of bytes → certain address.
- fast Storage.

- It perform some activities
  1) It keeps Track of primary memory.
  2) In Multiprogramming, the OS decides the order in which processes are granted memory access, for how long.
  3) allocated and deallocates the memory

allocated → when the process request it

deallocates → when the process has Terminated.



Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

2) Processor Management:

In Multiprogramming environment, the OS decides the order in which processes have access to the processor and how much processing time each process has. → This function of OS → Process Scheduling

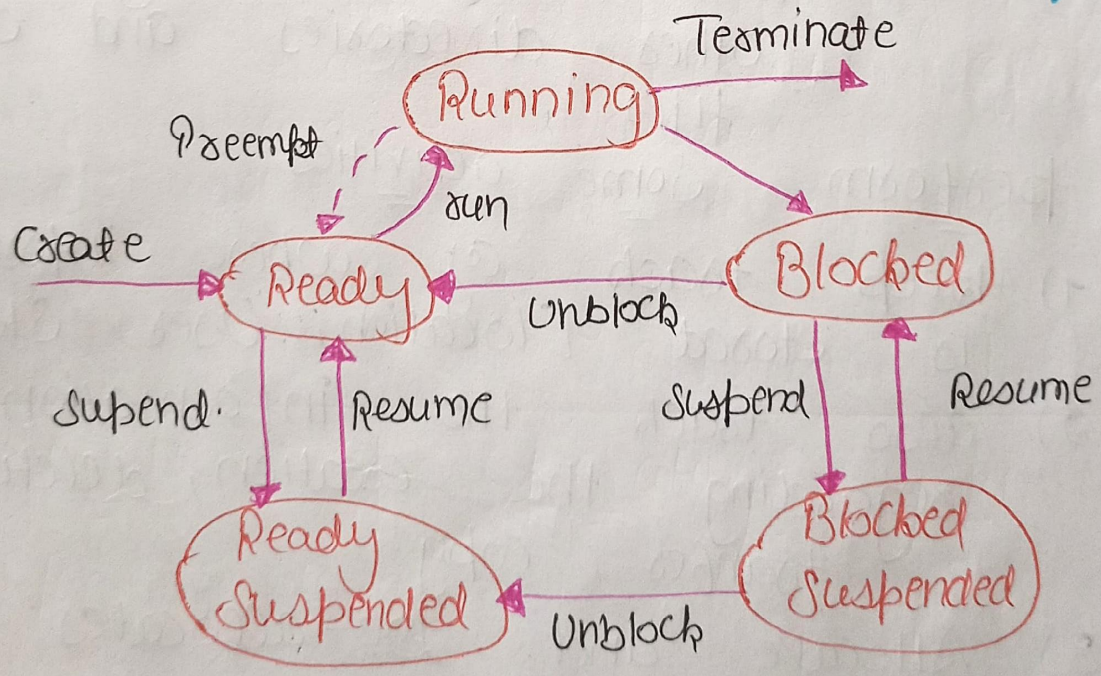
· It follow some activites.

1) keep track the Status of processes.
[The program which perform this task is known as Traffic Controller

2) Allocate the CPU and De-allocated processor

Allocate → that is a processor to a process

De-allocated processor → when a process is no more required.



3) Device Management:

an OS manage device communication via its respective drivers.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

· It perform some activities

1) keep Track all of devices. Program responsible for this task is known as [I/P controller)
2) Decides which process is accessed and for how long
3) Allocate the device in efficient way.
4) De-allocate the device, when no longer required.

4) file Management:- file system is organized into directories for easy to use.

- These directories may contain other directories and other files.

· It perform some activities

1) keep track of where information is stored [location, User, States]
2) also track the information regarding the creation, deletion, transfer, copy.
3) Allocate and De-allocates the resources
4) also maintain the integrity of data stored in these files.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

5) User interface :- also known as Command interpreter.

- It interact with CS through the OS.
- OS acts as interface b/w the User and computer hardware.
- User interface is offered through a set of commands

High level language → Interpreter → Machine language

one language into another

6) Booting the computer :- The process of starting or restarting the computer is called as booting.

- If computer is switched off, completely and if turned on, then it called Cold Booting.
- Warm Booting :- process of using the OS to restart the computer.

7) Security :- The OS use password protection to protect user data

- Protection against unauthorised access
- Protection the system memory against Malicious access.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

Types of OS

It is a type of Software that works as an interface b/w the system program and the hardware.

- Single User :→ It is a System in which only one user can access the computer system at a time.
  eg: MS-DOS
- Multi-User :→ It is a system that allow more than one user to access a computer system at one time.
  eg: Mainframes

Classification of OS

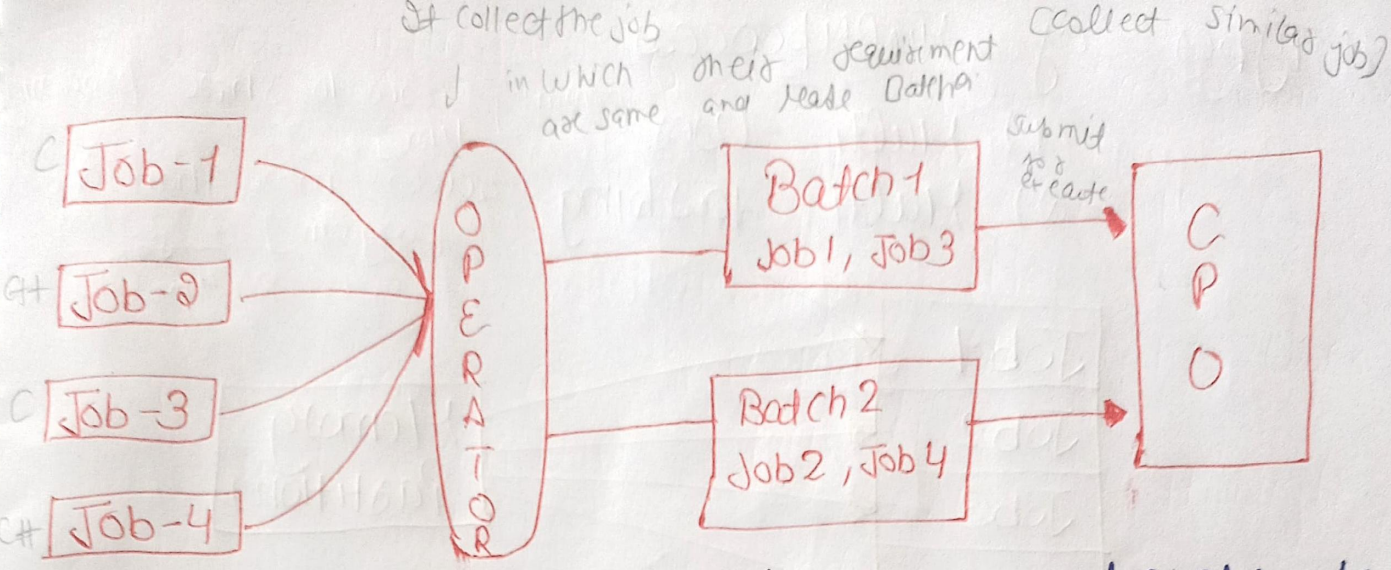
1) Batch operating System: This type of OS does not interact with the computer directly.

- There is an operator which takes similar job having the same requirement and group them into baches.

i.e Tasks or jobs are grouped together into batches.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner



- It is the responsibility of operator to sort jobs with similar needs.

Advantages

- easy to manage large work.
- Multiple users can share the batch system.

Disadvantage

- sometimes costly
- hard to debug

Example: Payroll System, Bank Statement

---

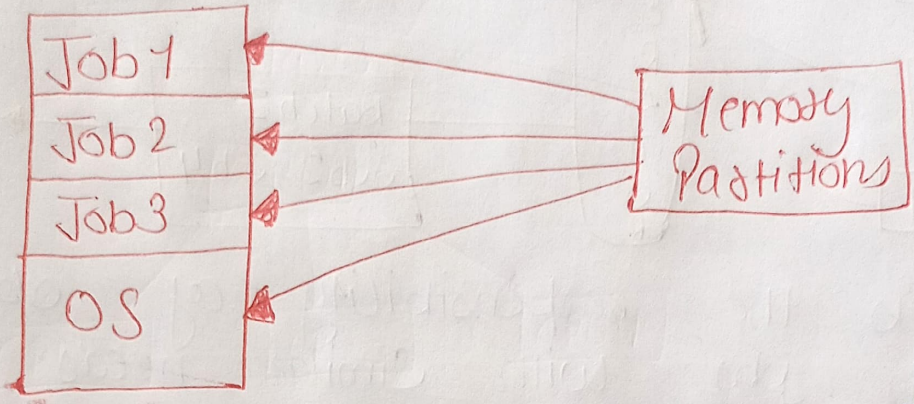
2) Multi-programming OS: } More than one program is present in main memory and one of them can be kept in execution.

Multi → More than; programming → execution of program.

- It increases CPU utilisation by organising jobs, so that the CPU always has one to execute
- Developed in 1950.

Scanned with OKEN Scanner

- Types of Multiprocess:
  1) Multi-User [many users to share processing time] etc
  2) Multi-tasking. [allow more than one program] at same time



- It is used for better execution of resources.

Advantage:
- helps in reducing response time.
- increase the throughput the system

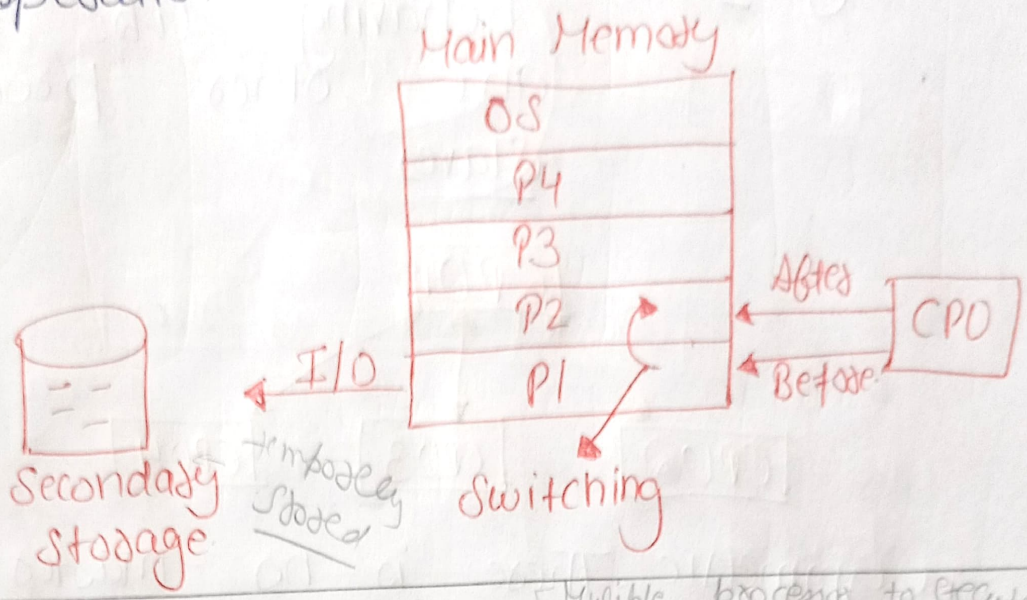
Disadvantage
- Not any facility for user interaction

features
- Need simple CPU for implementa
- High performance.
- CPU idle time reduced.

How its work
Multiple program are to be stored in memory and each program has to be given a specfic portion of memory.
- The OS system handles all these process and states.

Scanned with OKEN Scanner

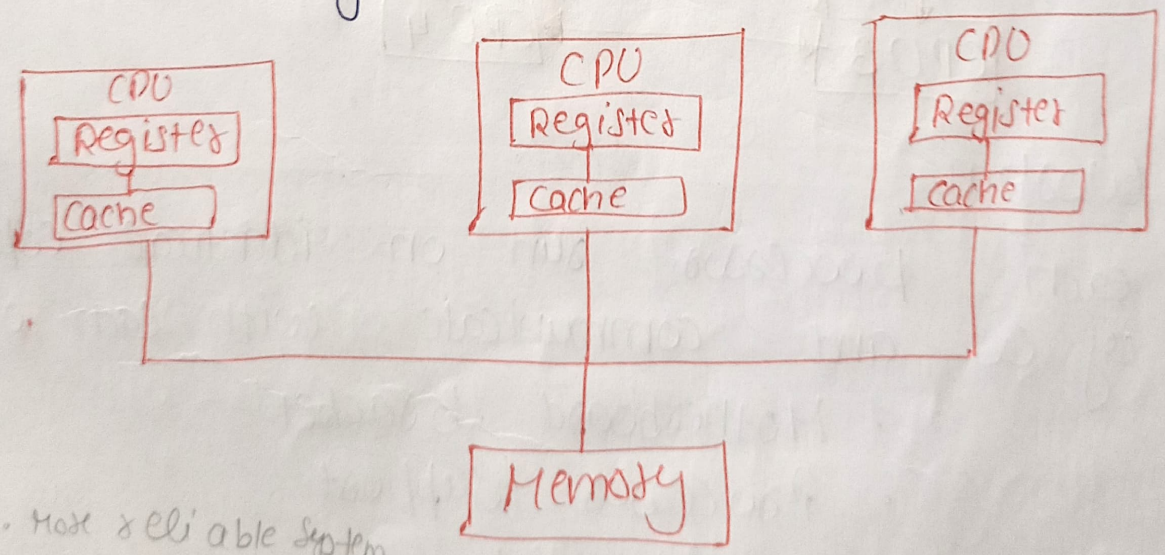
- Before process execution, the OS selects a ready process after CPU execution.
- I/O operation uses.



3) Multi-processing OS: } More than one CPU is used for the execution of resources. in single system

More than one processor present in system which can execute more than one process at same time.

- It betters the throughput the system.



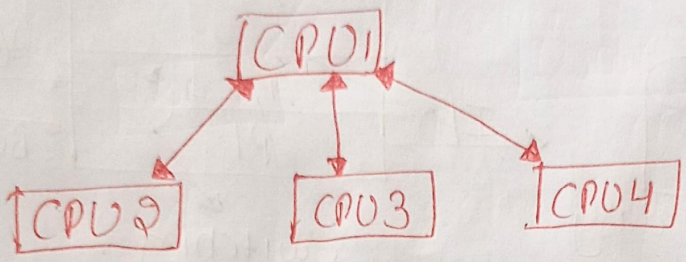
- More reliable system

Advantage: It has several processor, one fail and use another.

Disadvantage: Do to Multiple CPU, it can be more complex. large memory require

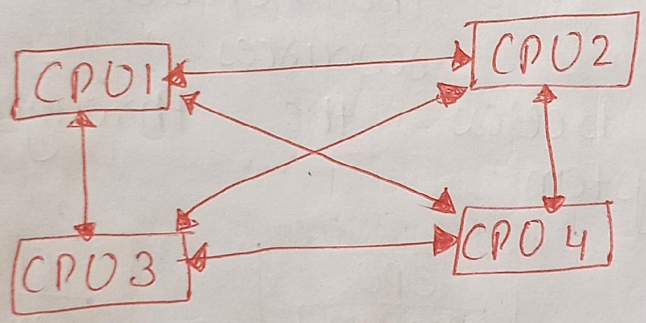
Types of OS

- Asymmetric: This involves one processor behaving like a master and other processor as slaves



← Master-Slave relationship

- Symmetric: There is no-master-slave relationship among the processors.
  - all the processor are identical and perform identical function.



→ peer-to-peer relations.

Working

each processor run on identical copy of OS and communicate with each other

- Motherboard bracket
- Processor support
- Shared data.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

Multiprogramming	Multiprocessing
• More than one program in main memory	• More than one processor per system. [Multiple CPU
• It occurs by switching	• It occurs through parallel processing
• Less throughput and efficiency.	• High throughput and efficiency
• The no. of CPU is one	• The no. of CPU is more than one.
• It takes more time to process the jobs	• It takes less time for job processing

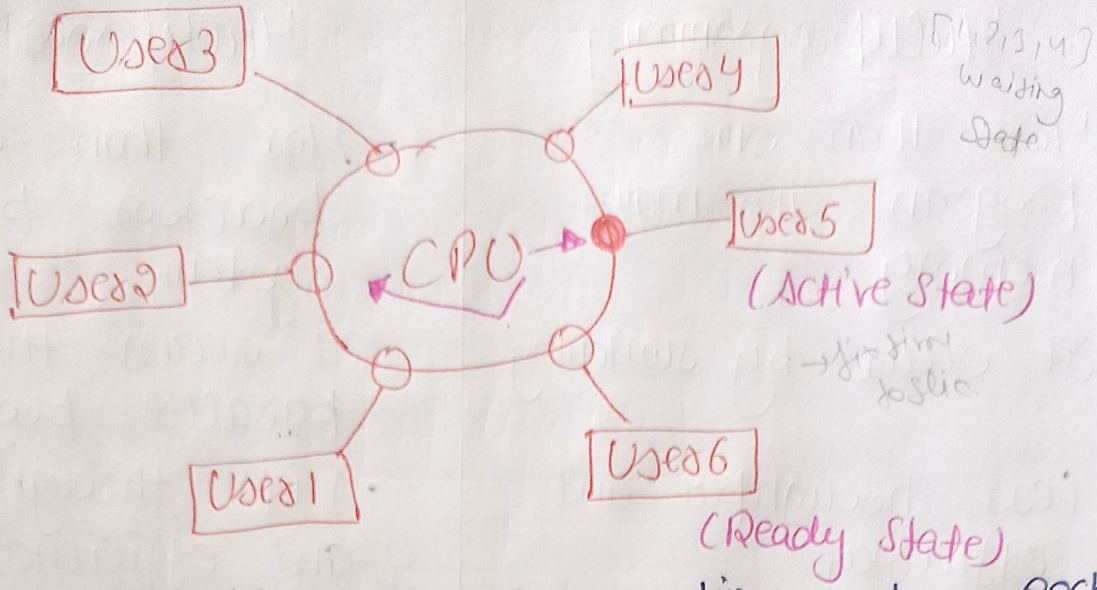
4) Time Sharing OS: [allow multiple user interact a computer] also known as ~~Time Sharing~~ Multitasking Systems. [divide CPU time into slices]

The CPU will provide a same time period to each on every process to complete its task as soon as possible. whether it is a long or short process

- each task is given some time to execute that all the tasks works smoothly.
- each user get the time of CPU as they use a single system. this system are also known as multitasking system.
- The task can be from a single user or different user after time interval OS switches to next task.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner



- CPU provide a same time to each process

Advantage
- each tasks get an equal opportun.
- CPU idle time can be reduced
- Resource sharing.

Disadvantage:
- Reliability problem
- Complexity
- Security Risks

Example: TSO, IBM

---

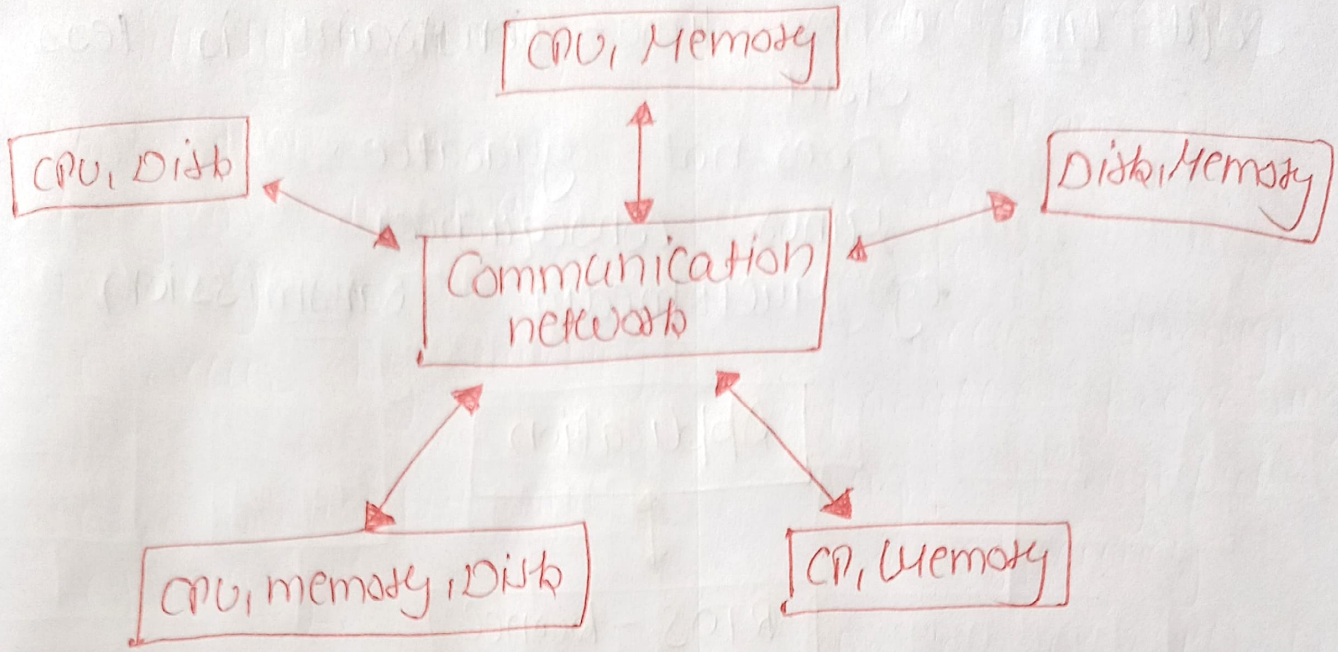
5) Distributed Operating System

when many computers are interconnected to each other through a network for the purpose of sharing their tasks.

- referred as loosely coupled system.

node are independent

own memory



Advantage:
- failure of one will not affect the other communication
- load on host computer reduces.

Disadvantage:
- failure of main network will stop the entire communication.

Example: LOCUS.

Real - Time OS: } when we required a quick response ex:- missile system, robots etc.

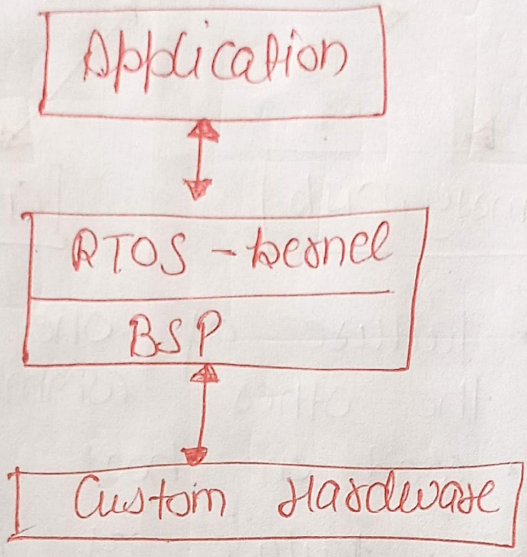
- Respone time: time interval to process and respond to input is very small.

→ Types of OS

- Hard :→ Time Constraints are strict
  - response time is millisecond.
  - Virtual Memory found.
  - Guarantee response within a specific deadline.

Scanned with OKEN Scanner

• Soft-Time: where time-constraint is less strict.
- Does not guarantee.
- Minor consequences
eg: Multimedia Transmission



Advantage: Maximum Consumption
- Error-free.
- Memory allocation

Disadvantage: Limited Tasks
- Complex algorithm
- Use heavy system

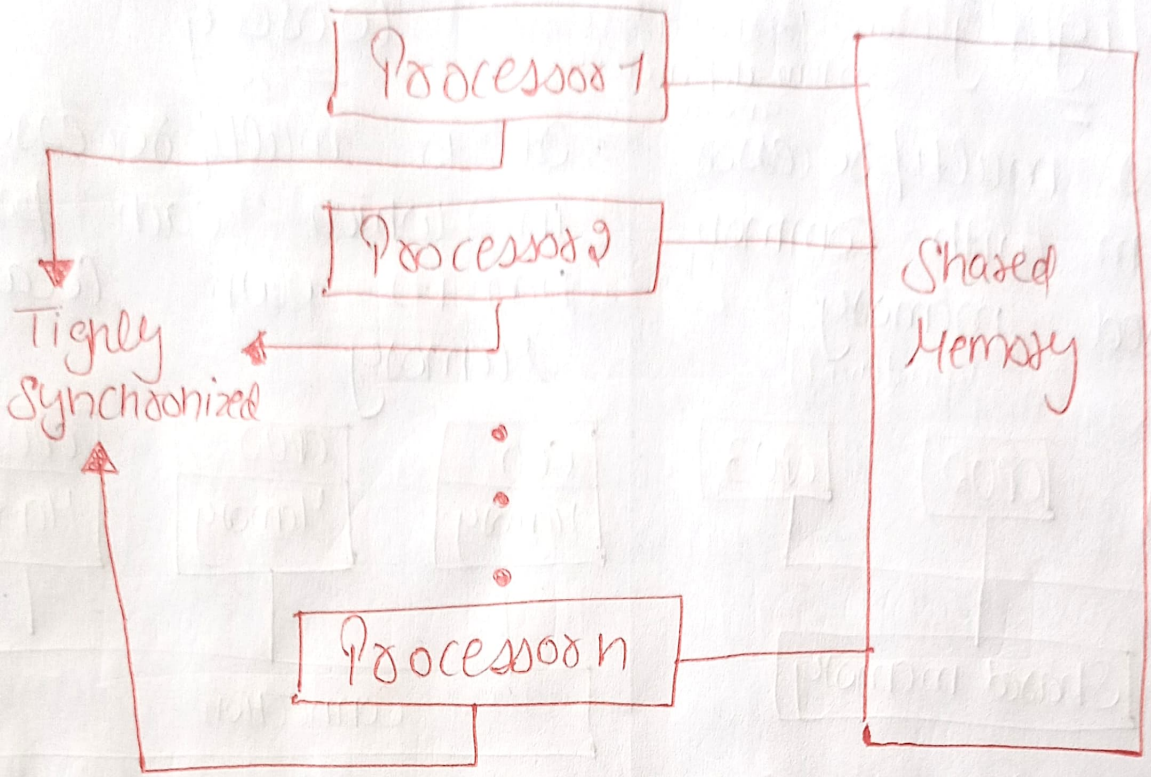
Ex: weapon system, air traffic control,

---

1) Parallel OS: → Tighly coupled system and Multiprocessor system
- It is designed to speed up the execution process of programs by dividing them into multiple segments.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner



Types :-> 1) Type-1 [act as native hypervisor]
2) Type-2 [hosted on hypervisor]

Application: • Database and Data mining
• Advanced graphics
• Argument reality

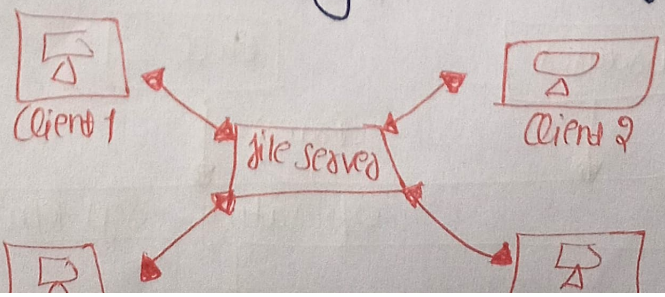
Ex: VMware, Oracle VM.

• Main reason → to execute virtual machine having different purposes

---

8) Network OS: → These system run on a server and provide the capability to manage data, users.

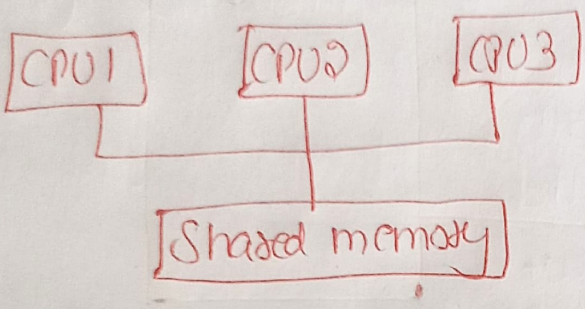
• Tighly-coupled OS



Tightly

[how much one part of a system depends on another]

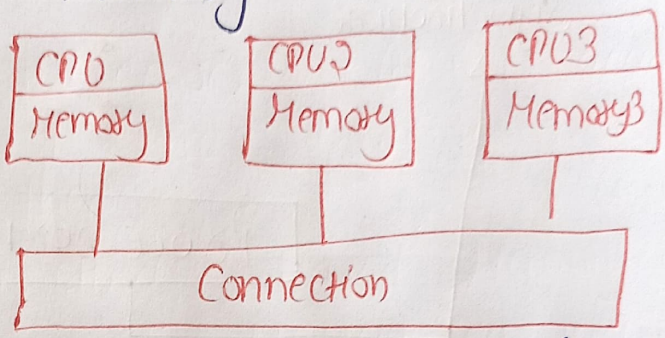
- It is multiprocessor system with common shared memory [depend on them]



- The information can be shared memory among the CPU by just placing in it
- Used in parallel processing
- Data are high.
- less flexible

Loosely

- It is multiprocessor in which each processor has its own local memory.



- Information is transferred from one processor to other by using message passing techn.
- Used in distributed computing system.
- Data rate is low.
- More flexible.

