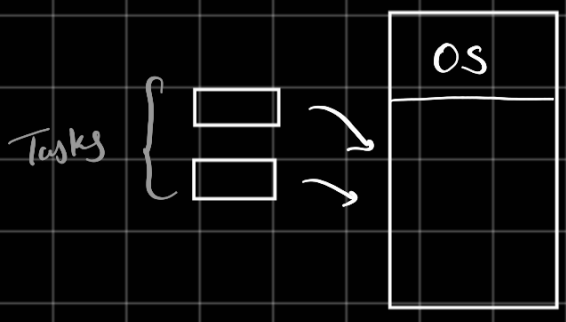
Operating System →

A software that works as an interface between a user and the computer.



for performing any sort of task an environment is created.

Batch OS →

So we create a group/batch of tasks performing on similar environment.

Types of OS →

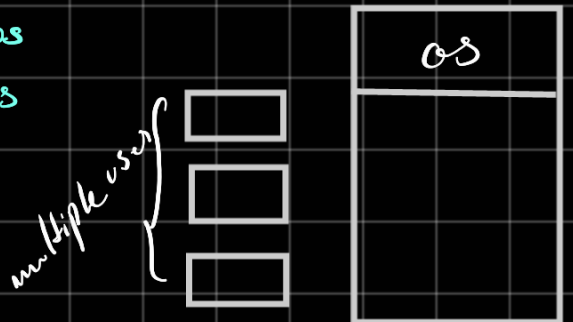
1. Batch OS →
2. Interactive OS →

One that allows users to directly interact with the operating system.

3. Multiuser

Multiple users using the operating system.

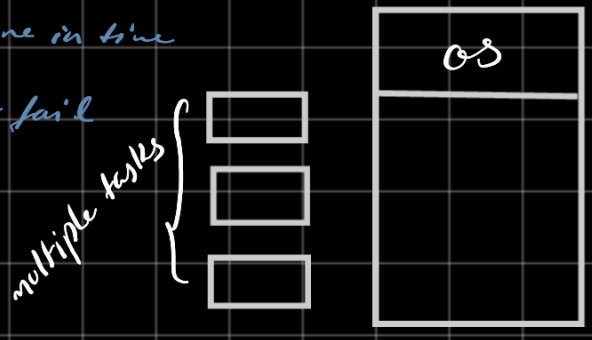
4. Multi tasking
5. Multi threading OS
6. Distribution OS
7. Real time OS.



for OS there is no difference.

Here for "multiuser" and "multitask" are same.

- Hard: under specific time — file if not done under time then 'fail'
- Soft: under range time — If not done in time range will not fail



Degree of multiprograming →

OS
1
2
3

degree of multiprograming → '3'

Burst Time →

Time required by processor for execution.

(on ram)

Batch OS →

Advantages →

- Repeated jobs done fast.
- Best for large organization.
- Doesn't required special hardware

Disadvantages →

- Lack of interaction between user and job.
- CPU often idel.

Multitasking →

Advantages →

- Reduce idle CPU
- Quick response

Disadvantages →

- Security.
- Resource consumption.
- Switching between tasks.

Destributed →

Advantage →

- Reduction of load on host computer
- If a system crashes, the system will survive.
- Sharing of data

Disadvantage →

- Complex System
- Security problems
- Networking problem

Real Time →

Advantages →

- Easy Memory Allocation
- Error free

Disadvantages →

- Not easy to program
- Expensive

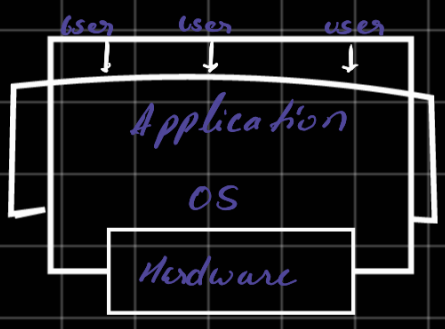
Functions of Operating System →

- Process management.
- Time management
- Memory management
- File management
- Input/Output management

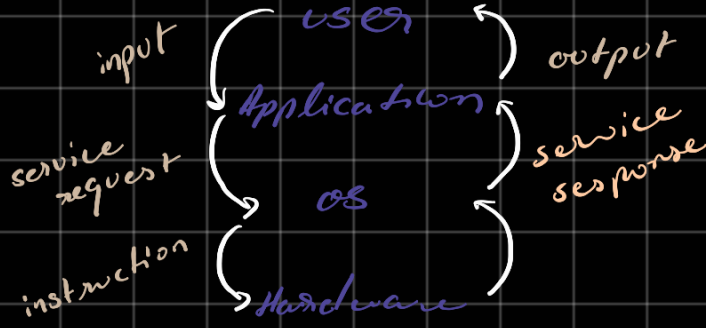
Response Time →
Total amount of time taken to respond to a request for services.