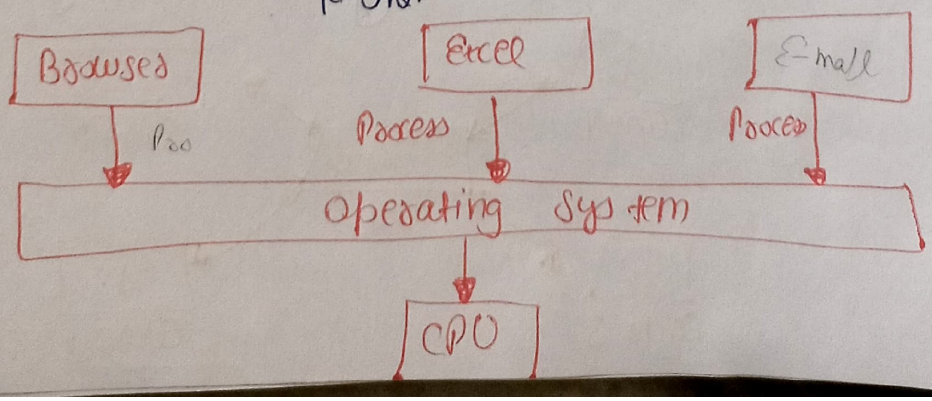
Mult-Tasking OS:

allow multiple user to perform multiple tasks at the same time.

- Time sharing
- Context switching

1, Closely related to Time sharing.

- More than one used work together to complete one task.



Scanned with OKEN Scanner

- **Preemptive Mult-tasking**: - The mechanism of tasking control of the operating System from one process and giving it to another process

the OS system can initiate a context switches from the running process to another process.

Windows

→ Round Robin. Scheduling

- **Cooperative Mult-tasking**: The OS Never initiates context switching from running process to another process.

eg: MAC OS.

OS Services

- provide services both user and program

→ **Program execution**

- load a program into memory
- Executes the program
- Handles program execution
- Provide mechanism for process synchronization, deadlock handling.

→ **I/O Operation**

- I/O operation means read or write operation with any file
- OS provide the access to required I/O device.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

3) File System Manipulation:

↓
Collection of related info.

- Program need to read a file or write
- backup of file.

4) Communication:

- Two processes often require data to be transferred b/w them
- two method:
  - Shared Memory
  - Message Passing

5) Error handling

- error can occur anytime and anywhere.

→ OS constantly checks for possible errors.

6) Resource Management: like CPU

- OS manage all kinds of resources using schedulers
- CPU scheduling alg. are used for better utilizati of CPU

7) Protection

[Without OS computer is useless → It manage the computer Memory and process.
→ also communicate

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

System Call

- System call is a programmatic way in which a computer program requests a service from the kernel of OS
- It provide the services of OS to user program via Application program interface.
- System call define the interface b/w the services and the user process provided by OS.

Features

- Interface
- Protection
- kernel Mode
- error handling

Advantage

- Access to hardware resources
- Memory Management
- Security

How it work

- User need special resource
- Program make a system call request
- OS see the System Call
- OS perform operation.

Scanned with OKEN Scanner

Et: System call in Window & Unit

- Process Control: responsible for creating, terminate, allocated process.

Window	Unit
Create process()	fork() → process to create objs
Exit process()	Exit()

- File Manipulation: responsible for file Manipulation jobs,

Window	Unit
Create file()	open()
Read file()	Read()
Write file()	Write()

- Device Management: responsible for device Management

Window	Unit
Set console mode()	IOCTL()
Reed Console ()	Read()
Write Console()	Write()

- Information Maintenace : handle for infon and its tranfer b/w OS. and manage data.

Window	Unit
Get Process ()	Getpid()
Set Time()	Alarm()
Sleep()	Sleep()

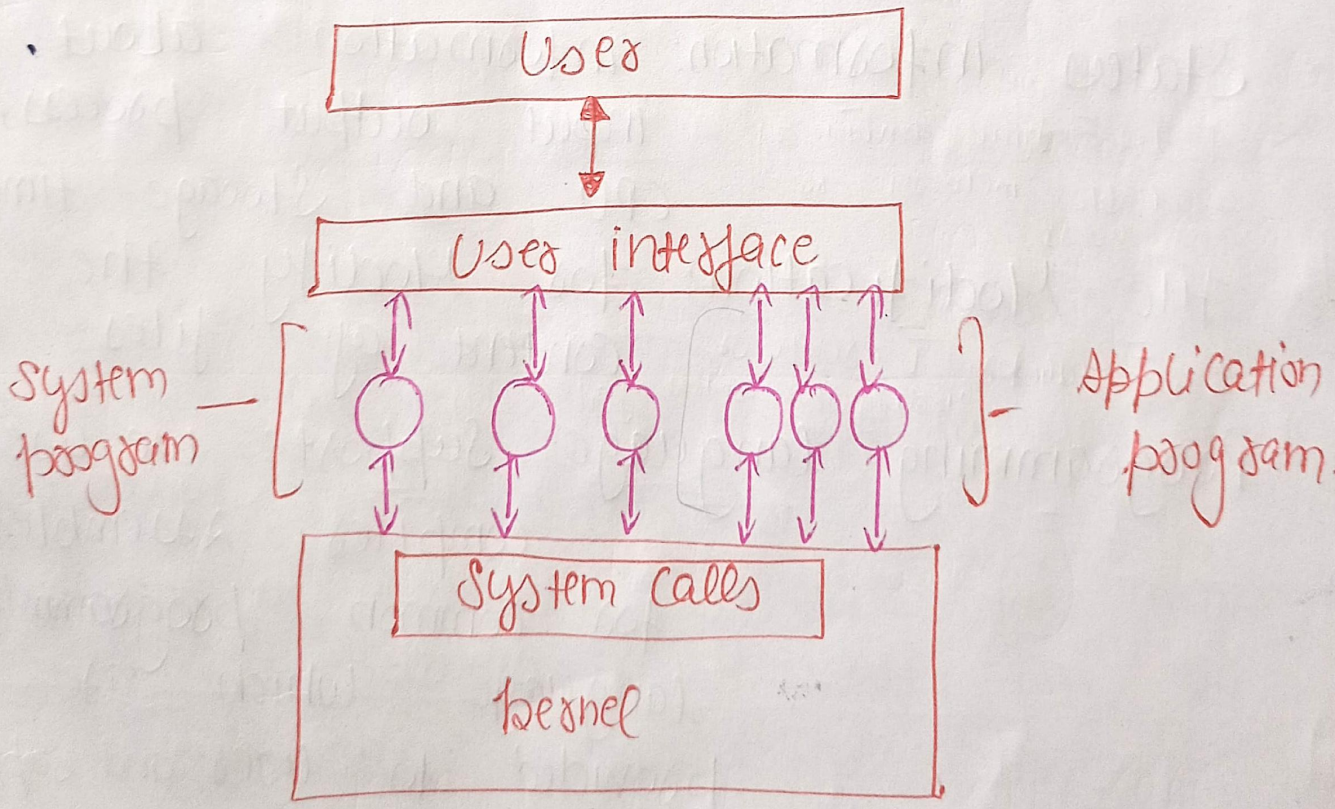
Scanned with OKEN Scanner

- Communication :- Useful for communication

Window	Unix
Createpipe()	Pipe()
CreateFileMapping()	Shmget()

System program

- provide a interface.
- It offers a environment in which program may be develop and run.
- Some system program are only user interface and other are complex.
-



- The user can only view up-to-the system program he can't see system calls.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

- File Management: collection of specific information stored in the memory of computer system.
  [create, delete, copy, rename]
  - Manipulates file and its manage [creation, modify]
- Command line interface: essential tool for user
  - provide user facility to write commands directly
- Device Drivers: work as simple translat for OS and devices
  [OS & devices]
- States information: information about input output process, CPU and storage time.
  [date, time, amount of available memory, no of users]
- File Modification: for Modify the content of files.
  [Text editor to create or modify the content of file]
- Programming language Support: we use Compiler, Assembler for common programming language which are provided to user and support
- Program loading & Execution: The program must be loaded in the memory for eacution.
  ↓ part of OS responsible for loading program.

[Communications]

Unit-2

Copyrighted by Codeleap.io
All rights reserved.

Process Management

It manges the allocation of process to a processor

Process

- A process is a program in execution
- A single program does not have complete control of system, then gave to rise the concept of "process"

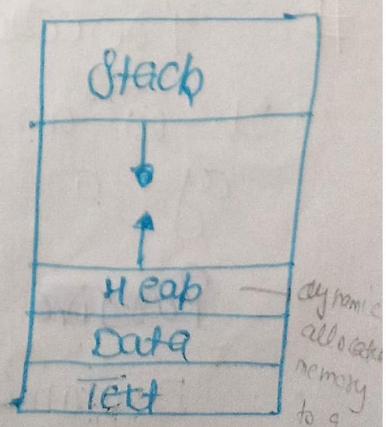
In Multiple program environment

eg: Text editor program running on a computer.

Need

- It manage and execute tasks efficiently on computer
  - handle Multiple tasks
  - Resource allocation
- When a program is loaded into memory, and it becomes a process, can be divided into four section.

→ Text Section: A process, sometime known as Text section, also include the current activity represented by the value of program counter



Scanned with OKEN Scanner

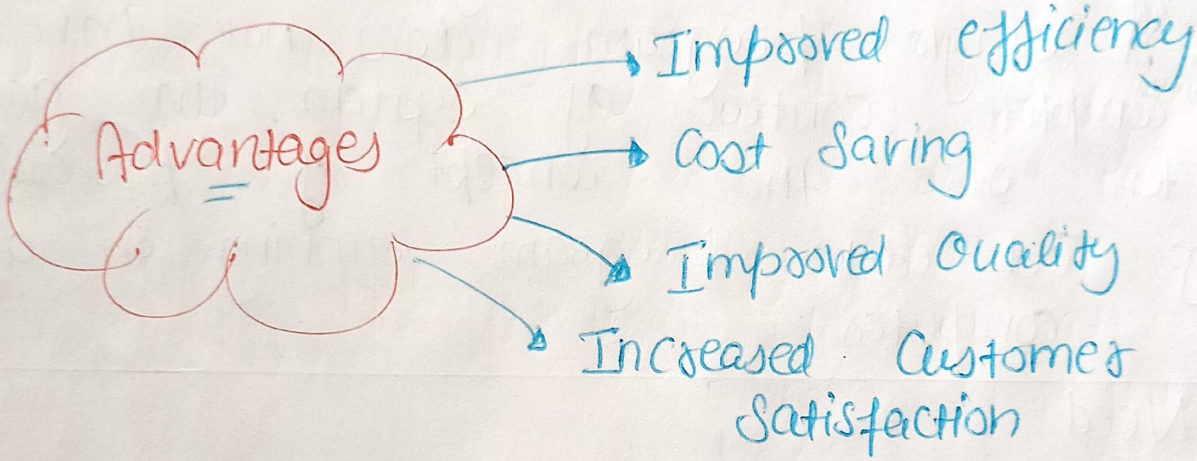
- Stack: It contain temporary data, function.
- Data Section: Contains global variable

Attributes of Process

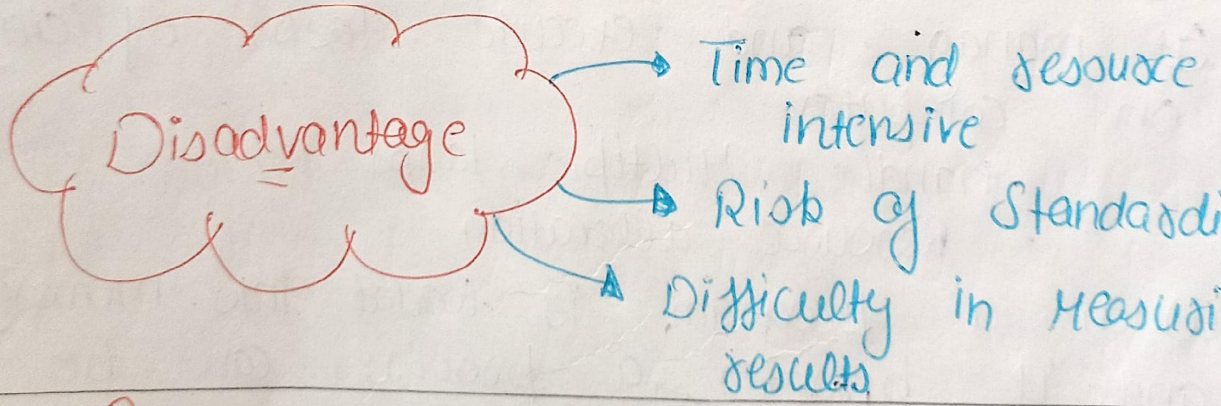
- Process id
- Process State
- CPU register
- Account information

Copyrighted by Codeleap.io
All rights reserved.

#



#



Process

- It is an execution of a program
- Active entity
- Dynamic entity (status) when its running

Program

- It contains set of instruction to complete task
- Passive entity (Secondary memory)
- Static entity

Scanned with OKEN Scanner

Process	Program
• High resource requirement	• not any resource requirement only require memory.
• It has it own control block	• does not have any control block.
• Sequence of instructio execution	• Contains instruction

Process State and their Transitions

(the current and)

- as a process executes, it change State
- State of a process is defined in part by the current activity of that process.

→ New: (The process is being created)

→ Ready: (The process is waiting to be assigned to a processor)

- after the creation of process, the process enter the ready State ie the process loaded into main memory.

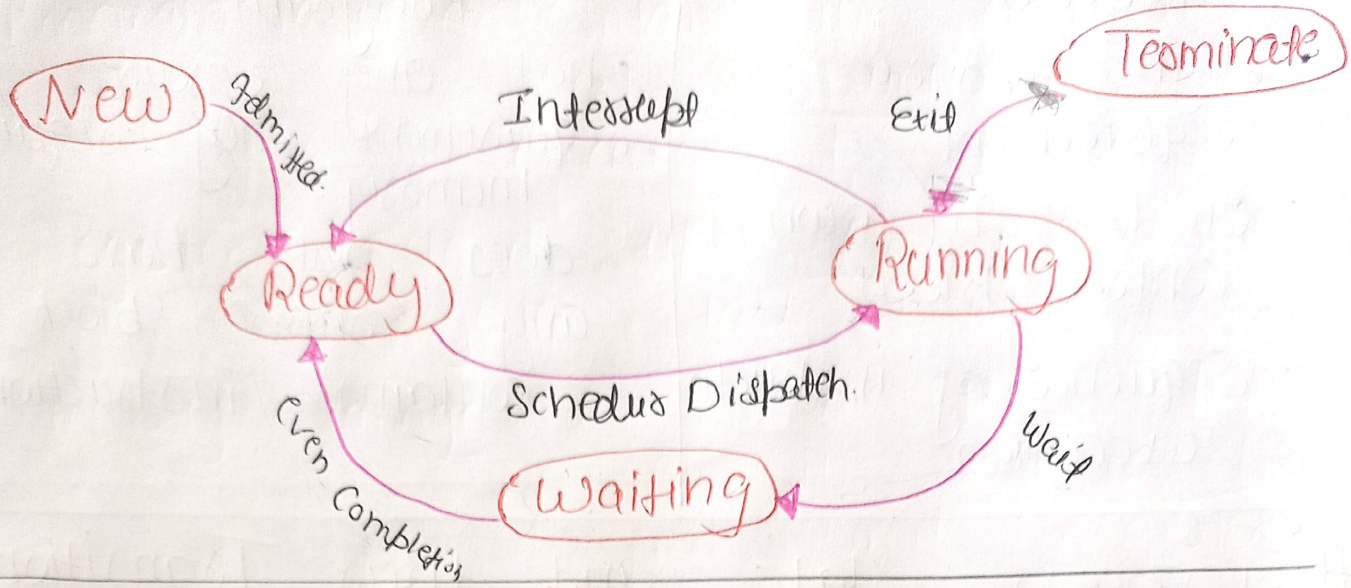
→ Running: (Instruction are being executed)

- The CPU is currently allocated to process and the process is in execution.

→ Waiting: The process is waiting for some event to occur.

→ Terminated: The process has finished execution.

Scanned with OKEN Scanner



=> Some transitions [various events b/w states]

1) Ready to Running :- A process is select out of ready queue and dispatched to running state

2) Running to Waiting: I/O operation reason for this type of process is waiting for I/O completions and do not need CPU for executing.

3) Waiting to Ready: The resons for a process to remain in waiting state is no longer applicable.

4) Running to Ready: The running process is pre-empted because some other process of higher priority has become ready.

5) Running to terminated

Process Control Block

- It is data structure containing certain information about the process.
- Each process has its own PCB to represent it in the operating system.
- PCB is a central store of information that allows the OS to locate all key information.

Process State	→ current state [New, ready]
Process Number	
Process Counter	→ contain address
Register [IP, SP]	→ depend upon CS
Memory limit	→ information include value of base.
List of open file	
⋮	

- Process State: about the current state of process i.e new, ready, running, waiting
- Process number: each process is identified by its unique process number.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

- Program Counter: Contain the address of next instruction to be executed for process.
- Registers: Depend upon CS. include index register, stack pointer.
- Memory Management: include the information as the value of base and limit register.
- CPU Scheduling algo: includes process priority, pointer to scheduling queue.
- I/O information
- Accounting information

Operation on Process

OS that manages process must be able to perform certain operation.

- Set of operation on process include
  - Create a process
  - Destroy a process
  - Run a process
  - Suspend a process
  - Get process information
  - Set process information
- It supply Mechanism.

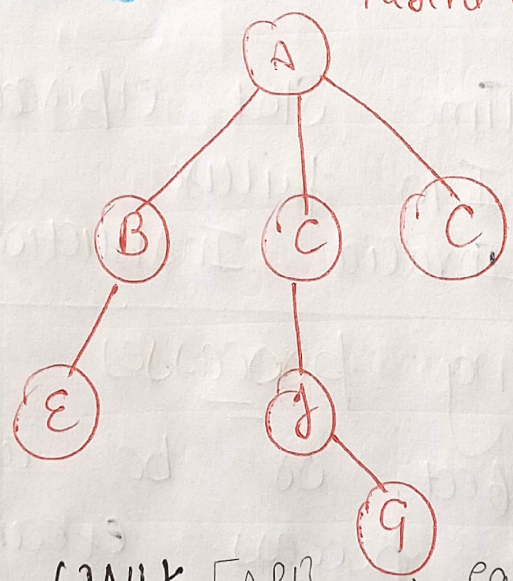
Process Creation | Process Termination

Process Creation

- Creating process by OS and involve some operation.
  - name the process
  - insert it in system
  - determine the process
  - create " "
  - allocate " "
- A process create several new process and creating process called parent process and created process called child process.

(Spawning)

Parent (only one parent need to create child)



childrens

- Done in UNIX [API] → each process has its parent.

→ Synchronous :} If a process is created from another. then the new process must comple execution before old one.

→ Asynchronous :} new process is created, then two processes may be run in parallel way.

Scanned with OKEN Scanner

Process Termination

ending of process.

→ when it finishes executing the final statement and asks the OS to delete it by using exit system call.

→ It involves returing of data to its parent process

→ All the resources of process including physical and virtual memory are deallocated by the OS.

⇒ Cause: Time Slot expired
I/O failure
Invalid Instruction.

8:30

Co-operating processes

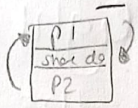
- It can affect or be affected by the other process executing in the system
- If a process cannot affected or cannot be affected by the execution of other processes called independent process.
- Any process that shares data with other process is a co-operating process.

- Reason why an environment for cooperating process are provide:
  1) information Sharing
  2) Computation Speed up
  3) Modularity [divide system function into parts]
  4) Convenience [Many works doing one time]

- Cooperation process require an interprocess communication mechanism that will allow them to exchange data and information

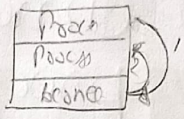
Methods of Cooperation | Model of IPC

- By Sharing



→ Produces - Consumer Problem [Buffer Counter] → Shared resources

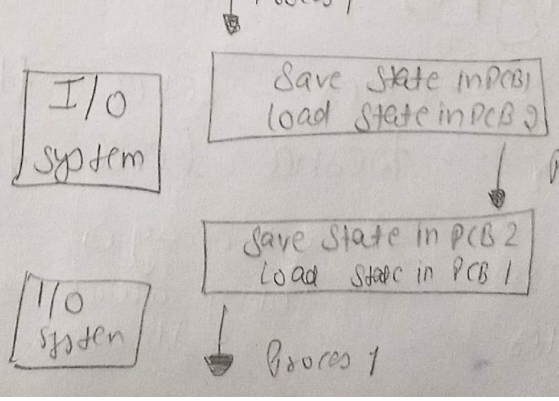
- By Communication [Message Passing]



→ Starvation [Deadlock waiting] [do not received]

Context Switch

- The process of saving the context of one process and loading the context of another process.
- It like a loading and unloading the process from the running state to the ready state.
- Interrupts, Multitasking, kernel are the triggers of Context switching
- represented as PCB
- Include CPU register value.
- More Complex
- Highly dependent on hardware support



Scanned with OKEN Scanner

- When a context switch occur, the kernel save the context of old process in its PCB & load the saved context of new process scheduled to run.

Threads [It has its own PC, registers set, stack]

- A thread is a single sequential flow of control within a program.
- also known as light-weight process.
- Threads share the resource of the parent process, also created within and belong to process.
- Thread itself is not a program - it cannot run on its own, but runs within a program.

=> Need of Thread [Issue → Single handling, Cancellation, Thread pools]

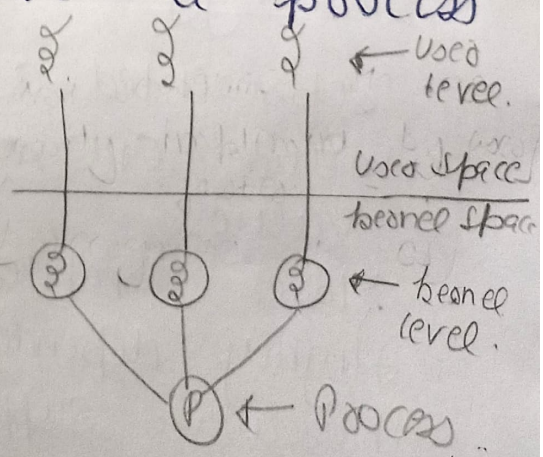
- Threads can share the common data,
- Context switching is faster when working with threads
- Take less time to create and terminate a thread than a process

=> Types

- User - level thread
- kernel level thread.

implemented by user, faster

recognize by OS, fully aware of all threads.



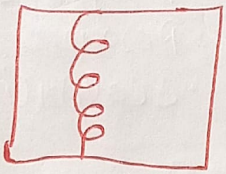
Scanned with OKEN Scanner

Process VS Threads

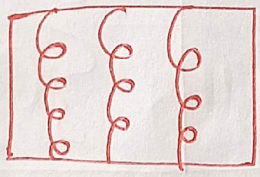
[fork() → Create new process
[exec() → run an executable file

- means any program is an execution
- The process take more time to terminate.
- It also take more time for Context Switching (system call required)
- Process are independent of each other

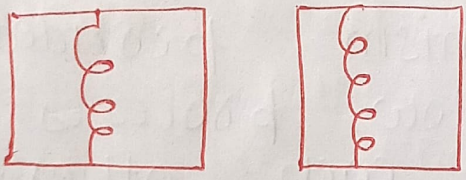
Thread means a segment of process
- The thread takes less time to terminate. (Not required)
- It take less time for Context switching
- are interdependent and share memory



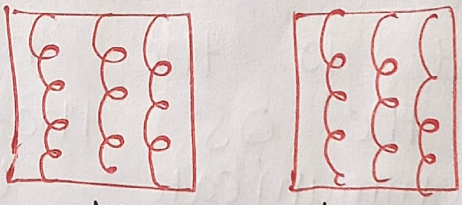
One process, One thread



One process, multiple thread



Multiple process, one thread per process (Unix)



Multiple process, Multiple thread (Linux).

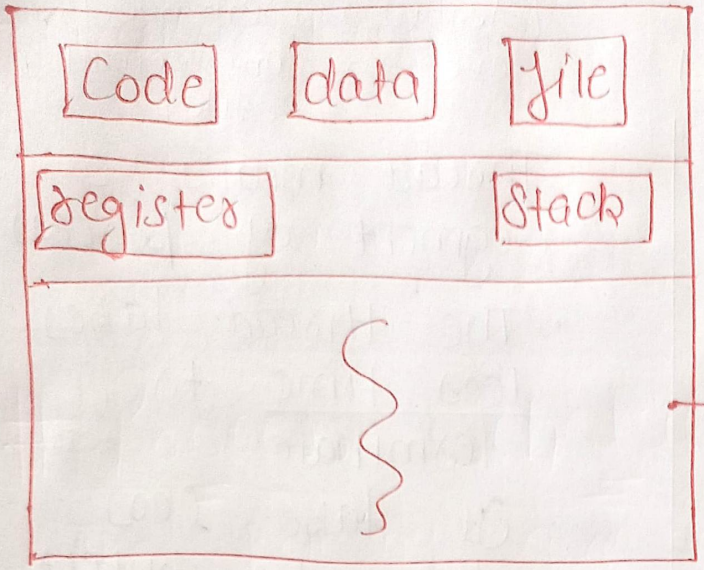
Multithreading

It allow you to execute different threads at same time.

- Window 95. Multithreaded.
- Multiple tasks to be performed to overcome the drawback of single-threading

[Use]

Scanned with OKEN Scanner

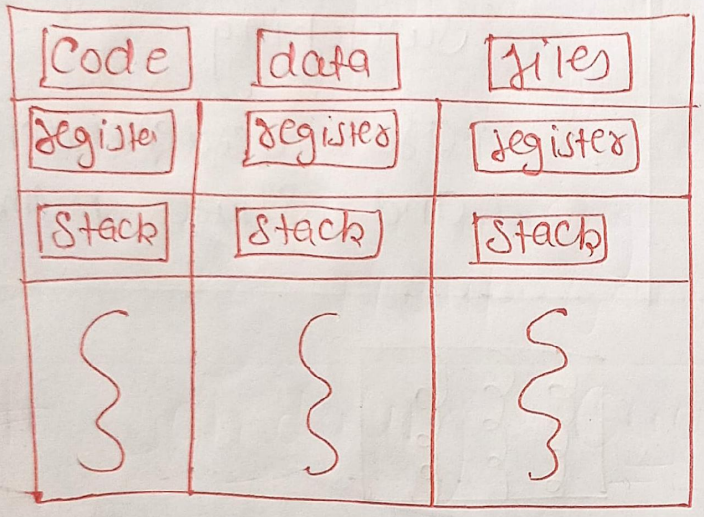


[only one task can be performed at a time]

→ Single thread.

Many to one
one to one
Many to Many

Copyrighted by Codeleap.io
All rights reserved.



→ Multithreaded.

- improve the performance of CS.
- Responsiveness
- Resource Sharing
- Scalability

Inter-process Communication

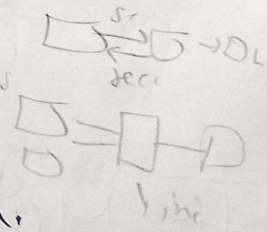
[operation
- sender
- recive]

It is the mechanism provided by the OS that allow processes to communicate with each other and to synchronize their actions

Process 1 ↔ Process 2

↓

interprocess communication.



→ Pipes → data chance. [unidir]

→ Shared Memory → two process commu → [send & recive] → link associated.

→ Direct Communication [bidi]

Unidir → Indirect [uni, bi] " [Message are directed and received from mailboxes]

→ Message Passing

Approaches

Categories
- Independent
- Cooperating

• Synchronization in IPC

- essential part of IPC
- It provided by IPC control mechanism, but sometimes it can also be controlled by communication process

1) Mutual Exclusion
2) Semaphore (type of variable)
3) Barrier

Method that used to provide the synchronization

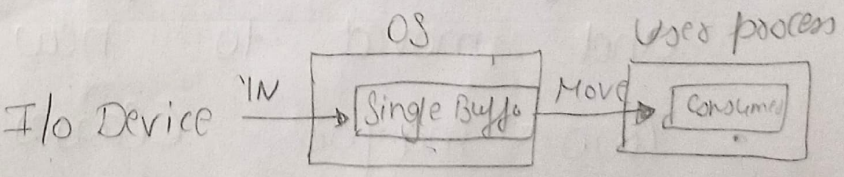
- **Blocking Protocol:** sender process is blocked until msg is delivered to reciever
- **Non-Blocking:** sender to continue execution wheather msg is delivered.

Buffering

Buffering is a technique which is used to enhance the performance of I/O operations of the system.

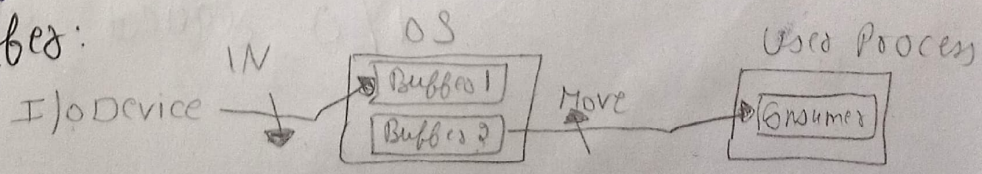
- Buffer is an area in the main memory used to store the data temporarily.

→ **Single Buffer:**



only one buffer is used to transfer the data b/w two devices.

→ **Double Buffer:**

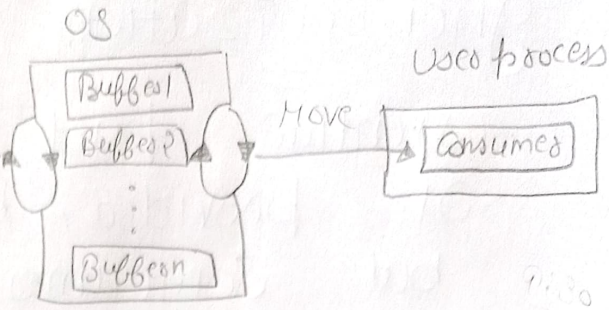


Scanned with OKEN Scanner

→ Circular Buffer

Operation
- Steam oriented
- Block -oriented

> operation



Process Scheduling

The primary goal of process scheduling is to optimize the utilization of CPU
- also Manage the execution of multiple processes in computer system.
- Process Scheduling is an essential part of a Multiprogramming OS.

Scheduling Queues

← manage the execution of process.

- The OS maintains all PCB in Process Scheduling Queue.
- The OS maintains a separate Queue for each of process states and PCB of all processes in same execution state
- If the state of process changed, PCB is Unlinked from current state and moved to new state queue.
- Two major Classes of queue in OS
  - → Ready Queue
  - → I/O request queue.

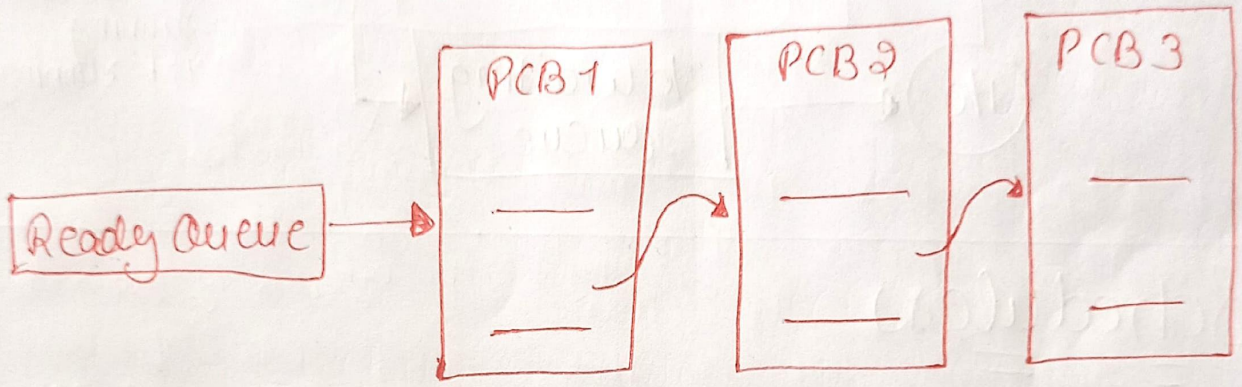
Scanned with OKEN Scanner

- **Job queue**: This queue keep all the process in the system

- **Ready Queue**: The ready queue contain all the process that are ready to execute and waiting for CPU

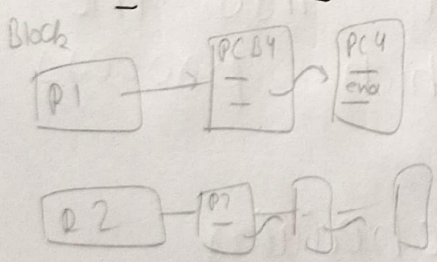
Copyrighted by Codeleap.io
All rights reserved.

  - each process represent by PCB.
  - stored in linked list



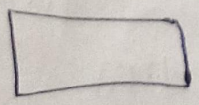
  - each PCB has a pointer filed that points to next PCB.

- **I/O request queue**: This queue consists of the process that are waiting for a particular I/O device or another resource called device queue or blocked queue.



  - each resource has its own queue.

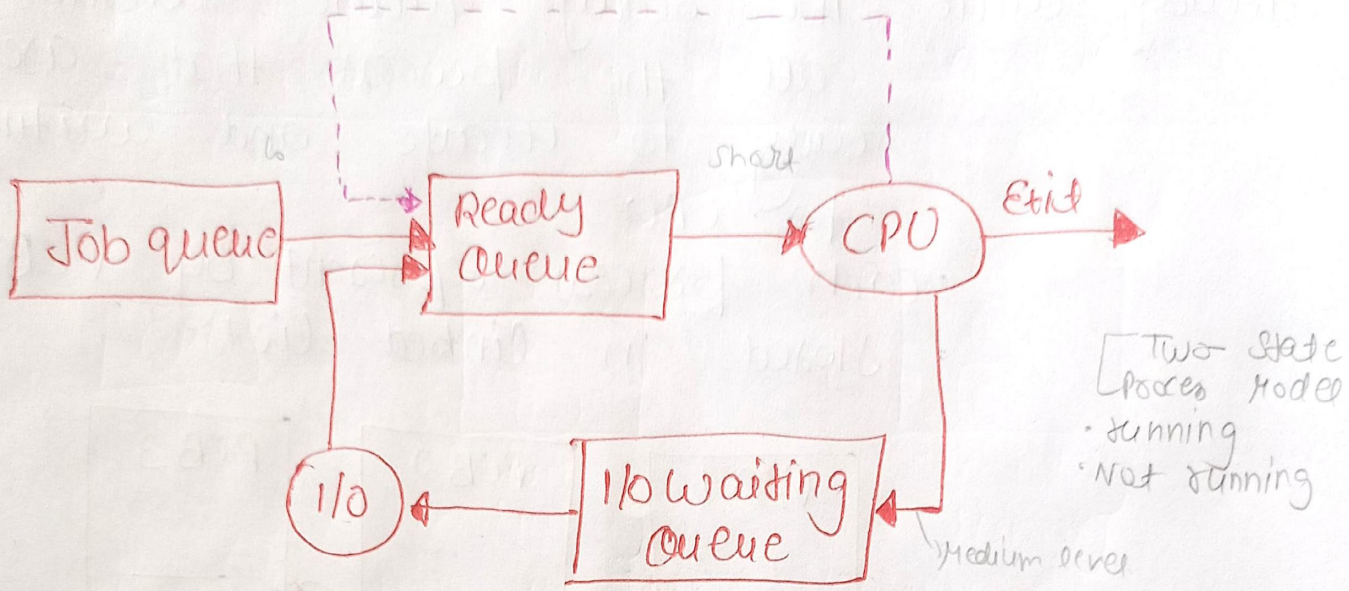
- **representation of process scheduling Queue**

 → represents a queue

→ represents the resource that serve the queue

Scanned with OKEN Scanner

→ : indicate flow of proccess in system.



Schedulers

← Manag & control execution of process

- Schedulers are special system software which handle process scheduling in various ways.
- Their main task is to select the jobs to be submitted into the system and to decide which process to run.

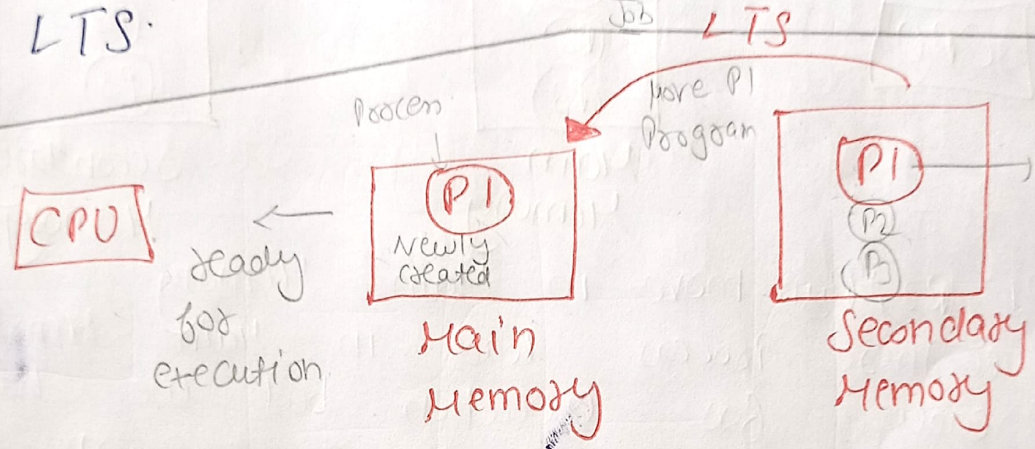
Copyrighted by Codeleap.io
All rights reserved.

- Long - Term Scheduler
- Short - Term Scheduler
- Medium - Term Scheduler

- a scheduler is a decision maker that select the process from one scheduling queue to another or allocates CPU for execution.

Scanned with OKEN Scanner

- **Long Term Scheduler:** → also called job Schedular / High-level.
- It select process from the queue and load into memory for execution. When the process change the state from new to ready, then there is use of LTS.

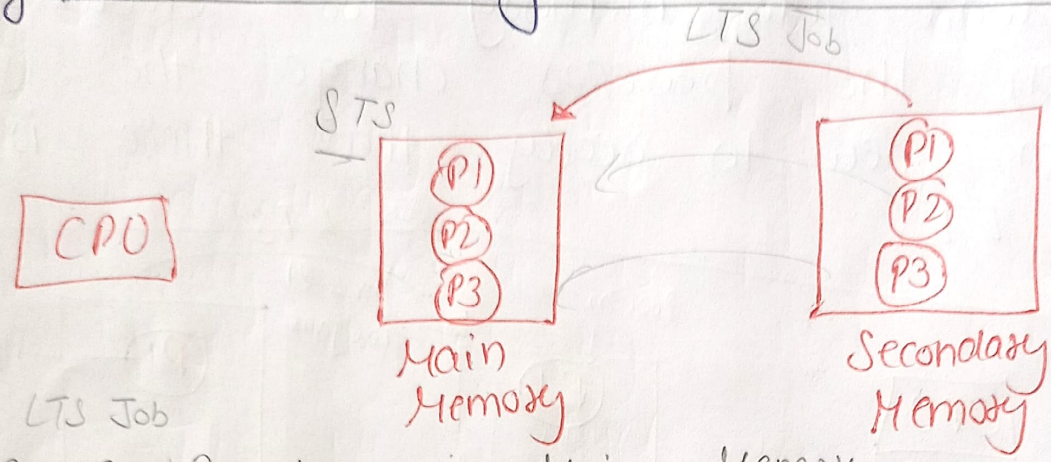


- It load all the process in main memory for execution [P1 load into Main]
- Select process from Secondary Memory and keep in primary Memory for ready to execution [for CPU Scheduling].

- Control the degree of Multiprogramming.
- Because of longer interval b/w execution, the LTS can afford to take more time to select a process for execution.

- **Short - Term Scheduler:** → also known as CPU Scheduler / Low-level.
- It select a process from among the processes that are ready to execute and allocate the CPU

- Short-term scheduler, also known as dispatcher, make the decision of which process to execute next. [loading the process selected by STS]
- faster than long-term schedulers



- P1, P2, P3 move in Main Memory
- all these process load in Main Memory and ready for execution but we have only one CPU at a time, only one process get a chance to execution. [CPU]

- Job of STS is, it select one of these process from main memory and allocate CPU. [P1, P2, P3 → decide which process give to first for CPU] [STS]
- [Decide] assume P2 assign, for the allocation of CPU, it take first. [control multitasking]
- Run in milliseconds

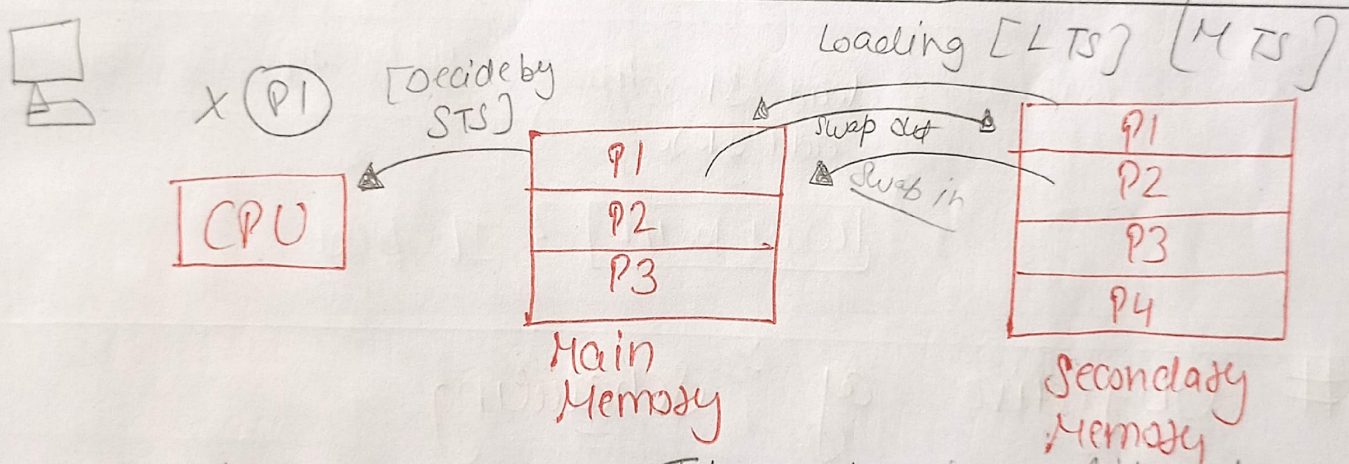
3) Medium level Scheduler: } part of swapping

SM to MM
swap

- It remove the process from the memory.
- reduces the degree of Multiprogramming.
- It is in-chase of handling the

Swapped out - processes.

- A running process may become suspend if it make an I/O request
- A suspended process cannot make any progress towards completion.
- In this condn, to remove the process from memory and make space for other process, the suspend proces is moved to storage (secondary) called swapping.
- Swapping in and out done by MTS



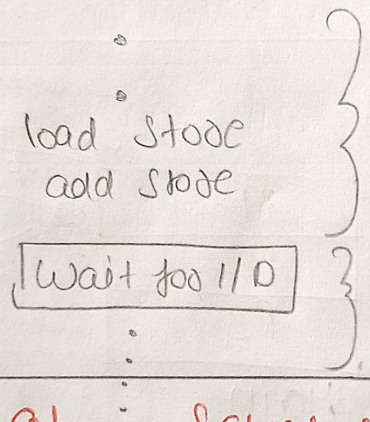
- In b/w P1 want I/O operation like keyword.
- It leave the CPU after swapped out
  [It exists keyword in b/w process when the keyword busy called swapped out. P1 It can't wait in main memory, it suspend to secondary memory. Then in Main Memory space left for other process.
- When keyword free than again load memory into main.

CPU Scheduling

← Manage the execution of process on computer
→ Maximize CPU utilization

- The CPU Scheduling is the process by which a process is executed by using the resource of CPU
- CPU scheduling deal with the problem of deciding which of process in ready queue is to be allocated CPU first, decided by Short-term Schedule

⇒ CPU-I/O Burst Cycle



Objective of Scheduling

- Maximize the system throughput
- be 'fair' to all user
- be consistent
- provide tolerable response.

Scheduling Criteria

- "How good a scheduling algorithm
- CPU Utilization
- Throughput
- Turnaround Time
- Waiting Time
- response Time.

Scanned with OKEN Scanner

- CPU Utilisation: CPU should remain as busy as possible.
  = [Max]

- Throughput: It refers to the amount of work completed in a unit of time should be maximized.
  [Dependent on job]

- Turnaround time: defined as interval from the time of submission of a process to the time of its completion. Should be less as possible.
  TA = Completion time − Arrival Time

- Waiting Time: amount of time a process waiting in ready queue.
  - This time as less as possible for process.

Waiting Time = Turnaround − Burst Tim

- Response Time: It is the best metric.
  - Time interval b/w the job submission and first response produced by job.
  - should be minimed in interactive system.

Some Terminologies CPU algo

- Arrival Time
- Completion Time
- Burst Time [by a process for CPU execution]
- Turn Around Time
- Waiting Time.

Scanned with OKEN Scanner

Types

Copyrighted by Codeleap.io
All rights reserved.

- **Preemptive Scheduling**: It is used when a process switches from running state to ready state. [interrupted or waiting to ready state. (Completed to I/O)]
  - Its resources allocated to the process for limited amount of time.
  - alg. include: - Round robin
    Priority scheduling
    Shortest remaing Time first

eg:

Process	Arrival time	CPU Burst Time
P_0	3	2
P_1	2	4
P_2	0	6
P_3	1	4

P_2	P_3	P_0	P_1	P_2

0 1 5 7 11 16

[flexible]

- The process is again placed back in the ready queue if that process still has CPU burst time remaining
- If process with high priority arrive the current running process is interrupted.

Scanned with OKEN Scanner

- **Non-Primitive** :- It is used when a process termination or when a process switches from running state to waiting state. (I/O request)

  - Once the resource is allocated to process, the process hold CPU till, it get terminated or it reaches a waiting state.

eg :- shortest job first, first come first serve.

[rigid]

P_2	P_3	P_1	P_0

0 6 10 14 16

- It is not interrupted even with high priority.

Dispatcher

- another component involved in CPU scheduling
- Dispatcher is a program responsible for assigning the CPU to process.
- Selected by short-term scheduler.
- Assign the CPU to ready state involve steps
  - → Context switching
  - → Switching to user mode
  - → Restart the execution of process

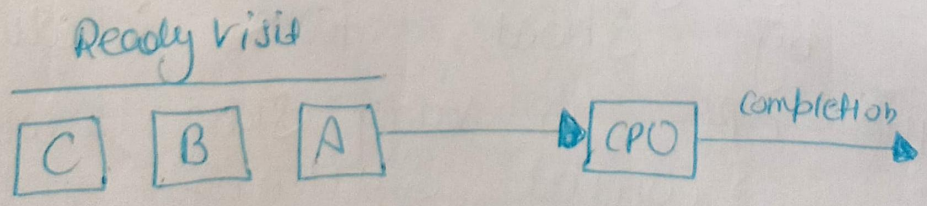
Scanned with OKEN Scanner

#CPU Scheduling algorithm

- It deals with the problem of deciding which of the process in the ready queue is to allocated CPU first.
- It allocated by dispatcher.
- Max Utilization of CPU.

⇒ First-Come-First Serve

- Jobs are executed on first come, first serve Basis
- It is a non-preemptive scheduling algorithm.
- easy to understand
- "allocate the CPU in order in which the process arrive".
- managed as FIFO queue, when the CPU is free, it allocated to proce, which is occuping in front of queue.
- once process go into running state, its PCB is removed from the queue.



- not much efficient in performance.
- wait time is high.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

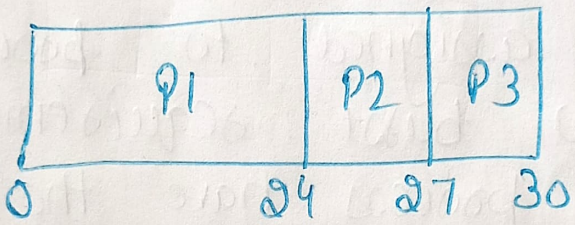
Ex:

Process	Burst Time	Waiting time
P1	24	0
P2	3	24
P3	3	27

- arrive time at : $\boxed{0}$ with length of CPU
- If process arrive in order P1, P2, P3 and served in FCFS order,

---

- Apply Gantt Chart



- Waiting time P1 = 0 milli second
  " " P2 = 24 "
  " " P3 = 27 "

- Average Waiting time $= \frac{0+24+27}{3}$

$\frac{\text{waiting Time}}{\text{No of Process}}$

$= \boxed{17\ ms}$

→ average waiting time under FCFS is very long. → It is rarely used

→ Suitable for Batch System

→ Not suitable for time sharing system.

Issue in GANNT Chart is graphical repr. of activity against time.

2) Shortest - Job - first Scheduling

- Non - preemptive algorithm.
- Best approach to minimize waiting time.
- Process have less Burst time and then execute.

"CPU is allocated to the process with least CPU - burst time"

- May suffer with problem of Starvation

→ Among the process in ready queue, CPU is assigned to process with least CPU burst requirement.

→ If two process have the same CPU burst, the one which arrived first, will be taken up first by CPU.

↓

reduce the waiting time of Job.

Ex:

Process	Burst	Waiting
P1	6ms	3
P2	8ms	16
P3	7ms	9
P4	3ms	0

- Set of process with length of CPU burst time in ms.

Scanned with OKEN Scanner

" Gantt Chart:

Copyrighted by Codeleap.io
All rights reserved.

P4	P1	P3	P2

0 3 9 16 24

Average waiting Time $= \frac{3+16+9+0}{4}$

$= \boxed{7ms}$

- It is optimal.

---

3) Priority Scheduling

← either both.
→ Static priority [Not change
→ Dynamic priority [Change at regular interval

- each process is assign is priority
- The process with the highest priority is scheduled to run first.
- Problem with Priority: ?, indefinite blocking or Starvation of a process. [low-priority may never execute]

: Preemptive: running process can be interrupted and moved back to ready queue by higher priority process

available for execution

Non → not be interrupted.
[untill complete.]
↓
continue execution

Solution: → Aging [increase the time progress of priority process low-priority]
↓
aging time increase at one moment of time it become high prior and get executed

Ex:-

Process	Burst time	Priority	Waiting Time
P1	10	3	6
P2	1	1	0
P3	2	4	16
P4	1	5	18
P5	5	2	1

Scanned with OKEN Scanner

P2	P5	P1	P3	P4

0 1 6 7 16 18 19

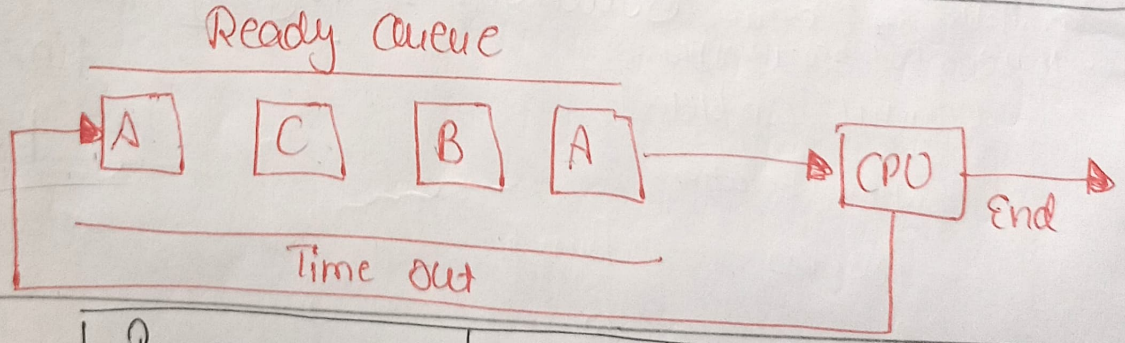
$$AT = \frac{(6+0+16+18+1)}{5}$$

$$= 8.2 \text{ ms}$$

4) Round - Robin Scheduling

(dispached in FIFO)

- each process is served by CPU for a fixed amount of time known as Time Quantum.
- Cyclic in nature and no Starvation
- preemptive in nature
- RR Scheduling is effective in time - Sharing environments in which the system needs to guarantee responsible response time for users.



Ex:

Process	Burst Time	
P1	24	53 - 20
P2	3	17 + 20
P3	3	68 - 20
P4		24

Set Time = 20

= 48

24 - 20

Set Time Slic = 4ms

- "Gantt Chart"

P1	P2	P3	P1	P1	P1	P1	P1

0 4 7 10 14 18 22 26 30

→ P1 = 0 + (10 − 4) = 6 ms

→ P2 = 4 ms

→ P3 = 7 ms

Average waiting Time = 6+4+7 = $\boxed{5.66 ms}$

---

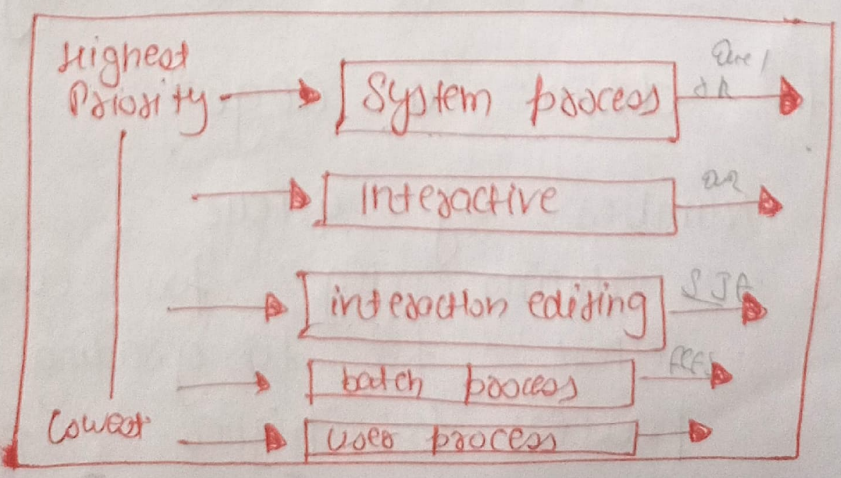
5) Multilevel Queue Scheduling

- It was created for situation in which processes are easily classified into different groups.

→ A MLQ Scheduling algorithm partions the ready queue into separate queue.

→ each queue has its own scheduling algo.

→ There must be scheduling algo b/w the queue and use fixed- priority.



← five Queues
Each queue has absolute priority over lower-priority queue.

eg :→ a common division is a foreground (interactive) process and (batch) process.

advantage

- Efficient allocation of CPU time
- Prioritization

Dis:

- inflexible in nature.

---

6) Multilevel Queue with feedback

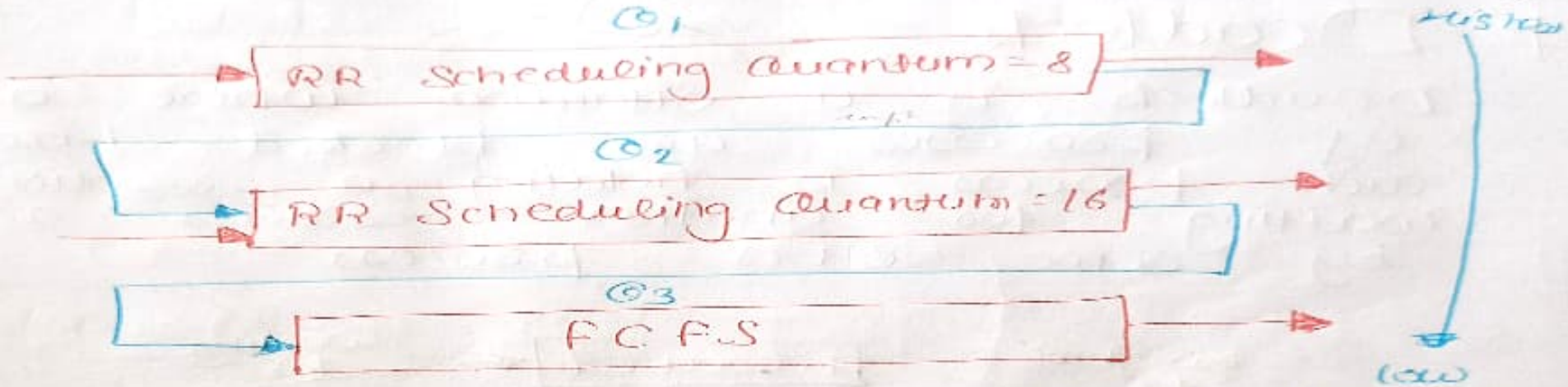
- If processes do not move b/w queues. [in Multiple-level-quese]
- but in case of Mult level feedback allow a process to move b/w queue.
- More efficient
- If a process waits too long in lower priority queue, it may be moved to highte-priority queue. ← technique called "aging".

⇒ How to define a algo?

- Number of queue
- Scheduling algo for each queue.
- Method for upgrading a process to higher-priority

- Downgrading to lower - priority.