- Turn around
- Waiting

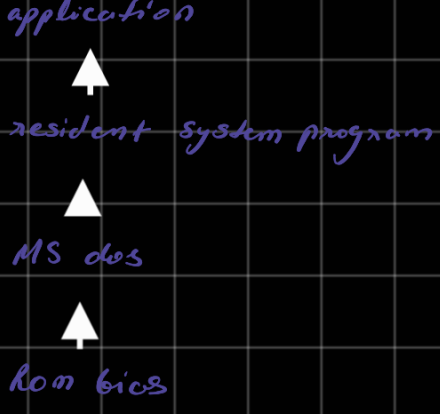
Static view of OS. →



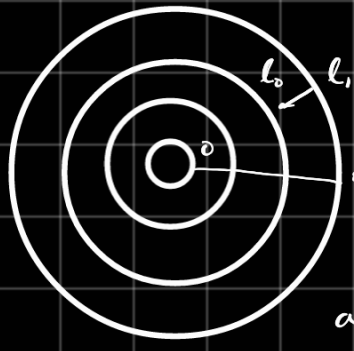
Dynamic →



Simple Structure →



Layered Approach →

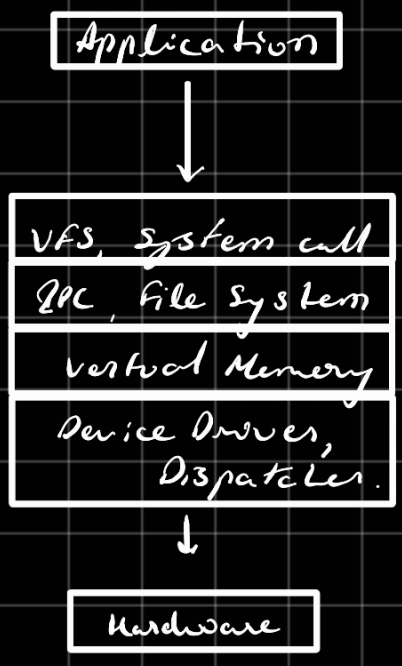


l_1 can only access l_0, but l_0 can't access l_1.

Kernal → Inner most layer of an OS which interacts with the OS.

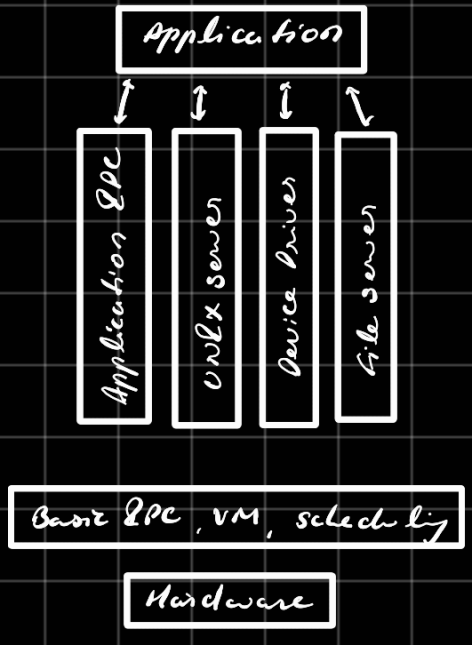
Monolithic / Microkernal →

Monlithic →



- User and Kernal are in same place.
- Larger than Micro Kernal.
- Hard to extend

Micro kernal →



- seperate Kernal and user
- smaller in size
- slower

system call →

It's a way for a user program to interact with the OS. The program requests several services and the OS responds by invoking a series of system calls to satisfy the request.

① Process

② Memory

Types of System call →

- Process control
  - create process, terminate process
  - end
  - load, execute
  - Debugger
  - Locks

- File Mangement
  - delet, create file
  - open, close
  - read, write

- Device Management
  - request, release
  - read, write
  - get, set attributes

- Protection
  - allow any deny user access
  - get, set permission

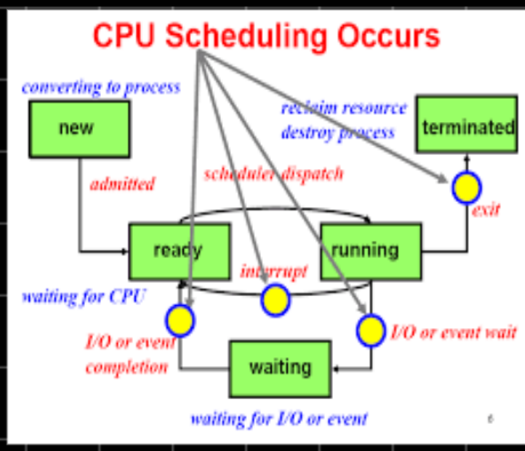
Remote Procedure Calls →

A remote procedure call is an inter-process communication technique for client-server application.