Ex: Scheduler with three queues [Q1, Q2, Q3]

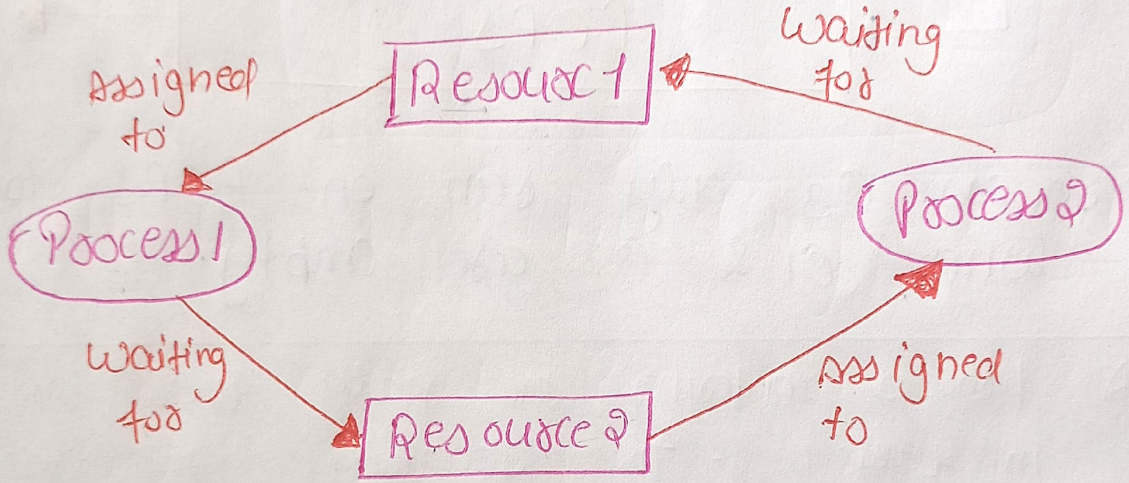


- Process Q3 only run on FCFS basis, when Q1 & Q2 are empty.

- A process entering the ready queue in put in queue 1 - The scheduler first execute to Q1 in which give time Q - 8 ms. not finish with time, moved to tail of Q2. The sch ex Q2 only when Q1 is empty.

Deadlock

- Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resources acquired by some other process.



- The resource may be I/O devices (such as printers)
- Mode of operation, a process may utilize a resource in only the following sequence.

[Syndrome]

- Request a resource
- Use the resource
- Release the resource.

Ex:→ Consider,

- When two trains are coming toward each other on the same track.
- But only once track, none of the trains can move once they are in front of each other.

//as situation occur in OS [When two or more processes that hold some resource and wait for resource held by other.

- P1 is hold Resource 1
- Wait for resource 2
- acquired by process 2
- process 2 is waiting for resource 1.

Necessary Cond^n for deadlock

- Coffman (1971) identified four cond^n.

- Mutual exclusion: → The resource are not sharable. only one process at a time, not at Multiple process

Printer → Two need

P1 → R1 ↛ P2

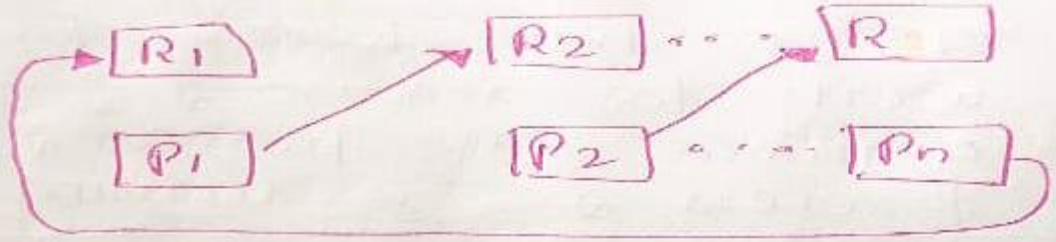
- Hold & Wait

: → a process holding at least one resource and waiting for another resource.

Scanned with OKEN Scanner

- No preemption :→ No resource can be forcibly removed from a process holding a resource can be released only by explicit action of process.

- Circular wait :→ each process waiting for resources in circular form



Methods for handling Deadlock

1) Deadlock Prevention
   - prevented by ensuring that atleast one of four condition.
2) Deadlock Avoidance
   - avoided by careful resource scheduling
3) Deadlock Detection & recovery
   - Detect Deadlock and when it occur, take steps to recover.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

Deadlock Prevention

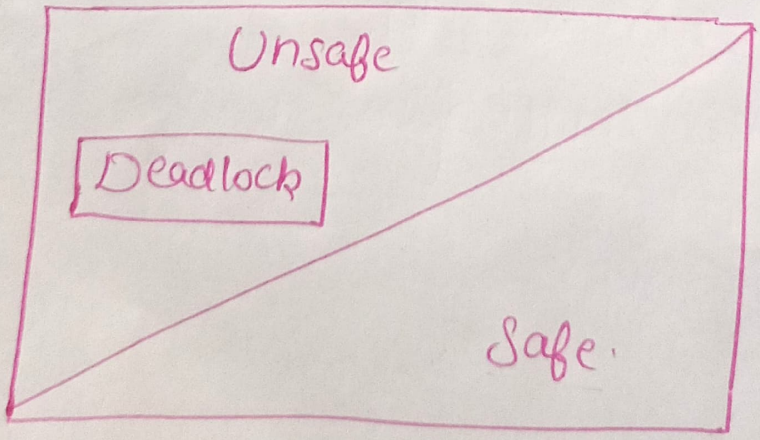
- It can be achieved by making any one cond^n false from the flowing four cond^n
  - Mutual exclusion
  - Hold and wait
  - No preemptive
  - Circular wait

at least one of conds cannot hold. Total Prevent → all cond^n ele

[idea

Deadlock Avoidance

- Deadlock Avoidance is a technique used to avoid deadlock
- To OS Check whether the system is in safe state or unsafe state. at every steps which the OS perform.
- The process continue until the system is in safe state.
- Once the system move to unsafe state, then the process must be wait



Scanned with OKEN Scanner

- **Safe State:** If deadlock not occur then Safe State.

→ all processes can complete their execution without getting into a deadlock.

→ The system must ensure that at least one sequence of resource allocation exists that leads to a safe state.

Ex:

Allocation			Max Demand			
A	B	C	A	B	C	
0	1	0	7	5	3	(P1)
2	0	0	3	2	2	(P2)
3	0	2	9	0	2	(P3)

- Three resource (A, B, C) and Three Process (P1, P2, P3)
- The system is in safe state if there exists a safe sequence P2 → P3 → P1.

- **Unsafe State:** If deadlock occur then unsafe state.
  - System not guarantee that all process will be able to complete their execution.
  - No safe sequence exist.

eg: → Circular wait creates deadlock.

- Deadlock avoidance method ensure that a system will never enter unsafe state

- One known algo. that is used for deadlock avoidance is Banker's algorithm.

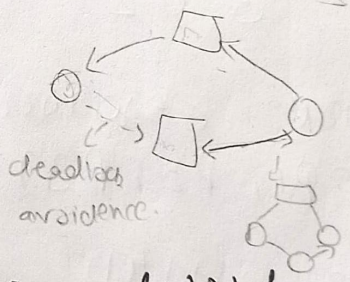
Banker's Algorithm

- The banker algorithm is an algorithm used in OS to prevent deadlocks in resource allocation
- It is based on the concept of a banker who can allocate resources to different processes in a safe and deadlock - free manner.

⇒ Data Structure used in this algo.

- Maximum Demand Matrix
- Alloc Matrix
- Avail Res
- Need Mat
- finish.

Resource allocation graph



→ only one instance of each resource type.

→ unsafe state.

⇒ How the Banker algo works

- Safety algorithm: It is used to the determine if a system is in a safe state or not.
- Resource request algorithm: It is used when a process request additional resource from system.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

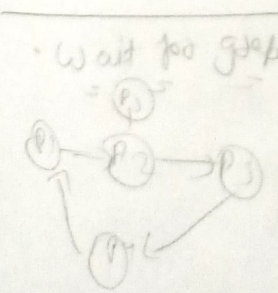
Deadlock Detection

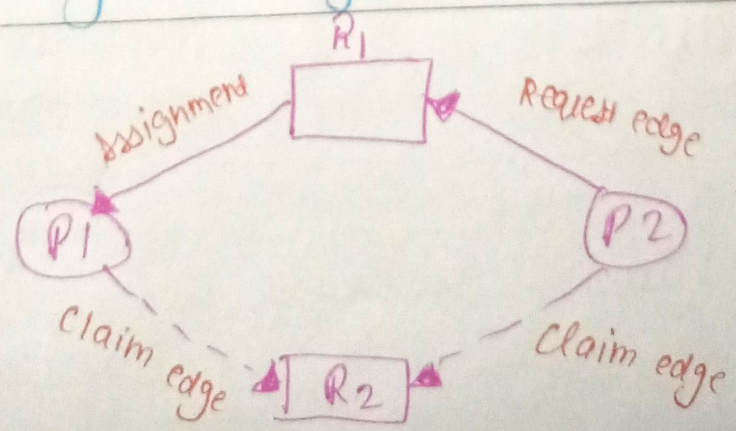
- Deadlock Detection is the process of actually determining that a deadlock has occurred in the system and identifying the processes and resource involved in deadlock.
- Once the deadlock has been detected in system, should be recovered from this state.

⇒ Resource allocation graph

- Using RAG, deadlock easily detected.
- It is a graph that represents the state of system pictorially.
  - i) Set of vertices $P = \{P_1, P_2, \ldots P_n\}$ of process, represent by circle.
  - ii) a set of vertices $R = \{R_1, R_2 \ldots R_n\}$ of resource, by squares
  - iii) request edge – directed edge $P_i \rightarrow R_j$
  - iv) assignment edge – directed edge $R_j \rightarrow P_i$

- Wait for graph





Safe State

Deadlock Recovery

- Deadlock recovery is the process of resolving a deadlock situation in OS.

→ Process Termination

To eliminate the deadlock, we can simply kill one or more process.

- about all deadlock process.
- " one process at a time and check for deadlock presence again.

→ Resource Preemption

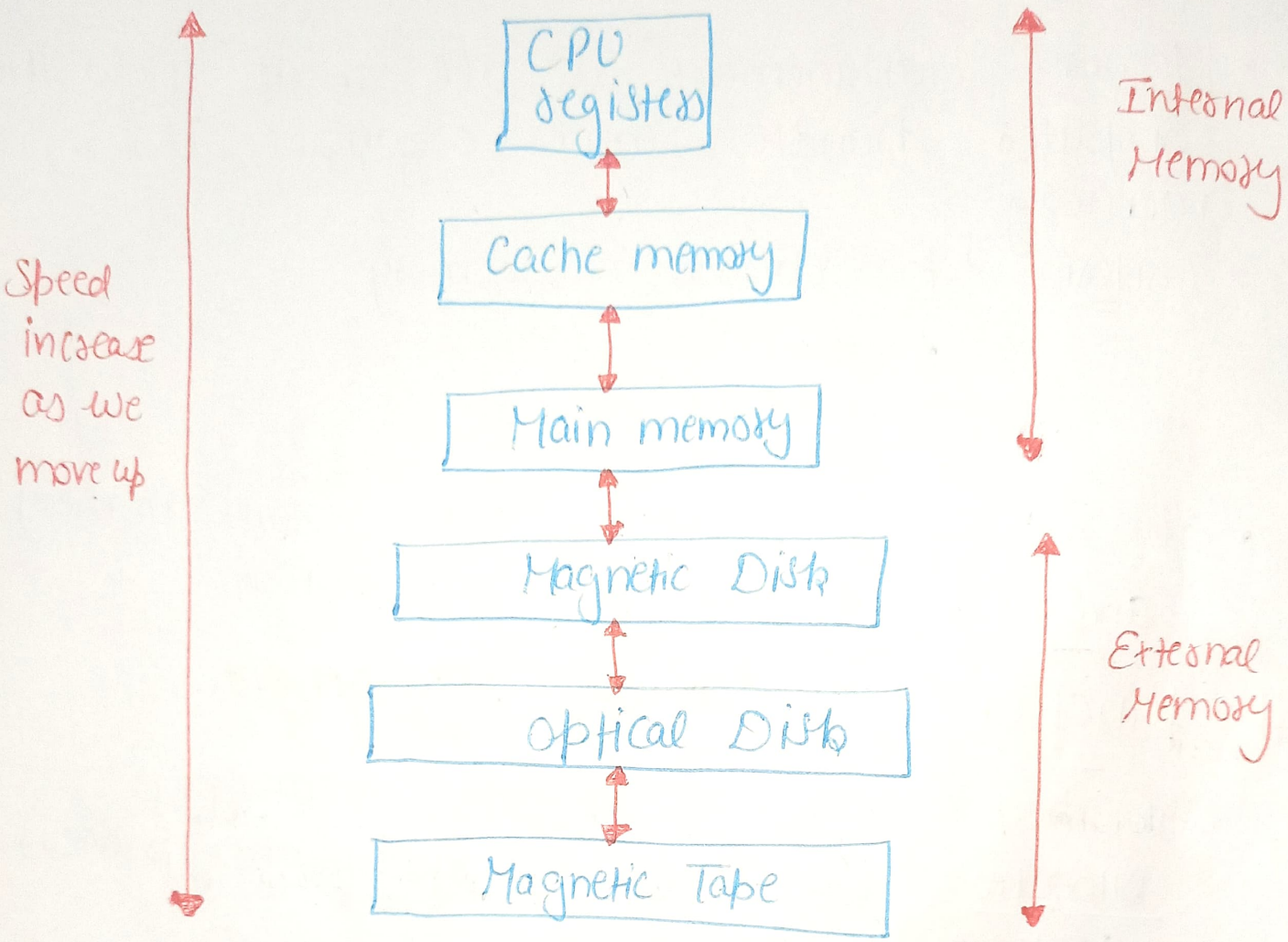
To eliminate deadlock using resource preemption, we preempt some resource from process and give to other.

→ Issue

- Selecting a victim : [determine which resource are to preempted in order to minimize cost
- rollback : [determine what should be done with the process from which resource are preempted
- Starvation: Machnism must be there which looks [when a process was for infinite time to get the resource.

Scanned with OKEN Scanner

Memory Hierachy

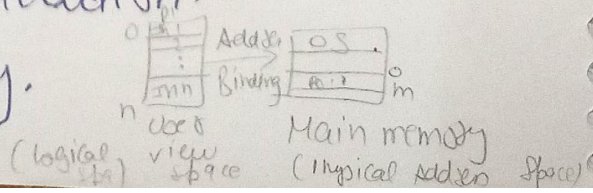


- increase access time
- increasing capacity
- Decreasing cost.

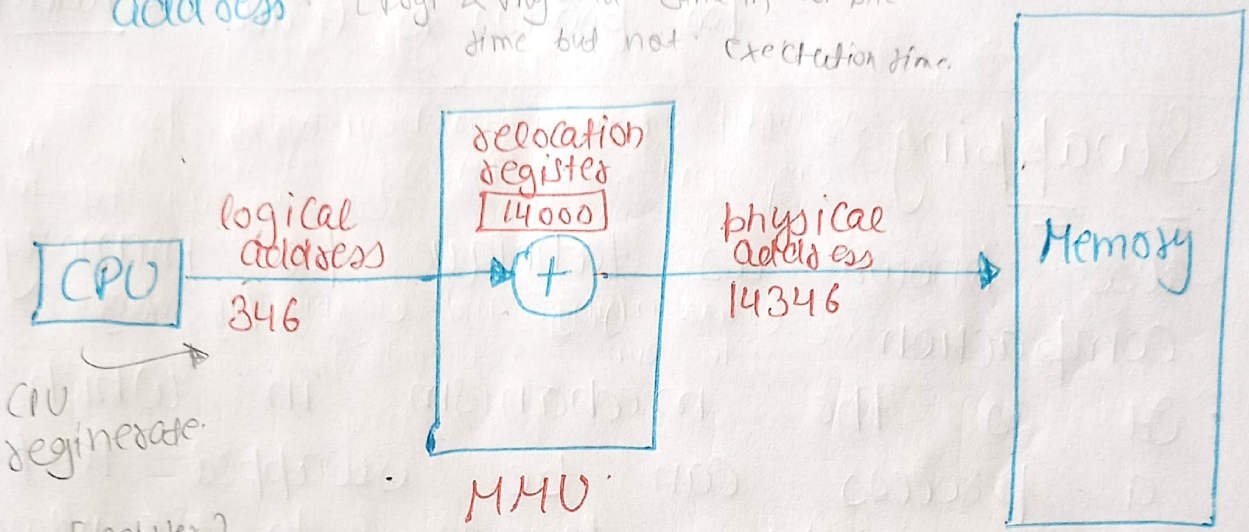
Logical vs physical Address Space

- Used to manage and access Memory.

- Logical Address :- also known as virtual address.
  - It is an address generated by CPU during program execution.
- not exist physically.



- **Physical address**: A physical address is the actual address in main memory where data is stored.
- It used by Memory Management Unit to translate logical address into physical address. [Logi & Phy are same in compile time but not execution time.



[register] 14000 value of MMU, it add with logical address then those address generate called physical address.

- represented in different format, such as binary, hexadecimal or decimal.

Program relocation

- also called "memory - management unit"
- MMU is an hardware device that maps logical address to physical address.
- The value in the relocation register is added to to every address generated by user process at time. called Program relocation & Dynamic relocation [physical address = logical address + content of relocation regis]

Static and Dynamic Loading

Static → loading the entire program into a fixed address

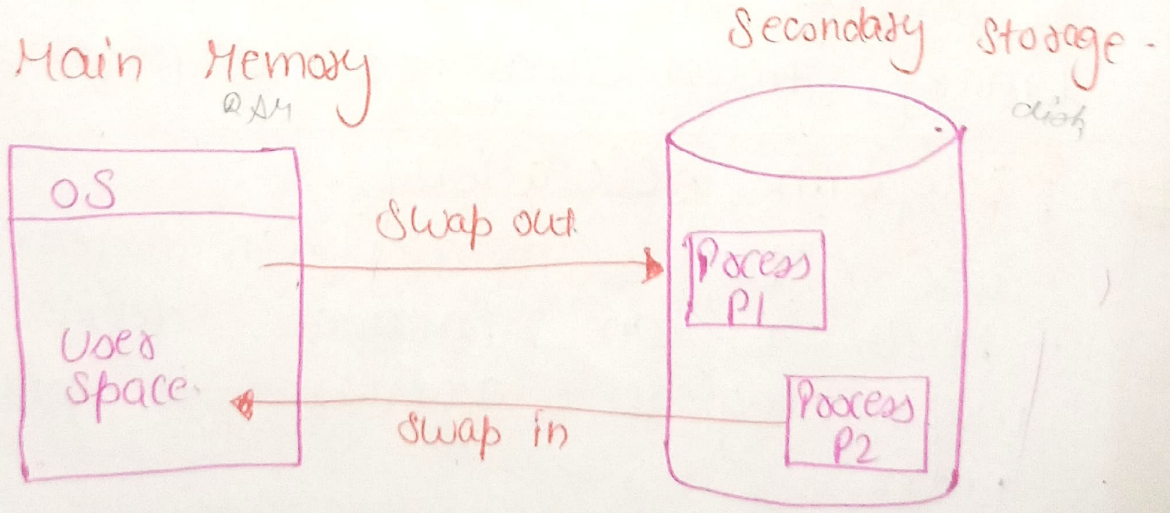
Dynamic → The entire program and all data of process must be in physical memory for process to execute

Loading → Process into the main memory is done by a loader

Swapping

- also known as technique for memory compaction.
- It is the mechanism in which a process can be swapped temporarily out of Main Memory to Secondary Memory later again swaps back to Main Memory

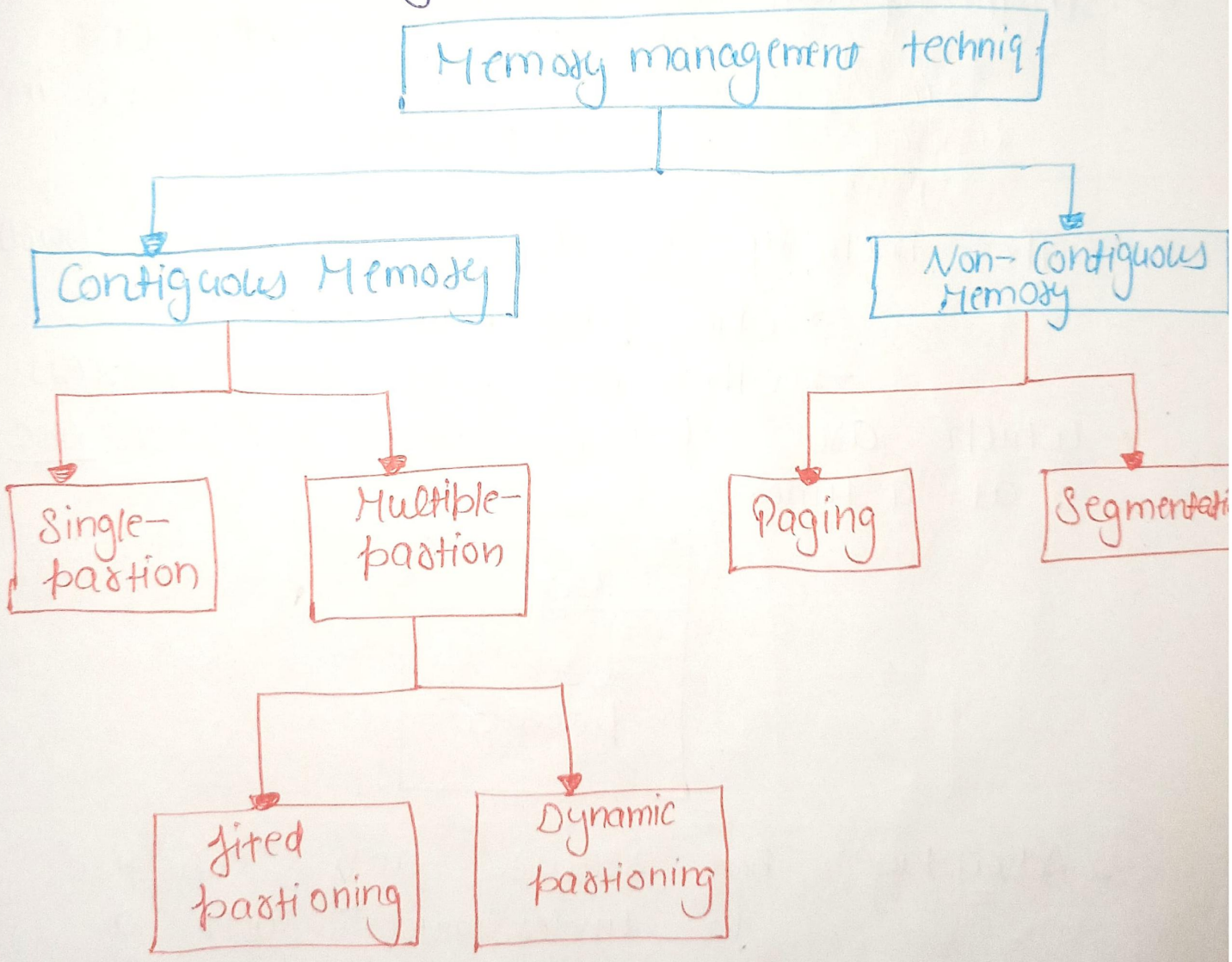
---



- Swap - out → removing a process from RAM and add to hard disk
- Swap - in → removing a program from disk hard disk and put main memory

Memory Allocation

- Memory allocation is an operating system refers to the process of assigning and managing memory space for program and data.
- It's like dividing the available memory into different section and allocating them to different program based on their memory requirements.



Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

⇒ Contiguous Memory Management

- In Contiguous Memory Management scheme, each program occupies a single block of storage location. i.e a set of memory location with consecutive address.

[Faster execution]

→ Single partion Allocation
→ Multiple partion Allocation.

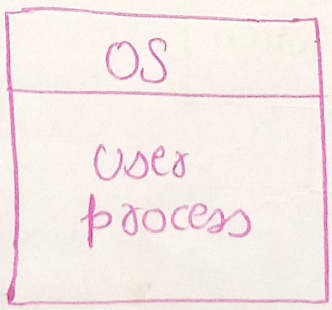
→ Single partion allocation:

Memory divided into different sector or segment

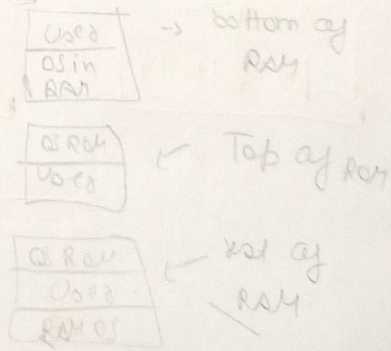
↳ Only one partion follow in mememy.

- The memory is divided into two parts
  - → one part hold the OS
  - → other one hold user process.
- Which are loaded and executed one at a time.

[MS-DOS]



variation



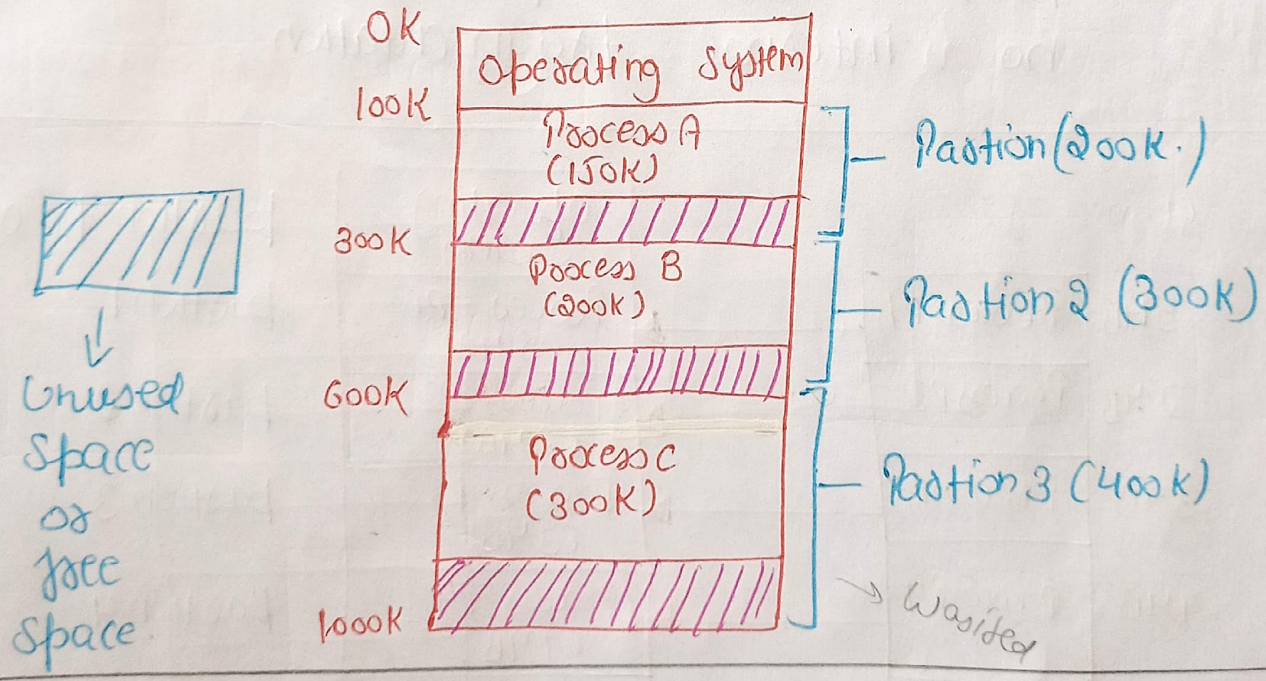
⇒ Advantage: not require any expertise to understand such a system.

⇒ Disadvantage: Poor utilization of Memory / Processors.
- Limited size.

→ Multipartion allocation:

- Multiple-partion allocation is a memory management technique where the system memory is divided into multiple partions.
  - fixed Partion / Static.
  - Dynamic Partion / Variable.

- **fixed Partion:** → Memory is divided into fixed-size partitions, and each partion can hold one process.



- Memory is consisting three area of size 200K, 300K, 400K. each hold a process.
- each partion contain unused space.
- The occurance of wasted space is reffered as internal fragmentation.

Scanned with OKEN Scanner

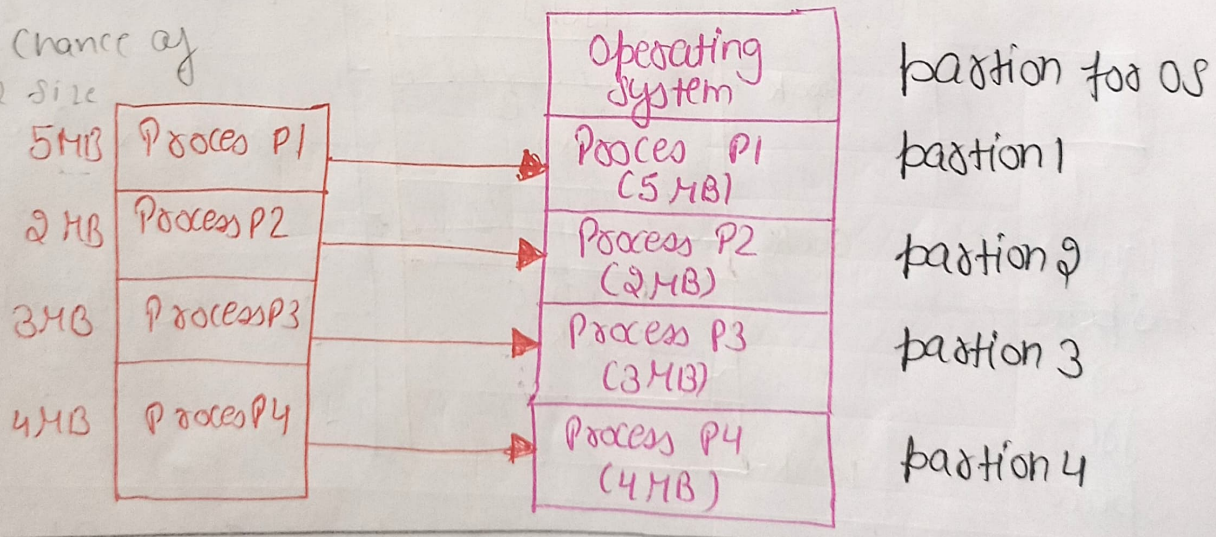
- This scheme was used for batch system and it was good for those system where space requirement of a process could be easily known before hand.

Advantage: easy to implement.

Disadvantage: Internal fragmtation, Limit in size.

- Dynamic Partion: } Partitions are created dynamically.
  - Memory is divided into variable-size partitions to accommodate process of different sizes
  - Partion size = process size, so no internal fragmentation.

No chance of internal size



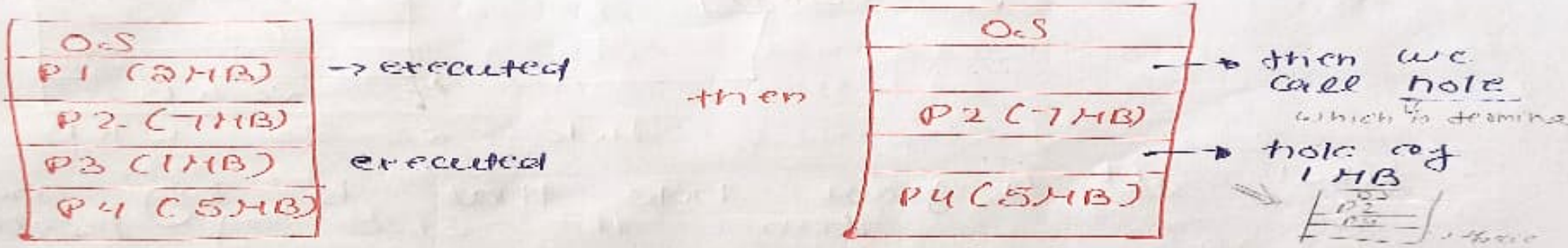
- allocate the exact amount of memory to the process it requires. to avoid internal fragmentation.

eg: Pizza cut according to need only during run time partion.

Scanned with OKEN Scanner

- Advantage : No internal fragmentation.
  No unused space remaining.
  - No limitation on size of process.
- Disadvantage :- External fragmentation.

eg :- - process P1 (2MB) and process P3 (1MB) complete executed or terminate.
- Hence two space are left i.e 2 & 1MB



- Let suppose process P5 of size 3MB comes. The space in memory cannot be allocated as no spanning is allowed in contiguous allocation. then this is the external fragmentation. in we use compaction

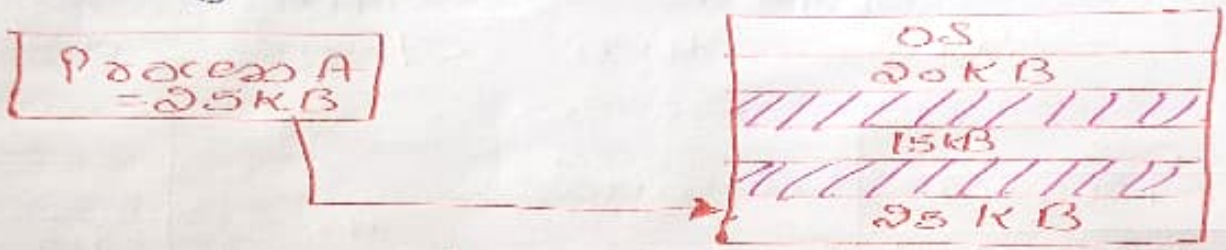
Storage Placement Policies

- When new processes have to be loaded using the variable partition scheme, it is necessary to try to select the 'best' location in which to place them.

- **Best fit policy** :→ allocate the smallest hole that is big enough.
  - It searches the smallest hole whose size greater than or equal to size of process.

Allocate the smallest that is big enough

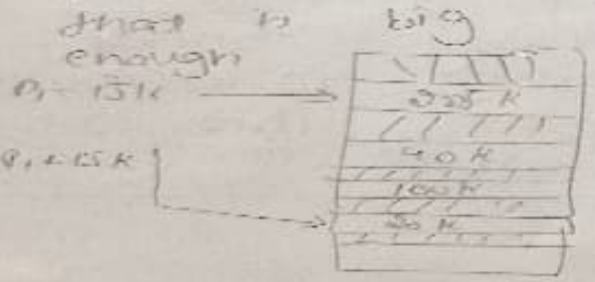
that hole search which least minimum fragmentation

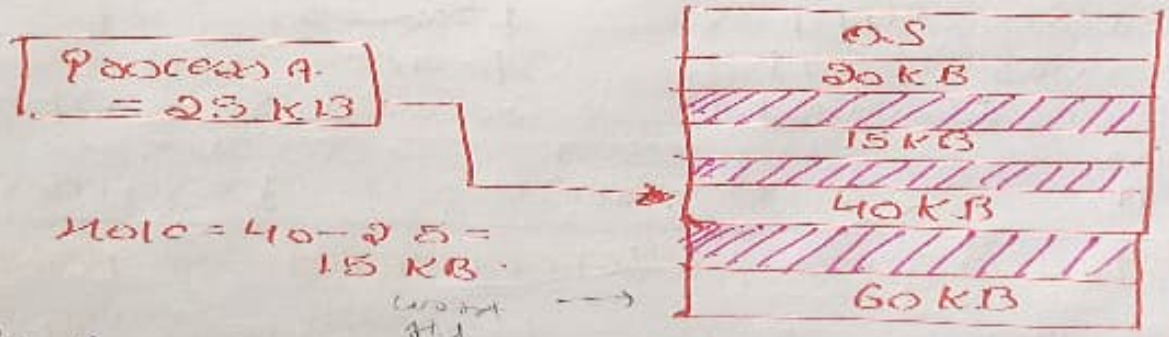


Hole = 25 − 25 = 0 KB.

- **First fit** :→ first hole that is big enough than the size of process

① allocate the first hole that is big enough





Hole = 40 − 25 = 15 KB.

worst fit

Worst

- **Worst fit** :→ largest hole that is big enough is allocated to prog.

allocate the largest hole.

→ where large hole

Compaction

- When partitioning creates multiple holes in memory, it combine them all in one big hole and moving downward these holes.
- Solve the problem of external fragmentation.

↓

Compacted.

↓

- 200K & 250K can be compacted into one hole of size 450K

- Now we can add new process D with size 300K.

Disadvantage

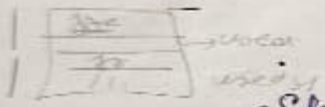
- Complex
- high cost

Scanned with OKEN Scanner

Fragmentation

- Fragmentation is an unwanted problem in the OS in which the processes are loaded and unloaded from memory.
- free memory space is fragmented.
- Cond^n of fragmentation depend on the Memory allocation system.
- Classified into two parts.
  - Internal fragmentation
  - External fragmentation.

- Internal fragmentation: It occurs with fixed partitioning.



: It occurs when there is unused space within a memory block.

Problem: due to fixed size of memory block.

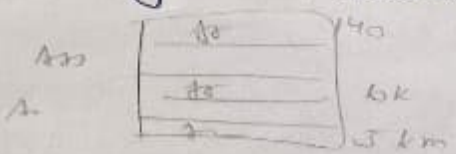
Sol^n: assigning space to dynamic partition.

- External fragmentation: It occur with variable - sized.

[best fit
first fit

- When dynamic memory allocation allocate some memory but leaves a small amount of memory.

Sol^n :→ Compaction.



⇒ Non-Contiguous Memory allocation

- It is a technique where the OS allocates memory to a process in non-contiguous block. → suitable for larger size memory

  → Paging
  → Segmentation

- **Paging** :- Paging is a memory-management technique that provide the non-contiguous address space in main memory.
  - It avoids external fragmentation.
- In this technique, physical memory is broken into fixed-sized block called frames.
  - Logical Memory is divided into block called pages → This division is part of a technique known as paging

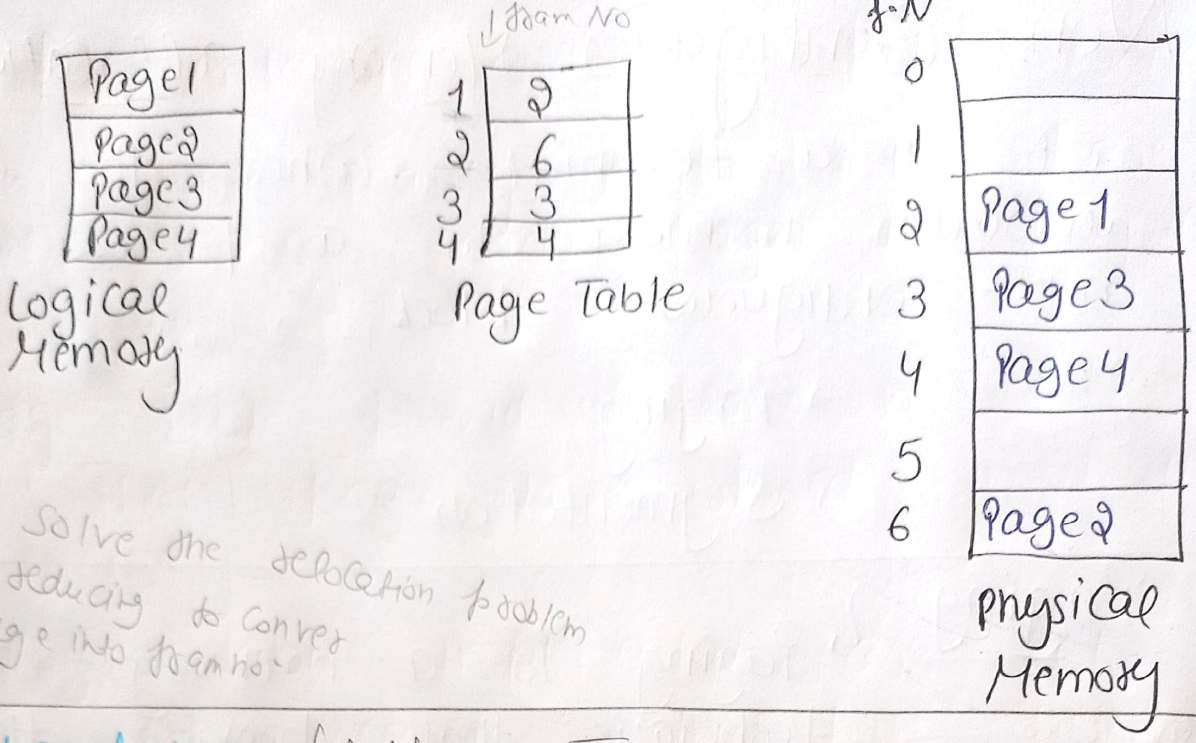
Main Memory

Secondary Memory.

- When a process is to be executed, its pages are loaded into any available memory frames.

Scanned with OKEN Scanner

eg:



To solve the relocation problem by reducing to convert page into frame no.

Hardware / Address Translation

- Memory Management Unit
  [Translating virtual address into physical address.
- Logical address
  → Page number
  → offset
- Physical address
  → frame number
  → offset

=> Logical address: CPU generates this address

- Page number: denoted by p.
  - used as an idex into a page table.
- Page offset: denoted by d [displacment]
  - no. of bits required to represent a particular word in page.

Scanned with OKEN Scanner

Page no	offset
l-p	p

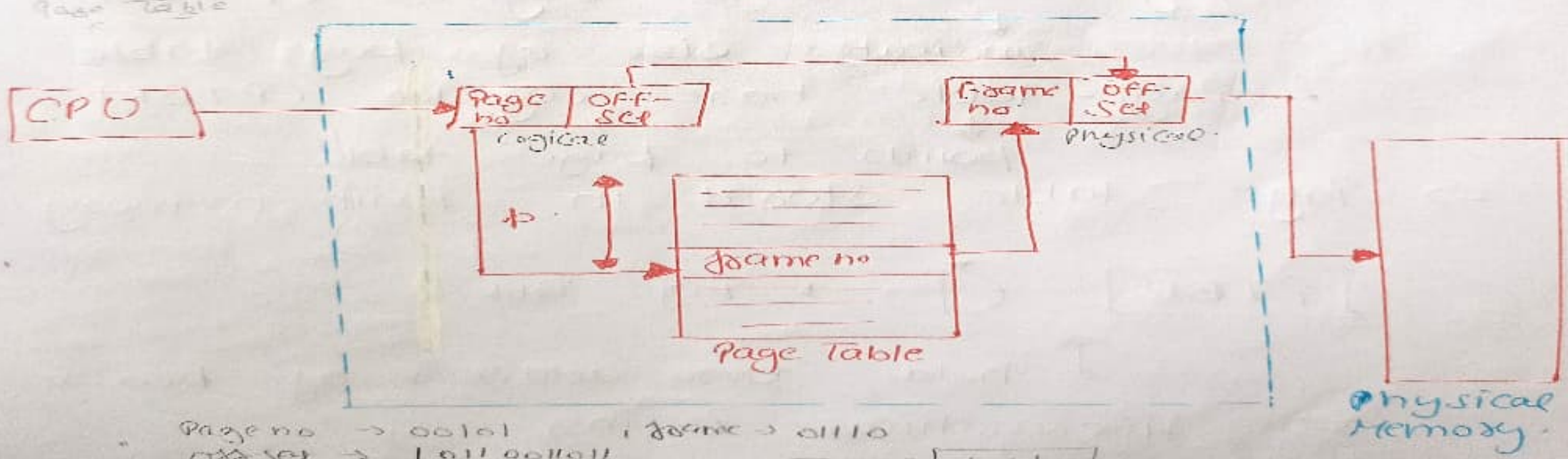
→ logical address

⇒ physical address : It classify two address

- frame number [f] : indicate the specific frame where request page is stored
- offset : denoted by d.
  - specific word to read from page.

frame no	offset
m-p	p

---→ Physical address



Page no → 00101 , frame → 01110

offset → 1011 00110011

Table	
0	0101
1	0010
2	0110
3	0100

Scanned with OKEN Scanner

Implementation of Page table

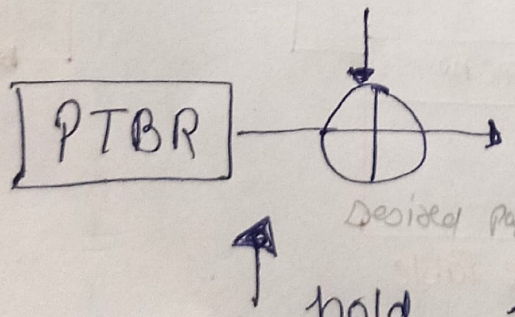
Maintain page informa

- Page table mainly contains the base address of each page in physical memory.
- Page table provide frame no. where that page is stored in main memory.

⇒ Page table implemented as set of register → That have high speed logic to make the paging

- Page-table length register (PTLR)
  ↓
  indicates size of page table
- Page-table base register (PTBR)
  ↓
  points to page table.

⇒ Page table stored in main memory.



hold base address of process.

- Generation of logical address
- Scaling → Page no.
- Gen of physical addr
- Getting actual frame no

⇒ Hierarchical Paging

⇒ Hashed page Table

Structure of Page table

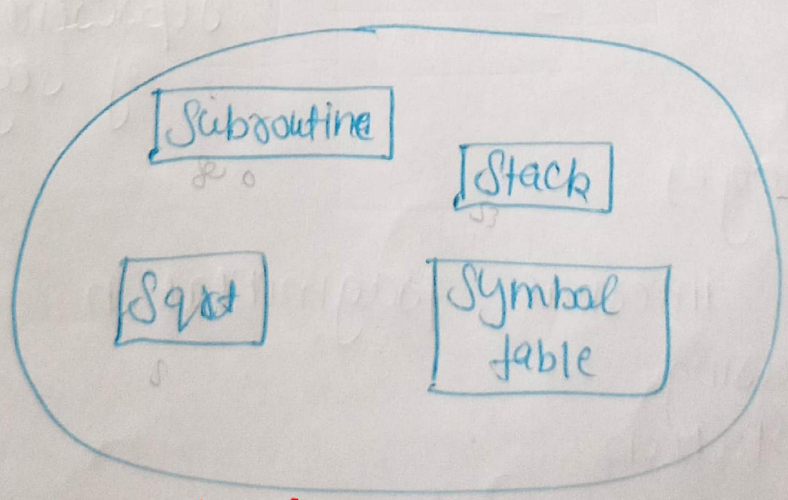
Segmentation

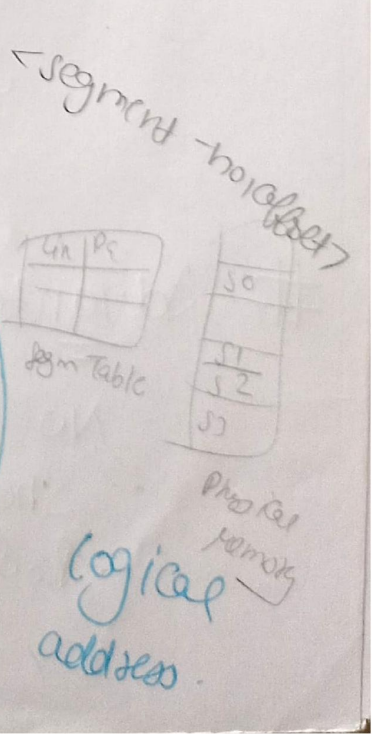
- Segmentation is a memory management technique in which the memory is divided into the variable size part.
- each part is known as a segment which can be allocated to a process. [Process divided into segment]

⇒ Types

1) Virtual Memory: each process is divided into a no. of segment, but the segmentation is not done all at once.
2) Simple Segmentation: each process is dividing into a number of segments, all are loaded into memory at run time.

User View of Program





Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

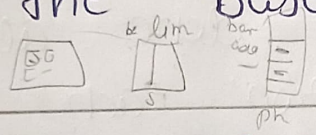
Segment Table

→ maintain the segment information

- each segment are stored in a table.
- It contains mainly two information about segment
  - → Base : base address of segment
  - → Limit : length or size of segment

Segment offset

- It has two column [length, Base]
- Segment table is stored as a separate segment in main memory
- Segment table base register (STBR) Store the base address of segment table.



⇒ The address generated by CPU is divided into:

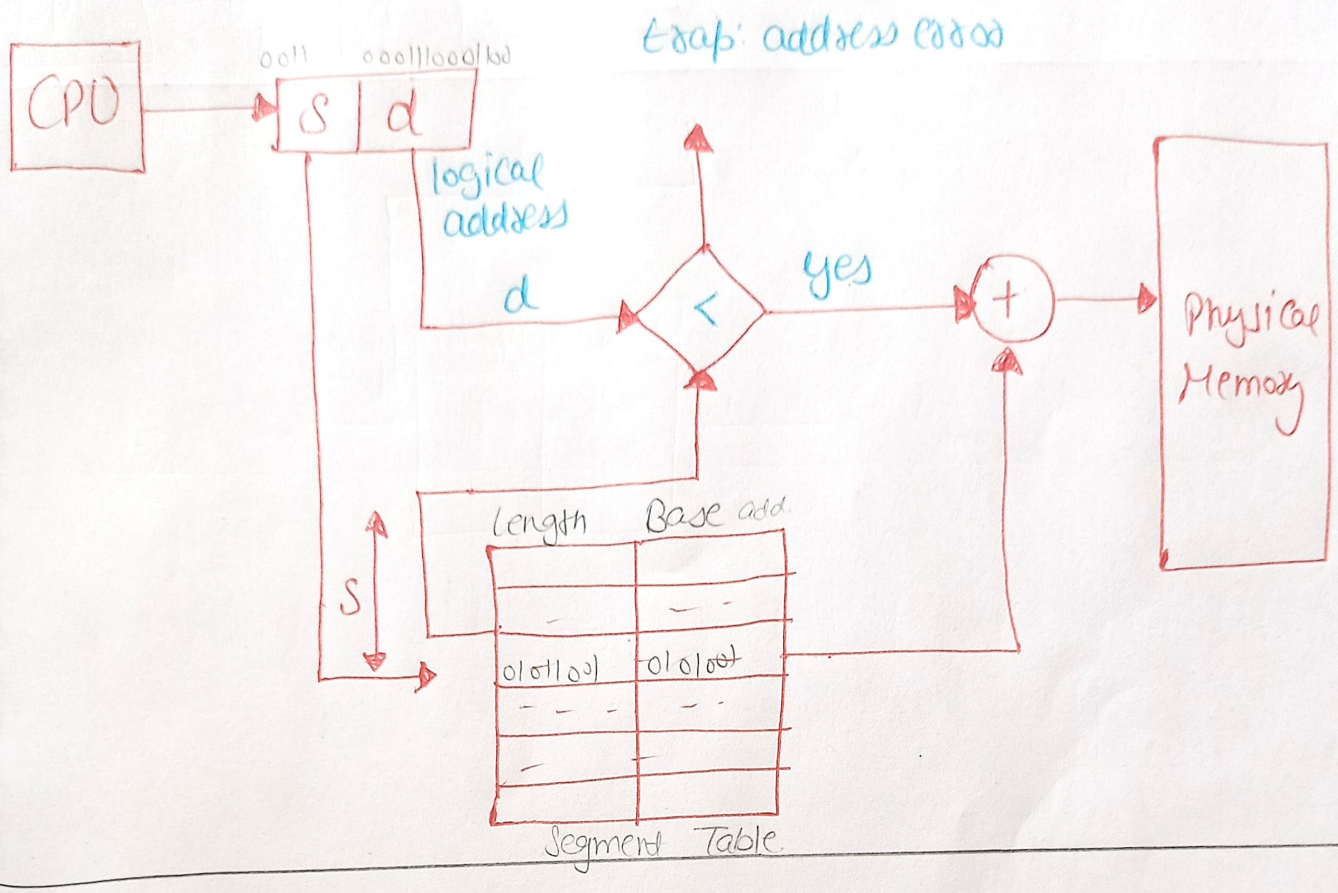
- segment number (S) : No. of bits require to represent the segment
- Segment offset (d) : No. of bits to represent the size of segment

Advantages

- No internal fragmentation
- Sharing
- Protection

Scanned with OKEN Scanner

The Conversion of a logical address into a segment address



Implementation

- As with paging, the segment table can be put either in fast registers.
- Segment table Base register or Segment table length register are implemented.

Protection

Protection is accomplished by associating protection bits to the segment table entries (in case Page table entries)
→ bit determine a page to be read-only.

Sharing

Sharing common program code.

Process Synchronization

- Process synchronization is the way by which processes that share the same memory space are managed in OS.
- It helps to maintain the consistency of data and cooperating process execution by using variable or hardware so that only one process can make changes to the shared memory at a time.

- On the basic of synchronization, processes are categorized in two types.

Independent process

→ The execution of one process does not affect the execution of other processes. [Doesn't share any data].

Cooperative process

→ A process that can affect or be affected by other processes executing in the system. [Share data with other processes]

- There are various solution for such as semaphores, mutex locks, synchronization hardware, etc.
- When process execute, it should be ensured that concurrent access to shared data does not create inconsistencies

How it works

- If a process 1 is trying to read the data present in a memory location while another process 2 is trying to change the data present at the same location, there is a high chance that the data read by process 1 will be incorrect.

Why we need ?

- Resource sharing
- Concurrency control
- Preventing Race condition.

Race Condition

- It occurs when two or more processes are executed at the same time, not scheduled in proper sequence & Not execution critical section correctly, which cause 'Data inconsistency & Data loss'.

→ Sol^n: is by treating the critical section as a section that can be accessed by only a single process at a time.

- This kind of section is called atomic section.

Scanned with OKEN Scanner

⇒ Benefits of Process Synchronization

1) Data integrity : It help to ensure that data is not corrupted when multiple processes are accessing it.

2) Resource efficiency : It helps to ensure that resource are not wasted when multiple processes are trying to use them at the same time.

3) Increased performance : Process synchronization can help to improve the performance by reducing the amount of time that processes spend waiting for resources.

Critical Section

- Critical Section is a code segment that can be accessed by only one process at a time. [No be executed more than one process at a time, it faces difficulties.
- Critical Section contain shared variables which need to be synchronized to maintain consistency of data variables.
- Other process have to wait until the data is free to be used.

→ atomic section required in CS to avoid

→ Critical - Section problem

- Consider a system consisting of n processes ($P_0, P_1, \ldots P_n$).
- Each process has a segment of code, called critical section.
  - → process may be changing common variable, updating a table, writing a file.
  - → when one process is executing in its critical section, no other process is to be allowed to execute
    [No two process are execute in critical section at the same time]. Solve race condn
- The critical - section problem is to design a protocol that the processes can use to cooperate. [It ensure that race condn never occurs]

General Structure of typical process P_i

```
do {
    [Entry Section]
        critical section
    [Exit Section]
        remainder section
} while (TRUE);
```

Copyrighted by Codeleap.io
All rights reserved.

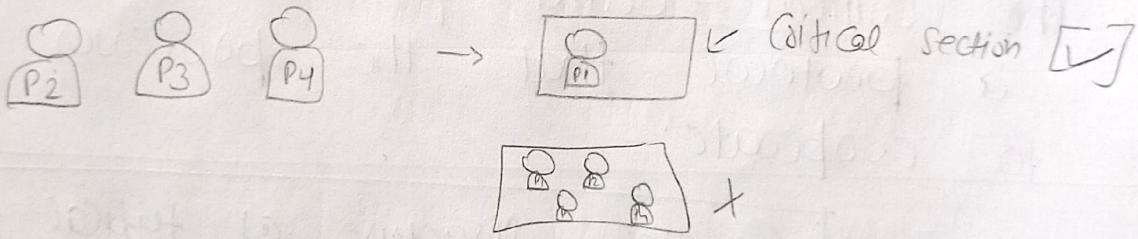
- Each process must ask permission to enter its critical section.

Scanned with OKEN Scanner

- The segment of code implementing this request is the entry section.
- The critical section may be followed by an exit section.
- The remaining code is the remainder section.

⇒ Sol^n to the critical - section problem must satisfy the following three requirements

1) Mutual exclusion :- If one process is executing inside critical section then no other process is allowed to execute in the critical section.



2) Progress :- If one process doesn't need to execute into critical section then it should not stop other process to get into the critical section

3) Bounded waiting :- It exists a bound, or limit, on the number of times that other processes are allowed to enter their critical section after a process has made a request to enter its critical section and before that request is granted.

Scanned with OKEN Scanner

Solution to critical section problem

Peterson
Mutex lock
Semaphores
hardware

- Single processor Sol^n

→ OS handle, only one process at a time.

→ Many system provide hardware support for critical section code. The critical section problem could be solved easily in a single-processor environment.

→ If each process disables all interrupts just after entering in its critical section and re-enable all interrupts just before leaving critical section.

Structure of this Solution

```
do {
    [Disable interrupts]
    critical section
    [Enable interrupts]
    Remainder section
} while (true);
```

Copyrighted by Codeleap.io
All rights reserved.

→ This Sol^n is not feasible in a multi-processor environment.

→ Disabling interrupt can be time consuming.

Scanned with OKEN Scanner

→ Two - process Solution

- also known as multiprogramming, It allow multiple processes to be executed concurrently
- Two process are named as P_0 and P_1, where one process is P_i and other is P_j $\boxed{\text{ie } j = 1 - i}$.

→ Alg. 1

- integer variable turn is used to keep track of whose turn it to enter the CS.

```
do {
    [ while (turn != i) do nothing; ]
      critical section
    [ turn = j; ]
      remainder section
} while (true);
```

→ Alg. 2

- the variable turn is replaced by an array.

boolean flag[2];

```
do {
    [ flag[i] = true;
      while (flag[j] do nothing); ]
      critical section
    [ flag[i] = false; ]
      remainder section
} while (true);
```

Scanned with OKEN Scanner

→ Peterson's Solution

- It is a Classical software based sol^n to the critical section problem
- In peterson's solution require two data items to be shared b/w the two process

→ int turn : indicates whose turn it is to enter into critical section.

→ boolean flag[2] : indicate when a process want to enter into critical section.

→ flag[i] = true implies that process P_i is ready!

→ flag[i] = false, no one enter the critical section

The structure of P_i in Peterson's Sol^n

```
do {
    Flag[i] = TRUE;
    turn = j;
    while ( flag[j] && turn == j);
        critical section
    Flag[i] = FALSE;
        remainder section
} while (TRUE);
```

Scanned with OKEN Scanner

- Peterson's Sol^n preserve all three Cond^n
  - → Mutual exclusion is preserved
  - → The progress requirement is satisfied
  - → The bounded - waiting requirement is met.

→ Disadvantages

1) It involves busy waiting.
2) The Sol^n is only limited for 2 processes.

→ This solution is based on the idea that when a process is executing in a critical section, then the other process execute the rest of code and vice-versa is also possible i.e only one process executes the critical section at one time.