Also known as Subroutine call or function call.

Process → Code
data
input/output

states → New state
Running state
Blocked or waiting
Ready state
Terminate

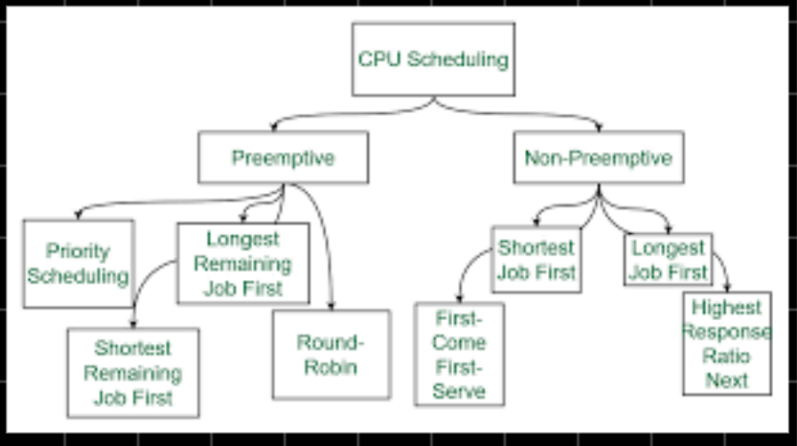


Process Controll bon →
Contain all the information about the process.

CPU Scheduling →
Process of deciding which process will own the CPU to use while another process is suspended.

cpu utilization decrease
throughput
Bust time / Respons time / waiting time / turn around

Preemptive / Non-preemptive →



Content switching →
Involves storing the content/state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier.

Throughput →
How many units of information a system can process in a given amount of time.

S.No.	Preemptive Scheduling	Non-Preemptive Scheduling
1.	The CPU is allocated to the processes for a certain amount of time.	The CPU is allocated to the process till it ends it's execution or switches to waiting state.
2.	The executing process here is interrupted in the middle of execution.	The executing process here is not interrupted in the middle of execution.
3.	It usually switches the process from ready state to running state, vice-versa, and maintains the ready queue.	It does not switch the process from running state to ready state.
4.	Here, if a process with high priority frequently arrives in the ready queue then the process with low priority has to wait for long, and it may have to starve.	Here, if CPU is allocated to the process with larger burst time then the processes with small burst time may have to starve.
5.	It is quite flexible because the critical processes are allowed to access CPU as they arrive into the ready queue, no matter what process is executing currently.	It is rigid as even if a critical process enters the ready queue the process running CPU is not disturbed.
6.	This is cost associative as it has to maintain the integrity of shared data.	This is not cost associative.

First Come First Serve → (N.P)

	Arrival time	Burst time
P_1	0	3
P_2	0	2
P_3	0	4
P_4	0	1

Avg waiting Time →

waiting Time

$P_1 \rightarrow 0$
$P_2 \rightarrow 3$
$P_3 \rightarrow 5$
$P_4 \rightarrow 9$

$\frac{17}{4} = 4.25$ s

P_1	0	3
P_2	1	2
P_3	2	4
P_4	3	1

Turn around Time = waiting time - Arrival time

⇒ P_1 P_2 P_3 P_4
0 3 5 9 10 ← waiting time

$$\Rightarrow \frac{(0-0)+(3-1)+(5-2)+(9-3)}{4}$$

$\Rightarrow 11/4$

	AT	BT
P_1	0	30
P_2	1	20
P_3	2	5
P_4	3	6

P_1 P_2 P_3 P_4
0 30 50 55 61

$$\frac{(0-0)+(30-1)+(51-2)+(55-3)}{4}$$

$$= \frac{129}{4} \Rightarrow \underline{32.25}$$

Shortest Job First → (Non-preemitive)

	AT	BT
P_1	0	2
P_2	1	3
P_3	2	5
P_4	3	2

P_4 has less burst time

P_1 P_2 P_4 P_3
0 2 5 7 12

$$\frac{(0-0)+(2-1)+(5-3)+(7-2)}{4}$$

$= 8/4 \Rightarrow \underline{2}$

	AT	BT
P_1	0	6
P_2	3	4
P_3	3	4
P_4	4	4

P_1 P_2 P_3 P_4

0 6 10 14 18

$$\frac{(0-0)+(6-3)+(10-3)+(14-4)}{4}$$

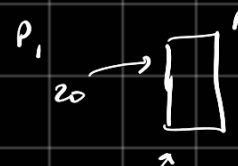
$$= \frac{3+7+10}{4} \Rightarrow \frac{5}{}$$

As bus time is same for P_2, P_3, P_4, so now we will check for arival time. Arival time of P_2, P_3 is less than P_4, so, P_2 and P_3 go first than P_4.