Locks

- We can prevent race cond^n by protecting critical section with locks

```
do {
    [acquire lock]
        critical section
    [release lock]
        remainder section
} while (TRUE);
```

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

→ Two General approaches are used to handle critical section

Preemptive

- allow preemption of process when running in kernel mode

Non-preemptive kernels

- run until exits kernel mode, blocks or voluntarily yields control of CPU.
- essentially free from race condn on kernel mode.

Semaphores

← Mathematic sol^n denoted by S

- Semaphores concept was introduced by Dijkstra in 1965 for managing concurrent process by using the value of simple integer variable.
- Semaphore is simply a variable which is non-negative and shared b/w threads. This variable is used to solve the critical section problem and to achieve process synchronization in the multiprocessing environment.
- It is accessed only through two atomic operation

  :- wait ( )

  :- signal ( )

Advantage

- solve various synchronization process
- machine independent implementation
- allow only one process in CS
- follow mutual exclusion

Dis

- complicated & wait & signal implement
- read priority inversion.

:- Wait():

also called P, means "to test".

· It decrement the value of its argument S

[S>0]
↓
passes block

[S-1]

```
wait (S)
{
  while (S<=0)
    ; // no operation
  S--;
}
```

:→ Signal():

Also called V, mean "to increment"

· It ~~decrem~~ increment the value of its argument S.

[S+1]

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

Copyrighted by Codeleap.io
All rights reserved.

Hence, it is guaranteed that once a semaphore operation has started, no other process can access the semaphore until the operation has completed.

→ absolutely solving synchronization problem

→ avoiding racing cond^n.

Scanned with OKEN Scanner

→ Type of Semaphore

Mutual exclusion

Binary [0&1]

- also known as mutex lock.
- The value of B.S can range only b/w 0 & 1 ← value is initialized to 1.
- It is used to implement the soln of critical section problem with multiple process.

No mutual exclusion

Counting (1,2,3...)

- Its value can range over a unrestricted domain.
- It is used to control access to a resource that has multiple instances. [More than one slot]

→ Semaphore Usage

a) To Solve the Critical Section problem.

→ Semaphore can be used to solve the critical section problem. The n processes share a semaphore mutex (mutual exclusion), initialized to 1.

→ each process P_i is organized

```
do {
    wait (mutex);
        critical section
    signal (mutex);
        remainder section
} while (true);
```

- The Mutex - exclusion implementation with semaphore (mutex) ensure that only one process in its critical section.
- When mutex = 1, first process execute, wait able to enter the critical section and set mutex -1. When first process exits from critical section, it will execute signal and increment the value of Mutex
  Mutex = Mutex + 1 = 1.

---

b) To Synchronize certain operation b/w process

→ Semaphore can also be used to synchronize certain operation b/w processes.

→ eg: Consider two concurrently running processes

1) P_1 with a statement S1
2) P_2 with a statement S2

→ Suppose we require that S_2 be executed only after S1 has completed.

→ first, create a semaphore named Synch that is shared by two process and initialize it to zero.

* P_1, insert the code:

```
S1;
signal (synch);
```

* P_2, insert the code:

```
wait (synch);
S2;
```

Scanned with OKEN Scanner

→ Implementation

- The main disadvantage of the semaphore definition that was discussed is that it requires busy waiting.
- While a process is in its critical section, any other process that tries to enter its critical section must loop continuously in the entry code.

→ Busy waiting wastes CPU cycle that some other process might be able to use productively. This type of semaphore is also called a spinlock because the process "spins" while waiting for the lock.

Sol: To overcome the need for busy waiting, we can modify the definition of the wait() and signal() semaphore operations.

→ Two operation apply

1) block() : place the process invoking the operation on the appropriate waiting queue.

Suspend the process that involve it

2) wakeup(): remove one of process from the waiting queue and place it in the ready queue.

resume the execution of a blocked process P.

Scanned with OKEN Scanner

→ Implement Semaphore under new definition.

Semaphore Structure

```
typedef struct
{
    int value;
    struct process *list;
} Semaphore;
```

· each Semaphore has an integer value and a list of processes.

Wait operation

```
wait (Semaphore *S)
{
    S -> value --;
    if (S -> value < 0)
    {
        add this process to S -> list;
        block ();
    }
}
```

Signal operation

```
signal (Semaphore *S) {
    S -> value ++;
    if (S -> value <= 0)
    {
        remove a process P from S -> list;
        wakeup (P);
    }
}
```

Scanned with OKEN Scanner

Deadlocks and Starvation

- The implementation of a semaphore with a waiting queue may result in a situation where two or more processes are waiting indefinitely for an event that can be caused only by one of the waiting processes.
- When such a state is reached, these processes are said to be deadlocked.

Ex: Suppose there are two processes P_0 and P_1 each processes accessing two semaphores S_1 and S_2, set the value 1.

P_0	P_1
wait (S_1);	wait (S_2);
wait (S_2);	wait (S_1);
⋮	⋮
signal (S_1);	signal (S_2);
signal (S_2);	signal (S_1);

- When P_0 execute wait (S_1), then P_1 executes wait (S_2)
- also when P_0 executes wait (S_2), it must wait until P_1 executes signal (S_2)
- lly, when P_1 executes wait (S_1), it must wait until P_0 executes signal (S_1).

Scanned with OKEN Scanner

- Since these signal operation cannot be executed, P_0 and P_1 are deadlocked.

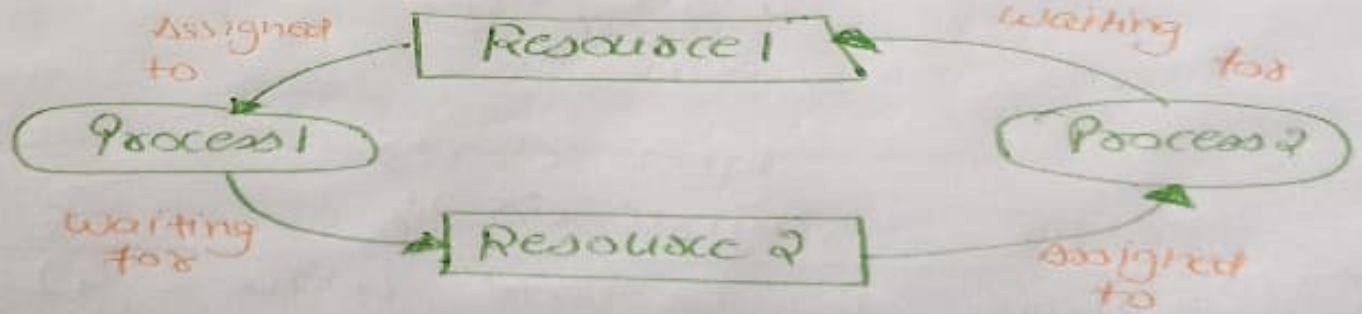
→ Another problem Starvation / Indefinite blocking

- It is a situation in which process wait indefinitely within the semaphore
- Indefinite blocking may occur if we add and remove processes from the list associated with a semaphore in LIFO (Last-in-first-out) order.

Deadlock

It is a situation where a set of process are blocked because each process is holding a resource and waiting for another resource acquired by other process.

prevent by avoiding the necessary condn for deadlock →



Starvation

that occur when high priority processes keep executing and low priority processes get blocked for indefinite time.

← prevent

Classical problem of Synchronization

- We present a number of synchronization problem as example of a large class of concurrency - control problems.
- These problem are used for testing nearly every newly proposed synchronization scheme. In our soln to the problem, we use semaphores for synchronization

1) Bounded - Buffer

- Also known as Producer Consumer problem
  → A producer process produces information that is consumed by a consumer process

→ Bounder - Buffer
- Assumes a fixed buffer size. In this case, the consumer must wait if the buffer is empty and the producer must wait if the buffer is full.

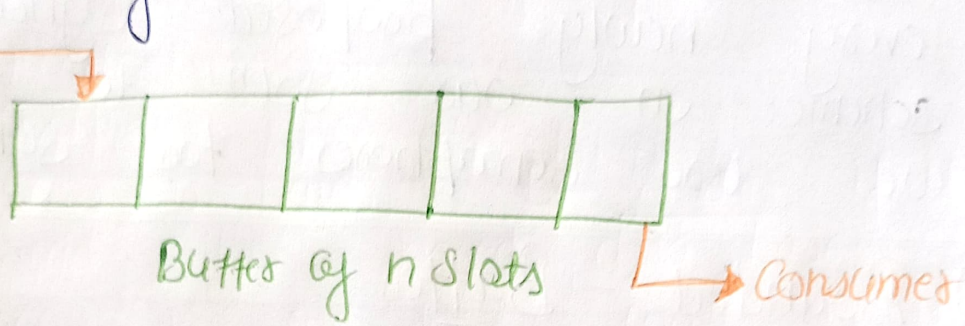
→ Unbounded buffer
- Places no practical limit on the size of the buffer. The consumer may have to wait for new items, but the producer can always produce new items.

→ Problem Statement :- In this problem, two processes → one is producer
→ other is consumer,
run concurrently

Scanned with OKEN Scanner

and those two processes won't produce the expected output if they executed concurrently.

→ There is a buffer of n fixed-size slot, and each slot is capable of storing one unit of data.



→ The producer tries to insert data into an empty slot of the buffer.

→ The consumer tries to remove data from a filled slot in the buffer.

→ We need to ensure that when a producer insert data in buffer, then at the same time consumer should not consume any data item i.e both work in an independent manner.

→ Solution :- 1, One solution of this problem is to use semaphore.

Mutex

- m, a binary semaphore which is used to acquire and release the lock.
- full, a counting semaphore, whose initial value is 0
- empty, a counting semaphore, whose initial value is n. ← All slot are

Producer

Copyrighted by Codeleap.io
All rights reserved.

- decrement the empty semaphore before adding an item to buffer

```
do {
    ...
    // produce an item in nextp
    ...
    wait (empty);
    wait (mutex);
    ...
    // add item to nextp to buffer
    ...
    signal (mutex);
    signal (full);
} while (TRUE);
```

- increment the full semaphore after adding an element
  → task of producer completed and consumer can access the buffer
  [Mutex value also ↑]

Consumer

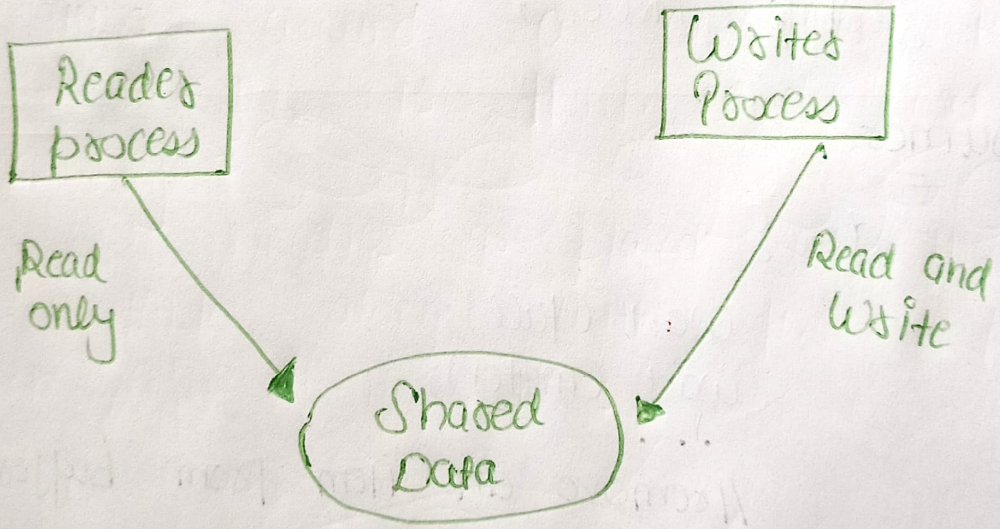
```
do {
    wait (full);
    wait (mutex);
    ...
    // remove an item from buffer to nextc
    ...
    signal (mutex);
    signal (empty);
    ...
    // Consume the item in nextc
    ...
} while (TRUE);
```

→ producer is producing full buffer for the consumer or the consumer is producing empty buffer for the producer.

Scanned with OKEN Scanner

2) Reader - Writers Problem

- Reader - Writer problem is another Classic synchronization problem.
- a database is to be shared among several concurrent processes.
- There are two types of processes
  - → Reader
  - → Writer
- → A process is said to be reader if it can read the shared data base.
- → A process is said to be writer if it can read or write a shared data base.



→ Problem :-

- Allow multiple readers to read at the same time.
- Only one single writer can access the shared data at the same time.

Scanned with OKEN Scanner

→ Solution: We use Semaphore to solve the problem

1) Mutex initialize to 1 (Binary Semaphore), used to ensure mutual exclusion when the variable readcount is update.

2) Wrt initialize to 1 (Binary Semaphore) common to both reader and writer, used to block and realse the writers.

3) Readcount initialize to 0 (Counting Semaphore) It keep track of how many processes are currently reading the object.

Structure for Writer process

```
do {
    wait (wrt);

    // writing is performed

    signal (wrt);
} while (TRUE);
```

Structure for reader process

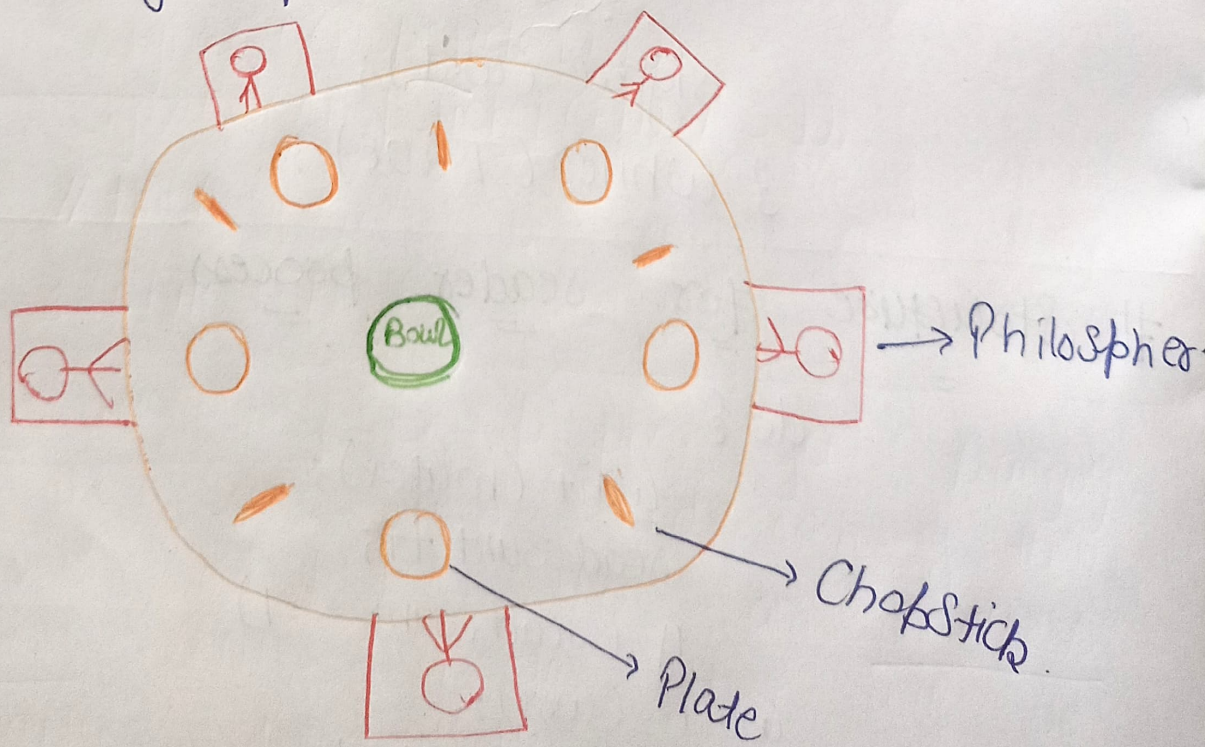
```
do {
    wait (mutex);
    read count ++;
    if (read count == 1)
        wait (wrt);
    signal (mutex).
```

Scanned with OKEN Scanner

```
//reading is performed
wait (mutex);
read count --;
if (read count == 0)
signal (wrt);
signal (mutex);
} while (TRUE);
```

3) Dining Philosopher Problem

- Dining philosopher problem is a Classical Synchronization problem in OS.
- In this problem, Says that five philosphers are sitting around a circular table and their job is think and eat alternatively.
- A bowl of noodles is placed at the center of the table along with five chopstick for each of philosophers.



Scanned with OKEN Scanner

→ Philosopher either

Thinks

- When a philosopher thinks, he does not interact with his colleagues.

Eats

- When a philosopher get hungry he tries to pick up the two forks that are closest to him (left & right)
- A philosopher can only eat if both immediate left & right chopsticks of the philosopher is available.
- In case of if both immediate left & right chopsticks of the philosopher are not available then the philosopher put down their (either left or right) chopstick and starts thinking again.

Copyrighted by Codeleap.io
All rights reserved.

- The dining philosopher demonstrates a large class of concurrency control program hence it's as a classic synchronization problem.

Problem → How we can prevent deadlock & starvation

Sol^n → represent each chopstick by a Semaphore

Scanned with OKEN Scanner

- a philosopher tries to grab a chopstick by executing a wait() operation on that semaphore
- for releasing a chopstick signal() operation can be executed

→ Shared data are

```
Semaphore chopstick[5];
```

∴ where all the elements of chopstick are initialized to 1.

Structure of philosopher i

```
do {
    wait (chopstick[i]);
    wait (chopstick[(i+1) % 5]);
    ...
    // eat
    signal (chopstick[i]);
    signal (chopstick[(i+1) % 5]);
    // think.
} while (TRUE);
```

- But if all the philoposophers get hungry at the same time, and each of them pickup one chopstick, then a deadlock situation occurs because they will be waiting for another.

Scanned with OKEN Scanner

→ possible sol^n to avoid deadlock

1) Allow only four philosopher at the table.
2) Allow philosopher to pick up chopsticks only when both are available, in a critical section.
3) Use an asymmetric sol^n: an old philosopher picks up first his left chopstick and then his right chopstick.
   - an ~~at~~ even philosopher picks up his right chopstick and then his left chopstick.

The Critical Regions ← Synchronization secure to maintain data integrity

- The time dependent error can be easily generated when semaphore are used to solve the critical section problem. → To overcom this difficulty, critical region was introduced.

→ A critical region is a section of code that is always executed under mutual exclusion and maintain the data consistency and race cond^n need to be prevented.

→ Critical regions shift the responsibility for enforcing mutual exclusion from the programmer to the compiler.

→ Variable that must be accessed under mutual exclusion.

Scanned with OKEN Scanner

- A new Statement that identifies a critical region in which the variable are accessed.

VAR v : SHARED T; ← Critical region Construct.

- The variable v can be accessed only inside a region statement

REGION v DO S;

- While statement S is being executed, no other process can access the variable v.

⇒ Characteristics

1) Mutual exclusion
↳ only one process or thread should be allowed to execute within the critical region at any given time.

2) Atomicity
↳ operation within critical region should be atomic i.e they execute indivisibly.

3) Synchronization
↳ Synchronization mechanism, such as semaphore, mutex, are used to enforce mutual exclusion.

Scanned with OKEN Scanner

Conditional Critical regions

(Both involve accessing shared resource.)

- A Conditional critical region is a variation of the critical region concept. It refers to a section of code where access to shared resource depend on certain condⁿ being met.

REGION v WHEN B DO S;

→ B is boolean expression.

→ Region referred as shared variable.

→ When a process enter the critical section region, the boolean expression B is evaluated

1) If the expression is true, statement S executed, if it false, the process relinquishes the mutual exclusion.

2) Delayed until B becomes true.

- Synchronization is necessary only under certain condⁿ.

→ Limitation

- more difficult to implement efficiently than semaphores
- Still distributed among the program code.
- There is no control over the manipulation of protected variable.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

Monitors

- Various type of errors can be generated easily when programmers use semaphores incorrectly to solve the critical - section problem
  ↳ To deal with such errors, researchers have developed a high - level language constructs i.e. Monitor

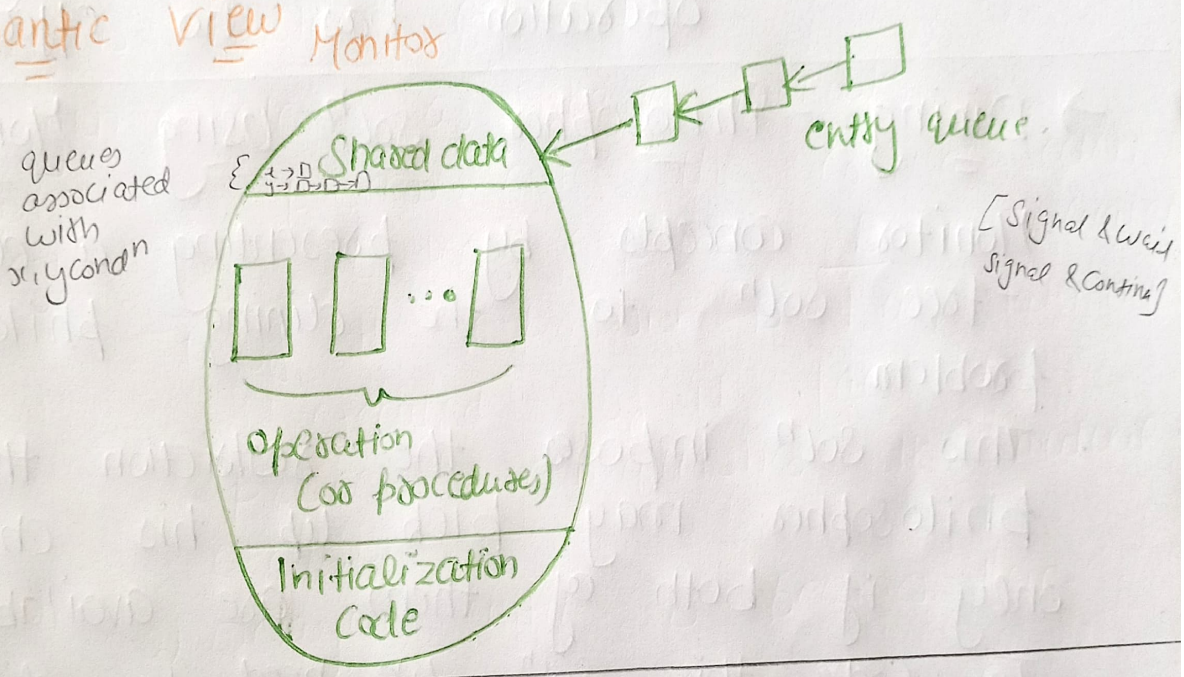
→ A high level abstraction that provide a convenient and effective mechanism for process synchronization.

Syntax

```
monitor monitor - name
{
  // Shared variable declaration
   procedure P1 (...) {
     ...
   }
   procedure P2 (...) {
   ...
   }
   :
   procedure Pn (...) {
   ...
   }
   Initialization code (...) {
   }
}
```

- A procedure defined within a monitor can access only those variable declared locally within the monitor and its formal parameters.
- Similarly, the local variable of a monitor can be acessed by only the local procedures
- <u>The monitor construct</u> → only one process at a time can be active within the monitor.

→ Semantic view Monitor



- each monitor has an entry queue and processes that went to call a monitor procedure join this queue.
- <u>Condition Construct</u> ← define two additional synchro.

$\boxed{\text{Condition } x, y;}$

→ The only operation that can be involved on a condn variable are
- → wait()
- → signal().

Scanned with OKEN Scanner

- x.wait();
  → means that the process invoking this operation is suspended until another process invoke x.signal()

- x.signal();
  → resumes exactly one suspended process. If no process is suspended, then the signal operation has no effect.

→ Dining - Philosopher sol^n using Monitors

- Monitor concepts by presenting a deadlock-free sol^n to the dining philosopher problem.
- This sol^n imposes the restriction that a philosopher may pick up his chopsticks only if both of them are available.
- To code this sol^n, we need to distinguish among three states in which we may find a philosopher

```
enum { thinking, hungry, eating } state[5];
```

- state[i] = eating
  (state[(i+4)%5] != eating)
  (state[(i+1)%5] != eating)

  ] Philosopher i set the variable

File Management

- It involves organizing, storing, retrieving, and manipulating data on computer storage devices

⇒ File

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disk, magnetic tapes and optical disks.
- The information in a file is defined by its creator

⇒ These are two important ~~feature~~ characteristic of file

File Activity

- refers to the operation & interaction that occur with a file over time

File volatility

- refer to the frequency of change to file content or attributes

File Naming

- It refers the rules used to give unique and identifiable named to files.

→ file names

→ file Extension

Scanned with OKEN Scanner

- file names :→ include letters, number and a variety of special characters.
  "/", "*", ":"
- The two parts of a filename are separated by a period (.)
  ex: student . cpp
- Blank space characters not allowed.
- file Extension:
  - identify the type or format of a file.
  eg: "txt" → indicate a text file.
  "jpg" → indicate a JPEG image
  [Reserved words not allowed as a file name]

file attributes

- file attributes are a type of meta-data that describe and may modify how file and directories in a filesystem behave.

- Name
- Identifier
- Location
- Type
- Size
- Protection
- Time, data and User identification

Scanned with OKEN Scanner

- **Name:** It is the only information stored in a human-readable form.
  - every file carries a name by which the file is recognized in the file system.
  - A name is usually a string of characters ["sample.c"]
- **Identifier:** every file is identified by a unique tag number within a file system.

  eg: extension. txt [Text file]
- **Location:** Specifies the location of file on the device.
- **Type:** It specifies the type of file such as archive file (.zip), source code file (c., .java).
- **Size:** Specifies the current size of file in bytes (kb, Mb, Gb etc).
- **Protection:** Specifies information about access control (Permission about who can read, write file).
  - It provide security to private information.

Scanned with OKEN Scanner

- Time, date and User identification
  - This information tell us about the date and time on which the file was created, last modified by which user.

other attribute is flag
- Read-only
- Hidden flag

---

file operations

- The various operation can be performed on a file like.
  - Create a file
  - open a file
  - Write
  - Reading of file
  - Delete a file
  - Truncate a file
  - Rename operation.

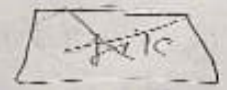
- Crate operation: To create a file in the file system, create operation is used.
  [folder and save, location]
- open a file: Once the file is created, it must be open.
  - It provide file name to open the particular file.

Scanned with OKEN Scanner

- Write operation : This operation is used to write the information into a file
  - System call write is issued that specifies the name of file and length of data has to be written to the file.

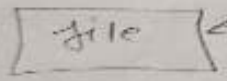
change in data

- Reading a file : After creating a file, it just to read the data.
- Delete a file : When some record is not required in the file, then we delete that record.
  - Complete delete the file.



← Delete all data
← also attribute delete

- Truncate a file : deleting the file except deleting attributes.



← Some information are write after applying Truncate, the instance information of file delete except attribute, like name, location

- Rename : This operation is used to rename the existing file.

Scanned with OKEN Scanner

Copyrighted by Codeleap.io
All rights reserved.

Type of file

- file are organized and classified based on their type and format
- Text file
  - Contain data / information in textual form
  - Extension :- .txt , .docx
- Source file
  - Contain the source code of a language
  - Extention :- .c , .java
- object file
  - Sequence of byte organized into blocks understanble by system linker
  - Extension :- .obj , .o
- Executable file
  - Contain instruction that the computer CPU can execute
  - Extension :- .exe , .com
- Word processor file
  - Contains text and various word-processor format
  - Extension :- .wp , .tex
- Library file : Contain the built-in funcⁿ
  - .lib

Scanned with OKEN Scanner

File Structure

- file type are also used to indicate the internal structure of a file.
- file can be structured in any of several way
- Three common possibilities are

1) Byte Sequence

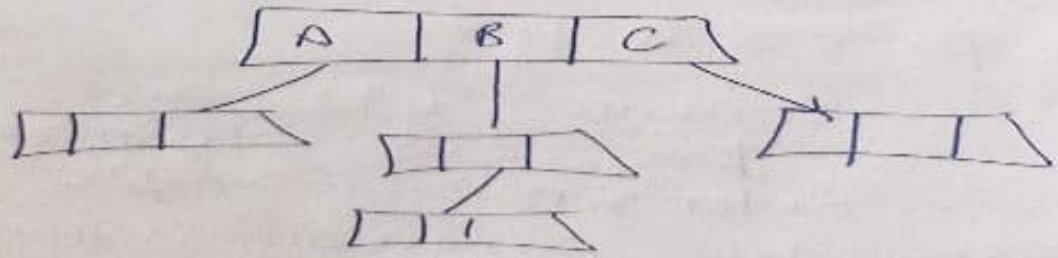
- UNIX and Window use this approach

- data is organized as an unstructured seque of bytes [i.e. linear sequence]

2) Record sequence

- data is organized into fixed - size record or variable - length record.
- record collection of related data field.

3) Tree.



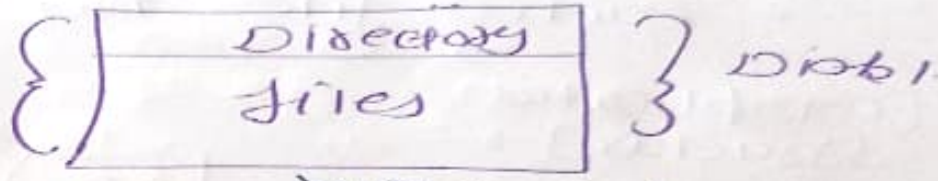
- Not necessarily all the same length.

Scanned with OKEN Scanner

Directory Structure

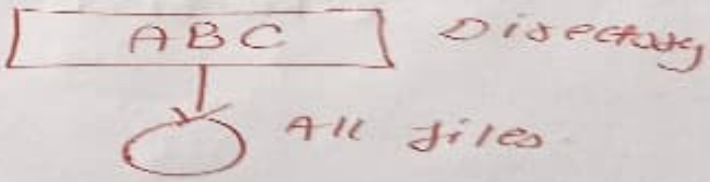
- A directory is a container that is used to contain folder & files
- It organizes file and folder in a hierarchical manner

- It must be designed -
  - → Searching a file
  - → Creating a file
  - → deleting a file.

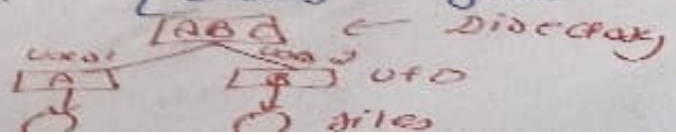


⇒ Type of Directory Structure
→ also known as " logical structure of a directory.

1) Single-level Directory
- Simplest directory structure
- All file are contained in the same directory.
  - → Name - Collision problem
  - → Confusion of file name



2) Two-level Directory
- Solⁿ of this problem is to create a separate directory for each user.
- each user has their own [user file directory]

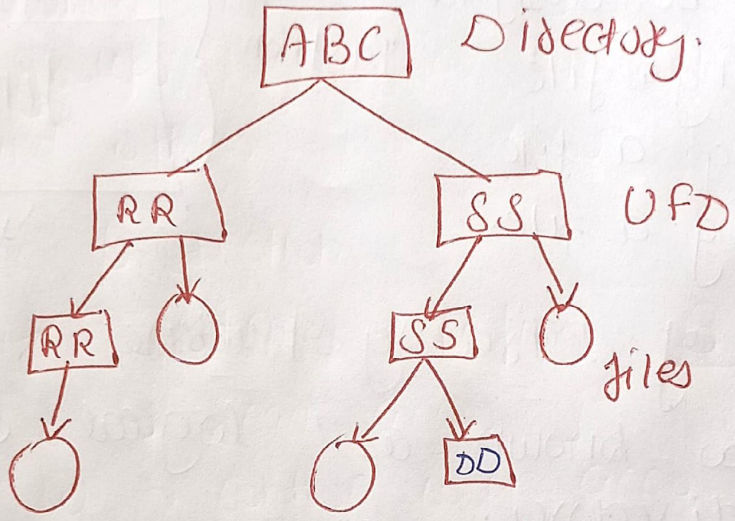


Scanned with OKEN Scanner

3) Tree / hierarchical - Structured Directory

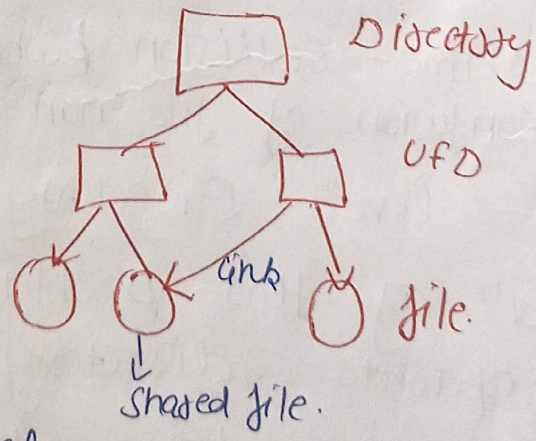
- Tree has a root directory. This allow user to create their own sub directories and organize their files.
- each file has unique path name.

[Complicated Structure]



4) Acyclic - Graph Directory Structure

- Tree Structure not allow the sharing of files, It allow directories to have shared subdirectories and file.



Directory operation

- Create
- Delete
- Rename.
- link
- Unlink.

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

Layered File System

application program
↓
logical file system
↓
file organization Module
↓
basic file system
↓
I/O control
↓
devices

File Protection

- When a file is shared among several users, it becomes necessary to develop a machanism that provides restricted access to file i.e provide file protection.
- process of securing file from unauthorized access.
- Several operation on file can be control.
  - → Read
  - → Write
  - → Execute
  - → Delete.

Scanned with OKEN Scanner

Technique

1) File permissions

- based on a set of attributes assigned to each file, indicating the type of access allowed for the
  - → owner [who create]
  - → groups [set of user who share the file]
  - → and others. [other users]

  ↓ define protection.

eg: UNIX system define 3 bit [rwx]

[r → read access]
[w → control write access]
[x → control execution]

2) Access Control list

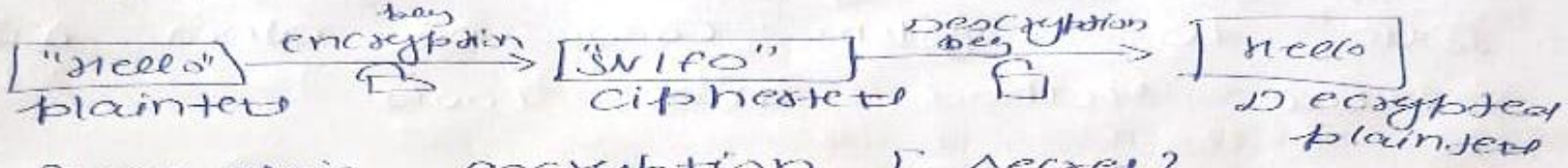
- set of permission attached to an object (such as a file or directory) that specify which user or system process are granted access.
- what operation are allowed.

3) User Authentication

- Ensuring that user are properly authenticated before accessing file.
- involve validity the identity of user password, biometrics. → fingerprint

4) Encryption

- process of converting data into a format that is unreadable.



→ Symmetric encryption [secret]
→ A Symmetric encryption [public key]

5) File Auditing

- involve tracking and recording file access event.
- Audit logs may include information about who access, what operation performed a file.

6) Security Policies

→ inclue rule & guidelines

Computer Viruses ← direct impact on data.

- A virus is a program that is designed to delete the information stored in computer.
- A computer virus enter into computer, without the knowledge of user.
  → Trojan Horse
  → Worms.

Scanned with OKEN Scanner

File recovery

Copyrighted by Codeleap.io
All rights reserved.

- involve the process of restoring lost or deleted files.
- Due to System crase or other reason.

→ Several Method and Tools

Window :→ Recycle bin
Back-up
recovery Softer [Third-party]

Macos :→ Trash
Back-up [Time Machine]
Terminal command Recovery Softer [Data Resue]

Linux :→
- Trash
- Back-up
- PhotoRec → command-line tool
work on Multible platform

Access Methods

- file store information, when it is used, this information must be accessed and read into computer memory.
- There are several ways that the information in the file can be accessed

→ Sequential access

→ Random (or direct access)

→ Indexed Sequential

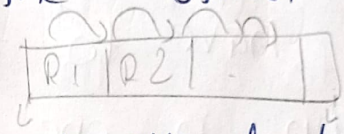
Scanned with OKEN Scanner

- Sequential access

→ Sequential file are those, which are read sequentially in an order.

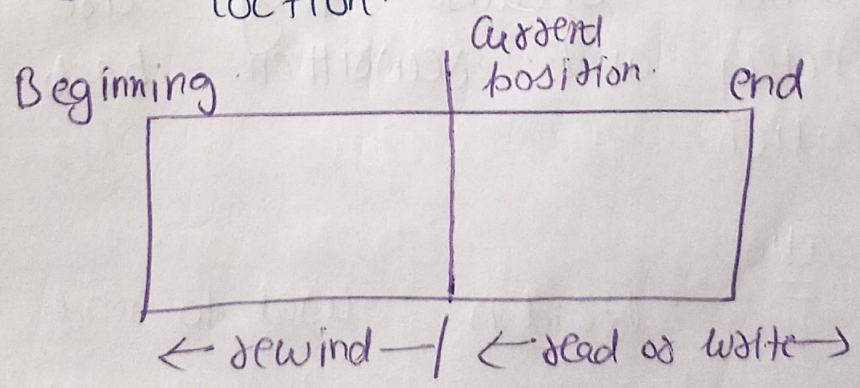
→ Starting at the beginning to the end of file.

→ In sequential file, each record is stored one after the other, without any index or key.



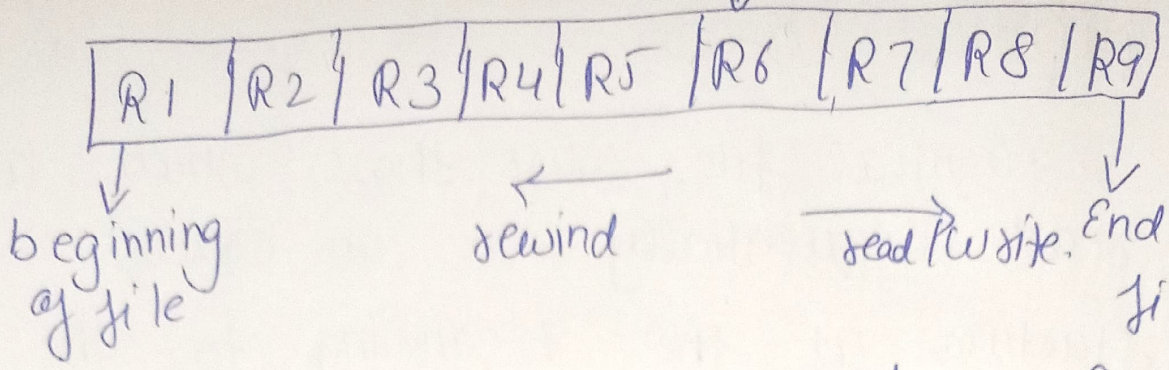
→ The sequential access method has three operation.

- Read operation → read next() [read the next portion of file.
  - It automatically advice a file pointer to next portion.
- Write operation → (write next()] [write & more next portion]
- Rewind : Moving back to the earlier location.



→ based on tape Model.

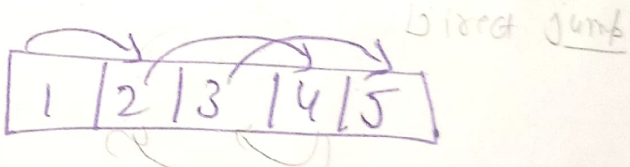
Scanned with OKEN Scanner



- Nine record in the file from R1 to R9.
- The current pointer is at record R6.
  If we want to access Record R8, we have to first access Record R6 & R7.
  - This is one of major advantadge

2) Random (or direct access)

- based on disk Model of a file.
- file whose bytes or record can be read in any order are called random access



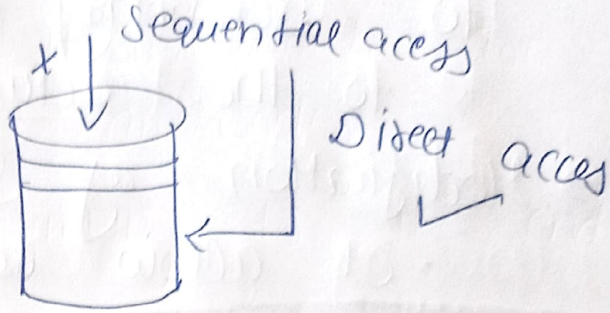
- The record does not need to be in sequence because they are updated directly and rewritten back in same location.

→ operation

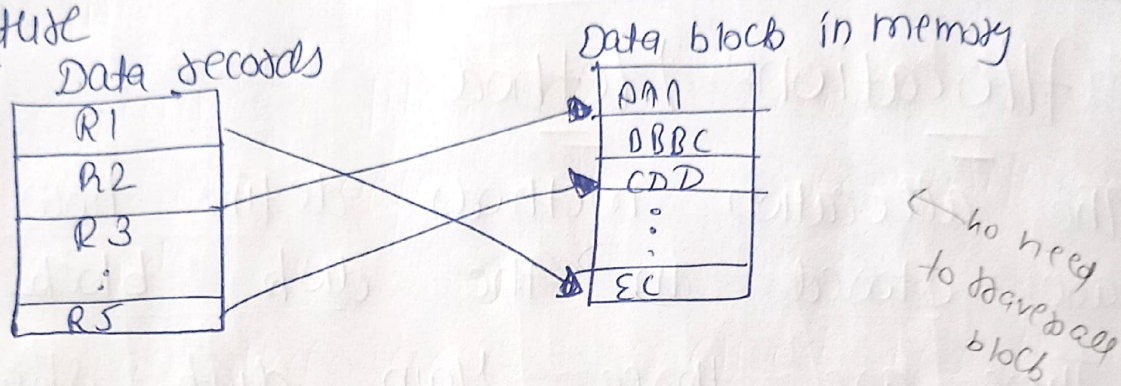
- Read n : used to read the nth block
- Write n. " " write "
- Goto n " directly access "

- Hashing algorithm → used to calculate the record number

eg: Database System

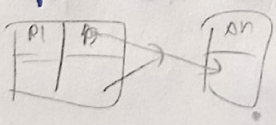


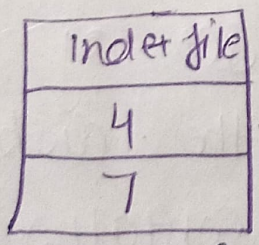
Structure



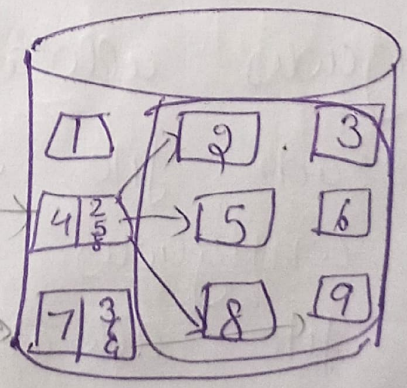
3) Indexed access

- This approach combines the advantages of both sequential access and direct access methods.
- Index contains pointer to the various records. To find an entry in the file, we first search the index, and then use the pointer to access the desired record.

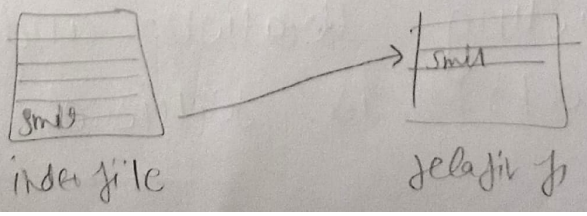




[Primary index block]



[Secondary index files]



index file relation file

- The primary index file contains pointer to secondary index file which point to the actual data items

→ Modification of Sequential access Method.
- It allow random access

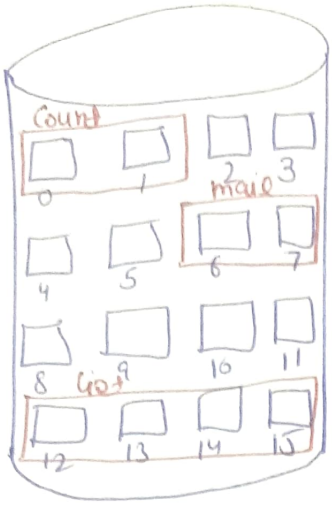
Allocation Method

The allocation methods define how the files are stored in the disk blocks

- These are three Main disk space or file allocation Methods.
  - Contiguous Allocation
  - Linked allocation
  - Indexed allocation
- The main idea behind these methods is to provide:
  - efficient disk space utilization
  - fast access to file blocks.

1) Contiguous allocation
- Each file occupies a set of contiguous blocks on the disk.
- file allocation table contains starting location and length (no. of blocks) are required.

Scanned with OKEN Scanner



Directory

file	Start	length
count	0	2
mail	6	2
list	12	4

- The file "Count" start from the block 0 with length = 2 i.e it requires 2 blocks. hence, in the contiguous allocation Method.
- It occupies. 0,1 blocks

→ If a file starting disk block address is x, then it will be allocated the block with x, x+1, x+2, ...

Advantage

- Simple to implement

Disadvantage

- fragmentation problem. [External]

2) linked list Allocation

- It work just like the linked list.
- The problem with contiguous allocation was that memory remained due to External fragmentation
- The Sol^n to the problem was to allocate the disk block in form of Linked list

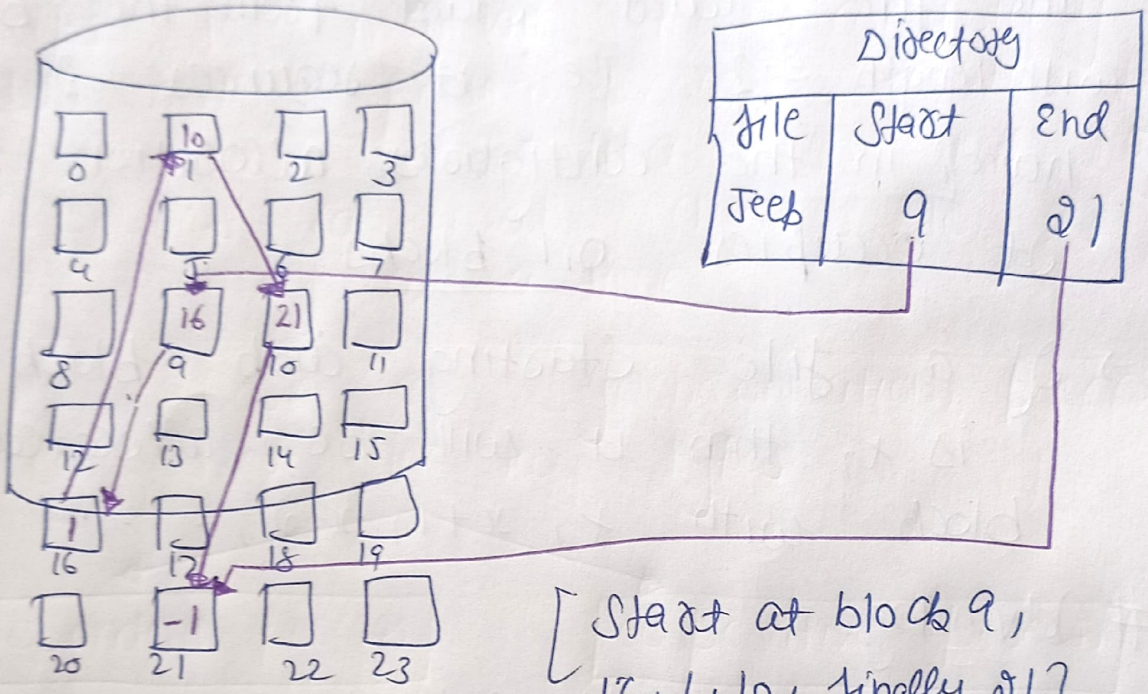
Scanned with OKEN Scanner

where every block was a node.

- The directory entry contain a pointer to the starting and the ending file block.

[Use of file allocation Table]

Pointer required extra overhead.]



[Start at block 9, 16, 1, 10, finally 21]

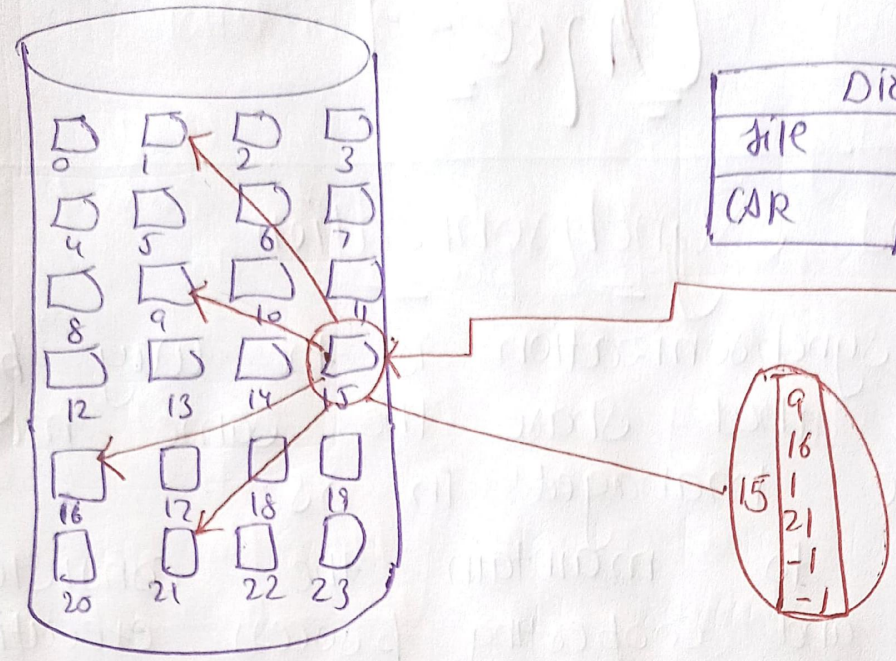
[file are linked together via pb]

3) Indexed Allocation

- Each file has its own index block.
- Indexed allocation solves the problem by bringing all the pointer together into one location. i index block.

→ No fragmentation.

→ No space is consumed by pointers.



master index is used to keep track of all file block.

- This allow for direct access.

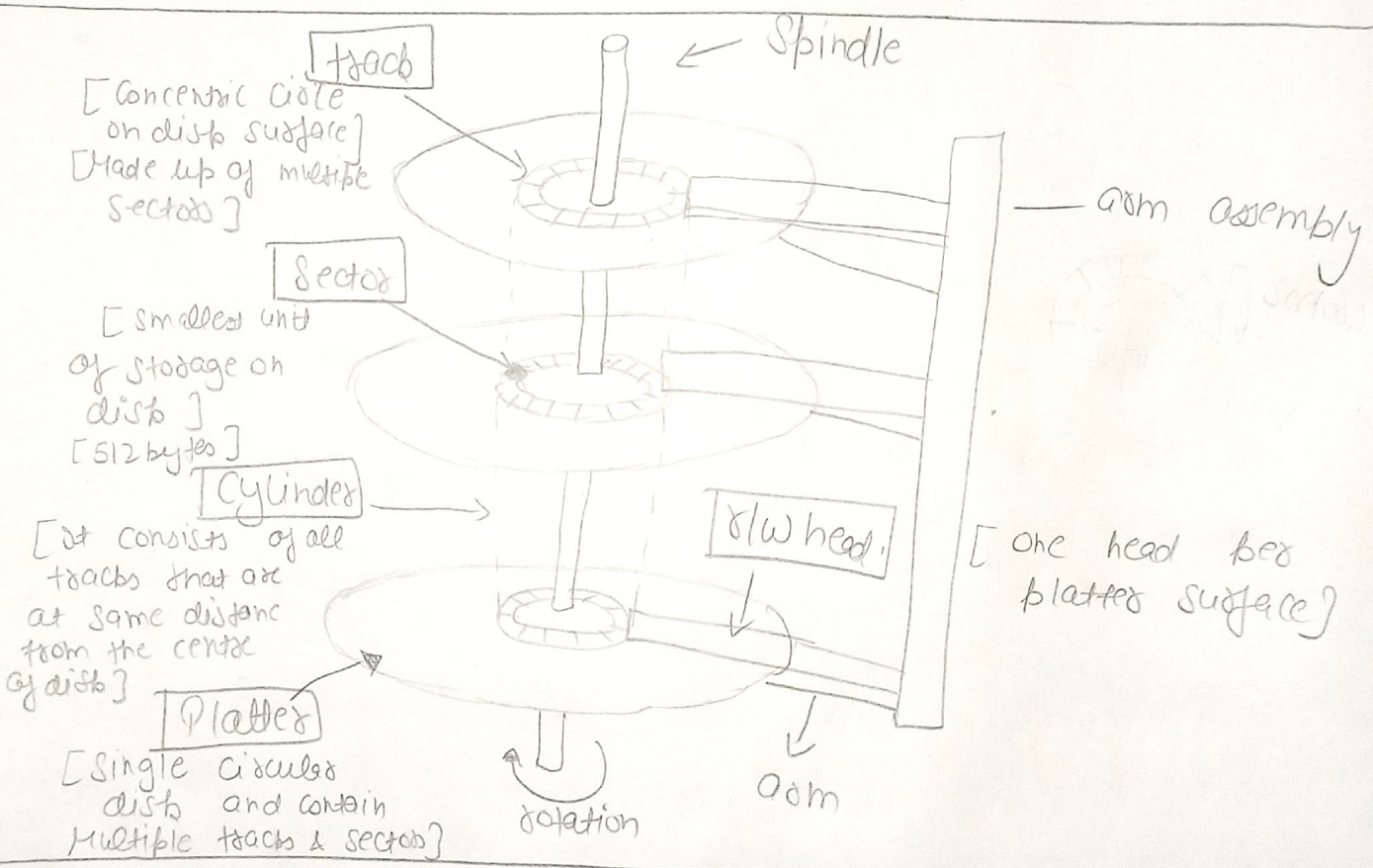
Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

Disk Structure

← Each Disk has a flat circular shape like CD or DVD

- Disk structure refers to how data is organized and stored on a disk storage device such as hard disk drives.
- Disk is a secondary storage device.



- disk contain several platter, each platter is divided into circular shaped tracks
- each track is further divided into sectors.
- A Read - Write head is used to read data from a sector of the magnetic disk.
- The speed of disk is measured as two parts.

Scanned with OKEN Scanner

- Transfer rate
- Random access time

Important term related to disk Scheduling

1) Seek time: Seek time is the time required for the disk arm to move the read/write head to the required track.
   - The scheduling alg. should aim to minimize seek time to enhance performance.

2) Rotational latency: It is the time taken by the desired sector of the disk to rotate into a position so that it can access the read/write heads.
   - So disk scheduling algo that gives minimum rotational latency is better.

3) Transfer time: It is the time to transfer the data.

4) Disk access time: The time required to access the desired location and obtain its contents.

$$= \underbrace{\text{Seek time} + \text{rotational latency} + \text{Transfer time}}_{\text{access time}}$$

5) Disk response time :- It is the average of time spent by each request waiting for the IO operation.

Copyrighted by Codeleap.io
All rights reserved.

Disk Scheduling

- Disk Scheduling is done by operating system to schedule I/O requests arriving for the disk in a system.

→ Objective of disk Scheduling algorithm

- → Minimize the response time / seek time.
- → Maximize the throughput i.e the average no. of request satisfy per unit time.

- fairness
- Minimal traveling head time

→ Goal / purpose

- The Main purpose of disk Scheduling algo is to select a disk request from the queue of I/O request and decide the schedule when this request will be processed.

→ Need

- because a process can make multiple I/O request and Multiple process run at same time
- This algo help in minimizing the seek time by ordering the request made by the processes.

Scanned with OKEN Scanner

→ Disk Scheduling Algorithm

- The list of various disk scheduling algorithm are described, each have some advantage & disadvantage.
- The limitiation of each algo lead to the evolution of a new algorithm.

1) FCFS
2) SSTF
3) SCAN
4) C-SCAN
5) LOOK.

---

1) FCFS :-

Stand for "first - come - first - served".

- This is the simplest disk scheduling alg. where request are served in the order they arrive.
- It involves a lot of random head movement and disk rotation.

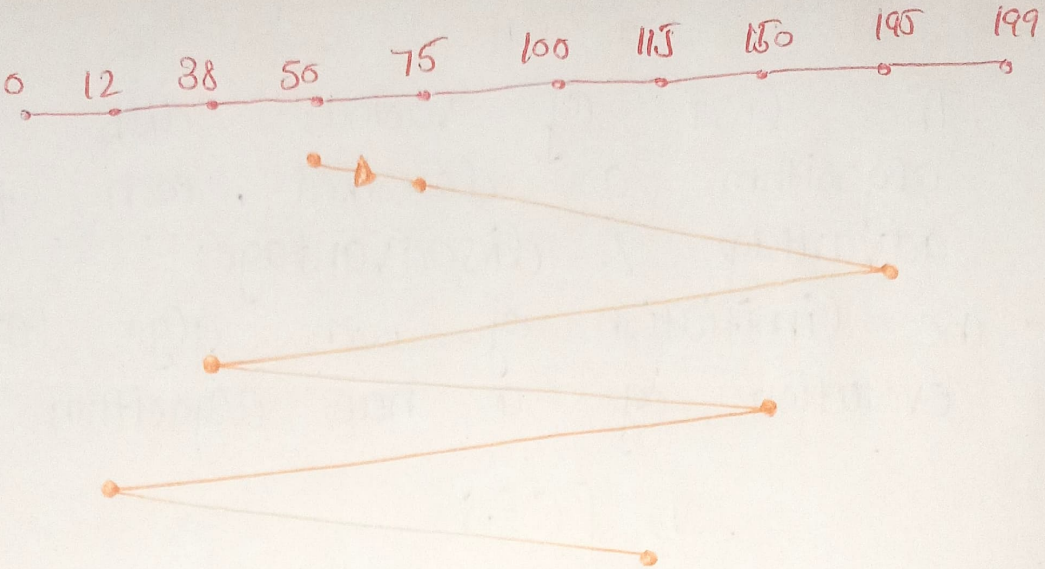
Ex :- → Consider a disk queue with request for I/O to blocks on cylinders.

75, 195, 38, 150, 12, 115

→ If the disk head is initially at cylinder 50. There are 200 cyclinders number from 0 - 199.

→ first move to 50 to 75, then 195, 35, 150, 12 and 115.

Scanned with OKEN Scanner



- Total no of head movement

$|50-75| + |75-195| + |195-38| + |38-150| + |150-12|$
$+ |12-115|$

$= 655$.

Advantage : Easy to implement
fairness.

Dis :- lead to high average seek time.
- Poor performance.

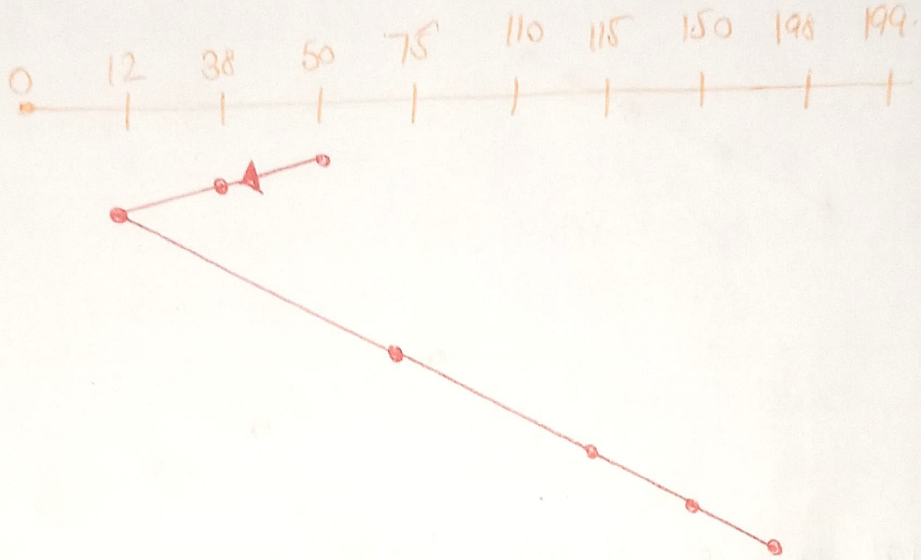
---

2) SSTF :- Stand for "Shortest - Seek - Time - first"
- As the name suggests, it searches for the request having the (least 'seek time') and execute them first.
- It aims to minimize seek time and provide the better performance than FCFS.

- SSTF Select the I/O request that is closest or shortest seek time to the current position of disk arm.

Ex: request sequence = 75, 195, 38, 150, 12, 115

head start at 50



- Total no. of head movements

$|50-38| + |38-12| + |12-75| + |75-115| + |115-150| + |150-195|$

$= 221$

Adv: • More efficient than FCFS

Dis: • Cause starvation for some request.

3) SCAN Scheduling :- In this algorithm, the head starts to scan all the requests in a direction and reaches the end of the disk.

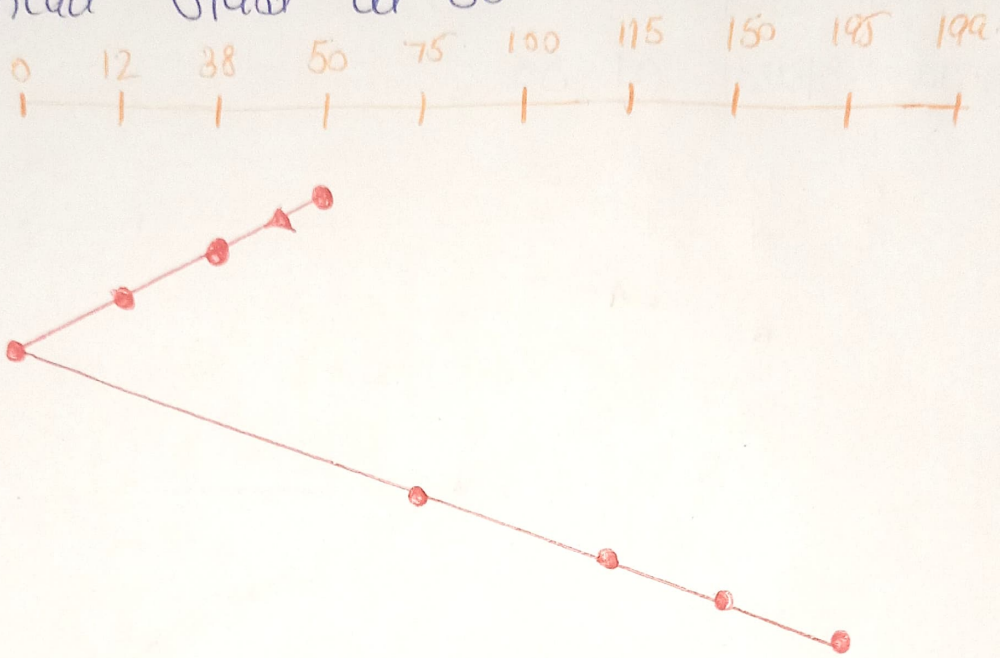
- After that, it reverses its direction and start to scan again the request in its path and serves them.

Scanned with OKEN Scanner

- Due to this feature, this alg is also known as "Elevator algorithm"

Ex: request sequence (75, 195, 38, 150, 12, 115)

head start at 50



- Total no of head Movements

$|50-38| + |38-12| + |12-0| + |0-75| + |75-115| + |115-150|$
$+ |150-195|$

$= 245$

→ Advantage

- High throughput
- Reduce average seek time compared to FCFS.

→ Disadvantage

- May lead to increased waiting time for request at the end of the disk.

Scanned with OKEN Scanner

5) C-SCAN:

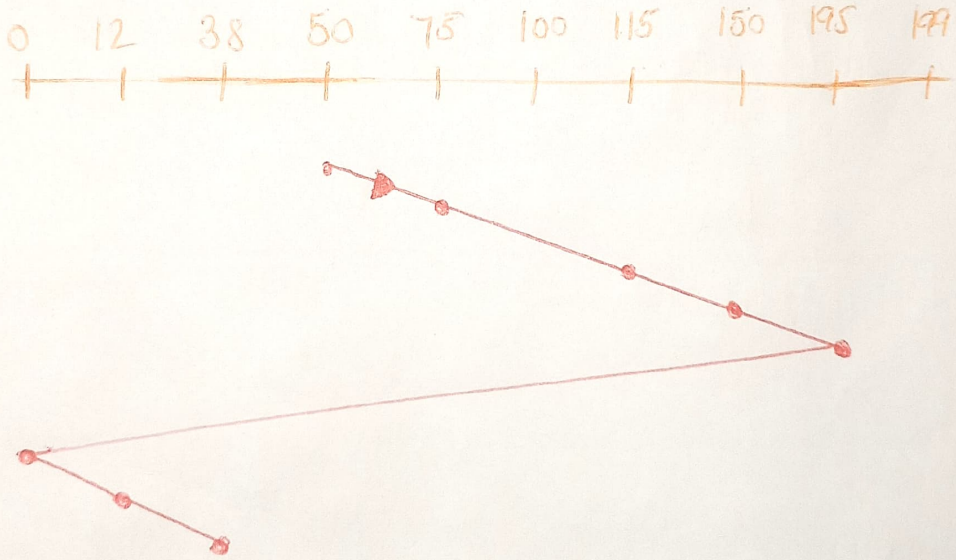
Stands for "Circular - Scan". This algorithm is almost the same as the Scan disk algorithm but one thing that make it different that

- "after reaching the one end, it jumps back to the beginning of the disks and again starts to move the other end of disk but without servicing request along the way"

reversing the head direction.

Ex: Suppose a disk having 200 tracks (0-199)

- request sequence (75, 195, 38, 150, 12, 115)
- head start at 50



- Total no of head movements

$|50-75| + |75-115| + |115-150| + |150-195| + |195-199|$
$+ |199-0| + |0-12| + |12-38|$

$= 386.$

Scanned with OKEN Scanner

Selection of Disk Scheduling algorithm

- Selection for the Disk Scheduling algo involves various factors:

  → System requirement
  - → latency requ.
  - → User Experience.

  → Workload Characteristics
  - → I/O request
  - → Request size & Interarrival time.

  → Performance
  - → Minimize Seek Time
  - → Throughput
  - → Reduced variability

  ↙ improve overall disk performance alg → SSTF, LOOK / C-LOOK.

→ In many disk-Scheduling algo, how we choose best one

- SSTF is common and has a natural appeal.
  - → SCAN or C-SCAN perform better for system that place a heavy load on the disk because they are less likely to have the <u>Starvation problem</u>
  - → With any Scheduling algo, performance depend on the number & type of requests

Scanned with OKEN Scanner

- Either SSTF or LOOK is a reasonable choose for the default algorithm.

Disk Management

- Disk Management is an important functionality provided by the OS which can be used to create, delete, format disk partitions and much more.

→ To access disk management in different OS
- → window OS
- → Linux OS
- → Mac OS.

→ It includes some disk Managements.

1) Disk initialization
2) Booting from disk
3) bad - block recovery

1) Disk initialization :-

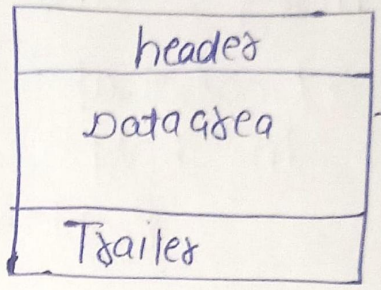
Also known as Disk formatting.

- A new disk is a blank state. We cannot store data on it directly.
- Therefore, in order to allow operation on disk, it should be initialized
- There are three steps of initialization
  - a) Low - level formatting
  - b) Partitioning
  - c) Logical formatting.

Scanned with OKEN Scanner

a) Low-level formatting

→ The purpose of low-level formatting is to divide the disk into sectors. Each sector is capable of storing the same amount of byte, which is known as block size.



→ 256, 512 bytes.

both contain information used by disk controller such as sector no and error-correcting code.

→ In order to increase the efficiency, file system groups blocks in chunks called clusters.

b) Partioning :

↳ Means that the disk is partitioned into one or more groups of cylinders.

- partions in a disk can be treated as separate and independent secondary storage device.

c) logical formatting :

↳ The purpose of logical formatting is to make a file system on the disk.

- In this system, the OS stores the initial file-system data structure onto the disk.

↓ It include allocate space, initial empty directory

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

2) Boot Block

- When the computer is turned on or restarted, the program stored in the initial bootstrap ROM, find the location of the OS kernel from the disk, loads the kernel into memory, and runs the OS start.

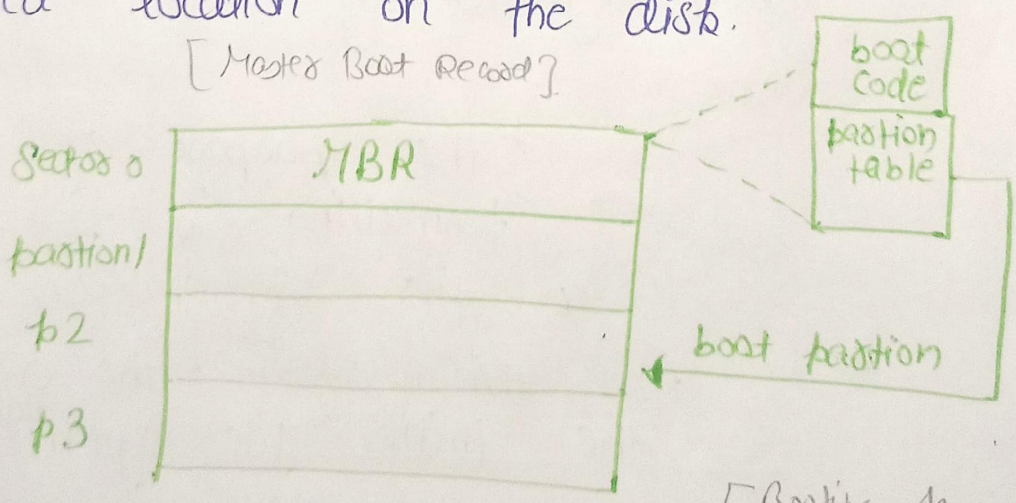
→ bootstrap program or code entirely resided in ROM

- Problem :→ To change the bootstrap code, we need to change the ROM and hardware chip.

- Solⁿ :→ Store only the small bootstrap loader program in ROM, whose only job is to bring in a full bootstrap program in disk
∴ the full bootstrap program can be changed easily.

→ The full bootstrap program is stored in a partion called boot blocks, at a fixed location on the disk.

[Master Boot Record]



[Booting from disk in window 2000]

Scanned with OKEN Scanner

3) Bad Block

- One or more sector become defective. Most disks even come from the factory with bad blocks. If a large no. of blocks go bad then the entire disk will need to be replaced.
- Bad block's data are generally lost and need special application to recover them or restore them from backup copy.
- There are different ways to treat bad blocks on the disk.

a) Manually:- Bad blocks on some disk with IDE controller can be handled manually but sometime the data could be recovered and may be lost.

b) By Sectors Sparing:- The controler (SCSI) maintains a list of bad block on the disks and logically replaced with one of spare sectors or forwarding.

- as an alternative to sector sparing, some controller can be instructed to replace a bad block by Sector Slipping.

Scanned with OKEN Scanner

- Disk Management involves organizing and maintaining the data on a storage device such as hard disk.

→ Advantage

1) Improved organization and management of data
2) Efficient uses of available storage space
3) Improved data integrity and security.

→ Dis:

- overhead due to disk Management costs.
- data loss due to errors.

Swap Space Management

← The area on disk where the swapped-out process are stored.

- Swap Space Management is another low-level task of OS.
- The Main goal for the design and implement of Swap Space is to provide the best throughput for the virtual-memory system.

→ Advantages

1) Increased memory capacity
2) Improved system performance
3) flexibility

→ Dis: ↳ Slower access time
Increase disk usage

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

→ Swap - Space Use

- Swap - Space is used by the different operating - system in various ways.
- The system which are implementing swapping may use swap space to hold the entire process which may include image code and data segments.
- The need of swap space on a system can vary from a megabytes to gigabytes but it also depends on the amount of physical memory.
- Some system like linux use multiple swap spaces.
- overestimate of swap space is recommended, since memory may waste here but no other harm.

table show different system using amount of swap space. →

System	Swap- Space
1) Solaris	Equal amount of physical Memory
2) Linux	Double the amount of P.M.

→ Swap - Space Location

- A swap - space can reside in one of two places
  - → file - system
  - → Raw partitions.

Scanned with OKEN Scanner

→ In file System :- If the swap space is a large file within the file system, we can use all the file operation like create, rename etc. to manipulate it.

→ This is an easy implementation but inefficient, narigating the directory structure and disk allocation data take time, extra disk accesses.

Solⁿ by caching the block location info in physical memory. → External fragmentation problem arise

→ In Raw partions :- If swap-space is create in a separated disk partition, a separate swap-space manager can be used to allocate or de-allocate the blocks:

- faster and more efficient.

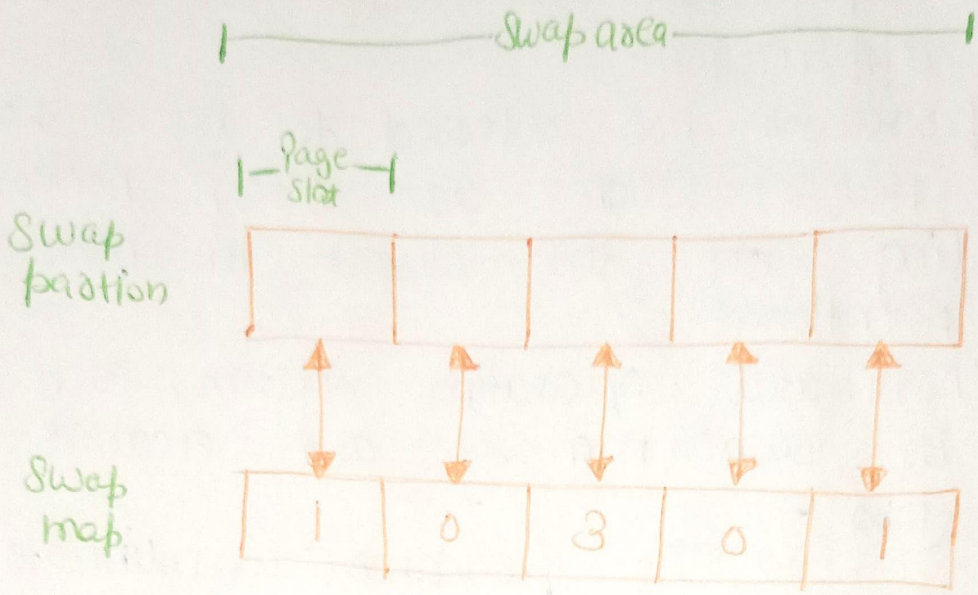
→ Internal fragmentation may increase, but it is acceptable, because the life span of the swap space is shorter than the files in the file system.

- Some OS are flexible and and can swap both in raw partitions and in the file system space, ex: Linux.

Scanned with OKEN Scanner

→ Swap - Space Management

- How managed in Linux, Linux allows one or more swap area.



- A swap area may be in either in swap file on regular file system or dedicated file.
- Each swap space area consist of 4-kb page slots, which are used to hold the swapped pages.
  - → Zero indicate free slots
  - → Non zero refer to how many processes have a mapping to that particular block.

→ amount of physical RAM
→ system workload
→ specific requirement for OS / recommendation
→ Disk space availability.

} factor decide swap space size

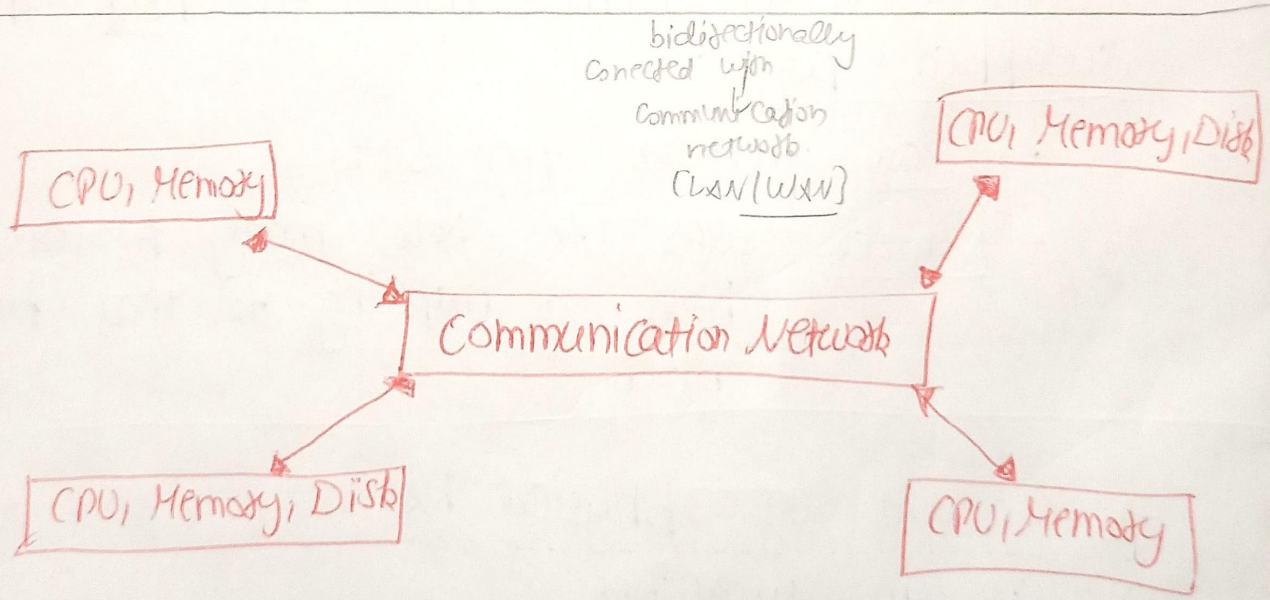
Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

Distributed System

- A distributed system uses multiple central processor to serve multiple real-time application.
- The processor are referred to by a number of names, such as sites, nodes, computer depending on the context in which they are mentioned.

→ In distributed operating system, each processor has its own memory and executed its own jobs ↳ The processor communicate with one another through various communication media. Such system are also known as loosely-coupled system.



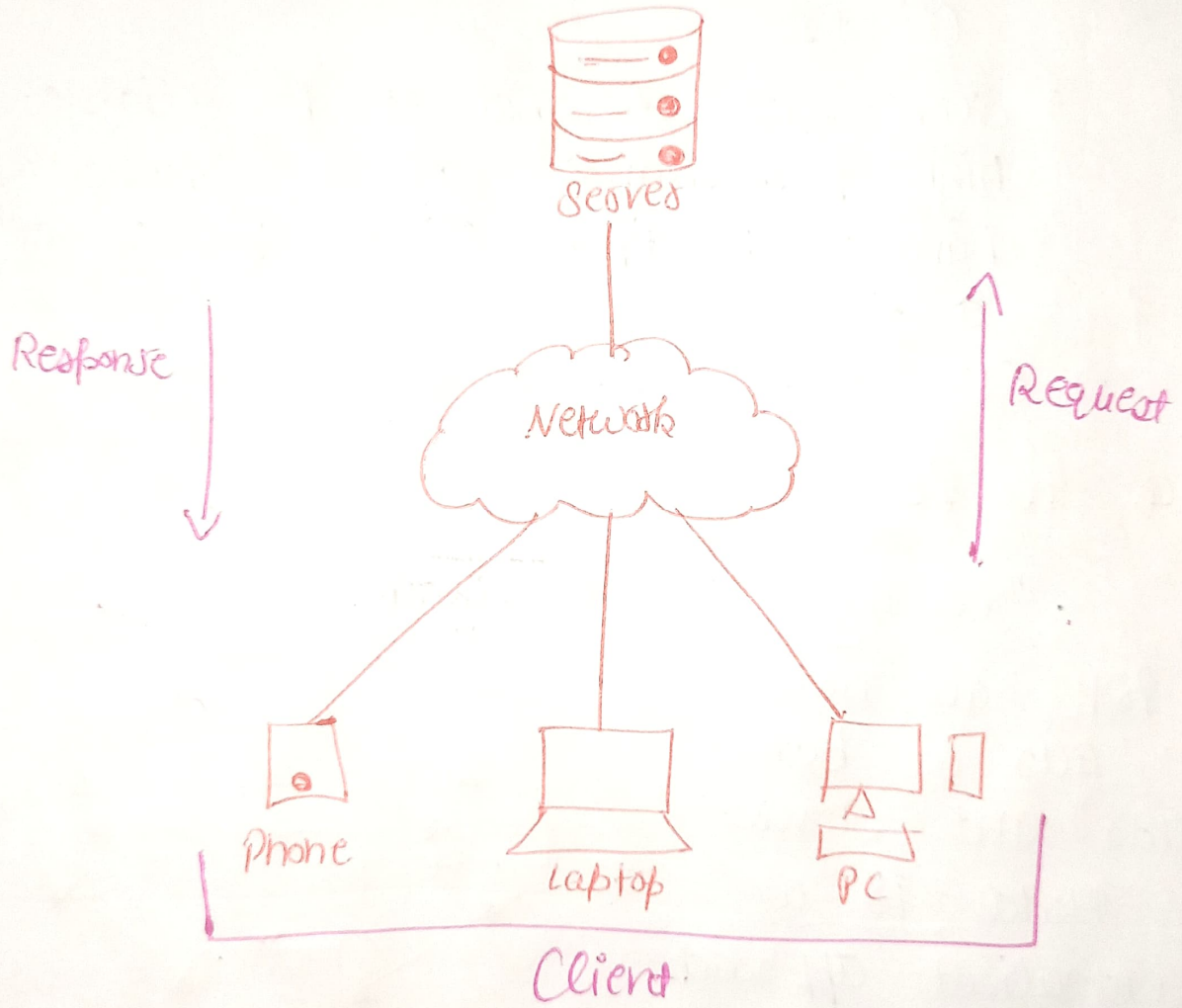
→ The design of distributed operating system is based on two models

1) Client - Server
2) Peer - to - Peer Model.

Scanned with OKEN Scanner

→ Client - Server Model

- The Client - Server Model used to facilitate communication and resource sharing among multiple interconnected computer through internet.
- It divides in two categories
  - → Client
  - → Server



→ Client :→ Clients are the end-user device that request service or resource from server or sends the request for data to server through internet.

→ "Tightly Coupled OS".

Scanned with OKEN Scanner

- They initiate communication with server to access data or execute their requested tasks.

→ Servers :- Server are computer or software application that provide service to Client upon their request.

provide service

→ Server accept the Client request and deliver response back to Clients.

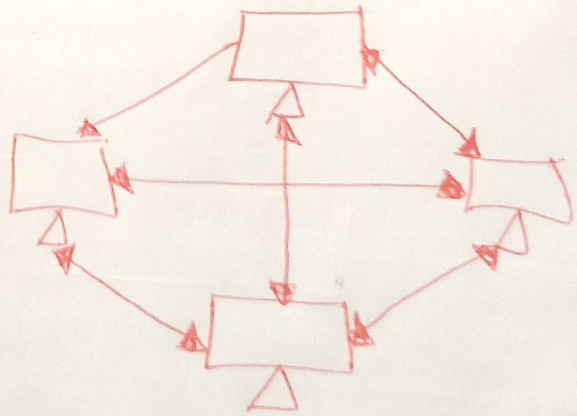
→ Server are designed to be reliable, high-performance system capable of handle multiple client request at a time

→ Ex: Web Server, database Servers

→ Peer to Peer Model

- "Loosely Coupled System".

- In P2P, all nodes can acts as both Clients and Servers.



- P2P Model is a decentralized approach where multiple nodes or computer within the network, acts a both Client and Server
- Each Nodes has equal privileges and responsibilities and they communicate directly with each other without the need for a central server.

Scanned with OKEN Scanner

→ Scalability
→ resource Sharing } Advantages

→ Middleware

- Middleware acts as an intermediary layer b/w the OS and application running on it.
→ Main goal of Middleware is to provide transparency to distributed application. This means hiding the details of network.
→ It provide authentication and encryption mechanism to safeguard data.

→ Advantages

- Resource Sharing
- Enhanced performance
- Improved reliability
- Better Service to the Customes

→ Disadvantage

- The cost of setting up distributed system is high
- The failure of main system might affect the complete system.
- Complex

→ Solaris
→ Macos } Et.
→ Locus

Copyrighted by Codeleap.io
All rights reserved.

Scanned with OKEN Scanner

→ Distributed Operating System

- It is a software over a collection of independent networked, communication and physically separate computational nodes.

- Data Migration

  - Data migration involves moving remote data to the host where a request is made.
  - Data Migration transfer data by transferring entire file or transferring only those portion of file necessary for the immediate task.

eg: Suppose that a user Node A want to access data from Node B. There are two method for the system to transfer the data.

1) Transfer the Entire file

- When Node A request a file from Node B even if Node A only needs a small part of the file, Node B send the whole thing.
- This Method is inefficient, Expensive to ensure consistency.

Scanned with OKEN Scanner

ii) Transfer only Desired position of the file

- When Node B only transfer the part of file that node A needs.
- If Node A changes something, it only send back the part that were modified

→ Cost effective

Computation Migration

→ In Computation Migration, the Computation migrates to another location rather than the data. Migrating computation may be efficient under certain circumstances.

→ Computation migration, one computer may require another computer states

↳ It is more efficient and safely getting the information without needing to know all its private details.

↳ Remote Procedure Call is one Method used for this, where computer can ask each other to perform tasks and share information i.e providing communication b/w computers.

↳ An RPC uses Datagram protocol to execute a routine on remote system.

Scanned with OKEN Scanner

Ex: A two computer A and B

→ Process P on computer B to send a message directly to computer A, asking it to do a specific task

→ Computer A, then create a new process to handle the task requested by Process P on computer B.

→ After new process is done, it send the result back to Process P on computer B.

÷ Dis ↳ Costly.

Copyrighted by Codeleap.io
All rights reserved.

Process Migration

→ Process Migration is one in which a process can be moved during its execution

→ When a process is submitted for execution, it is not always executed at the site at which it is initiated.

→ The entire process, may be executed at different sites. This scheme may be used for several reasons.

1) Load balancing :→ Distribute processes across network to even the workload.

2) Computation Speedup :→ Subprocesses can run concurrently on different sites.

Scanned with OKEN Scanner

3) Hardware Preference : Process execution may require specialized processor, that is available at a particular site.

4) Software Preference : Required software may be available at only a particular site

→ There are 2 technique that can be used to move processes in CN

- System can attempt to hide the fact that the process has migrated from the client
- the user is allowed to specify explicitly how the process should migrate.
  ↳ satisfy hardware and software preference.

Copyrighted by Codeleap.io
All rights reserved.

Network operating System

- A Network operating system provides an environment where users, who are aware of the multiplicity of machine, can access remote resources by either logging in to the appropriate remote machine or transferring data from the remote machine to their own machine.

Scanned with OKEN Scanner