P_2, P_3 are having same 'AT', 'BT', then we will follow given sequence.

① For bigger problems will take more time.

② Can't change at run time.



Shortest Job First → (Preemitive)

	AT	BT
P_1	0	5
P_2	1	3
P_3	2	1
P_4	3	2

P_1 P_3 P_4 P_2

0 5 6 8 11

$$\frac{(0-0)+(5-2)+(6-5)+(8-1)}{4}$$

} Non-Preemitive

P_1 P_2 P_3 P_2 P_4 P_1

0 1 2 3 5 7 11

} Preemitive

	AT	BT
P_1	0	6
P_2	1	3
P_3	2	1
P_4	3	2

P_1 P_2 P_3 P_2 P_4 P_1

0 1 2 3 5 7 12

} Preemitive

$$\frac{(7-1)+(3-1-1)+(2-2)+(5-3)}{4}$$

$$\Rightarrow \frac{6+1+0+2}{4} \Rightarrow 9/4$$

Priority →

	AT	BT	Priority
P_1	0	6	1
P_2	1	3	2
P_3	2	1	3
P_4	3	2	1

if order of priority not mentioned then
1 → 7
increasing

Non-Preemitive

P_1 P_3 P_2 P_4
0 6 7 10 12

$\frac{(0-0)+(6-2)+(7-1)+(10-3)}{4}$

2) $\frac{4+6+7}{4}$ 2) $17/4$

Priority → BT → AT → sequence

Preemitive

P_1 P_2 P_3 P_2 P_4 P_1
0 1 2 3 5 8 12

Avg.T => $9/4$

Round Robin → (Premitive)

TQ → Time Quantum

if TQ is increased significantly then, RR will act like "FCFS".

TQ = 3

	AT	BT
P_1	0	6
P_2	1	7
P_3	2	4
P_4	3	3
P_5	4	8

P_1 P_2 P_3 P_4 P_1 P_5 P_2 P_3 P_5 P_2 P_5
0 3 6 9 12 15 18 21 22 25 26 28

WT - AT WT - AT WT - AT WT - AT WT - AT

$(12-3)-0 + \left(3+(18-6)+(25-21)\right)-1 + \left(6+(21-9)\right)-2 + (9-3) + \left(15+(22-18)+1\right)-4$

$$\frac{9+17+16+6+14}{5} = \underline{\underline{12.4}} \text{ Avg. T}$$

$$\frac{BT+WT}{\text{Total 'p'}} = \text{AVG Turn around time}$$

Process Synchronization →

1. Process will not affect other process (Indipendent)
2. Process will affect other process

Race Condition

happens under critical section.

Critical section Programming (CSP)

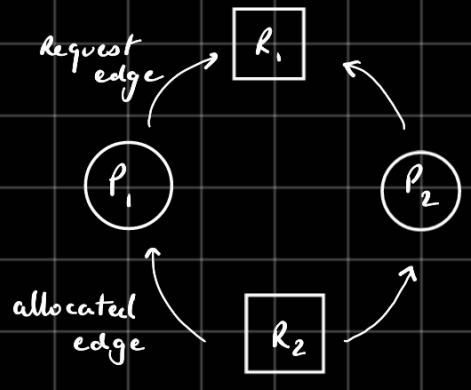
under CS only one process is allowed to function at point of time.

Dead lock →

Resource allocation graph →

Process → ○

Resource → □



Condition's for dead lock →

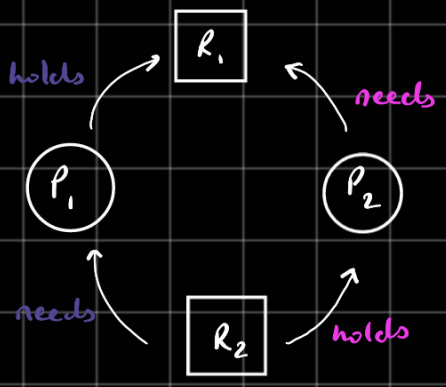
1. **Mutual exclusion**
   - Only one process can use a resource at any given time.
   - The resources are non-shasable.

2. **Hold and wait**

   A process is holding at least one resource at a time and is waiting to acquire other resources held by some other process.

③ Circular wait

A set of processes are waiting for each other in a circular fashion.



④ No-preemption

The resource can be released by a process voluntarily.

i.e. After execution of the process.

Handling Method →

i) Prevention
- ME
- Hold & W
- C W
- No P.

} make any of these false.

ii) Avoidence
- single instance (if resource is of single type)
- Multi instance — Banker's Algorithm

iii) Detect and Recovery

Detect, Kill process, Recover the process again

iv) Ignore (Restart) / Ostrich method

Deadlock Detection and Recovery →

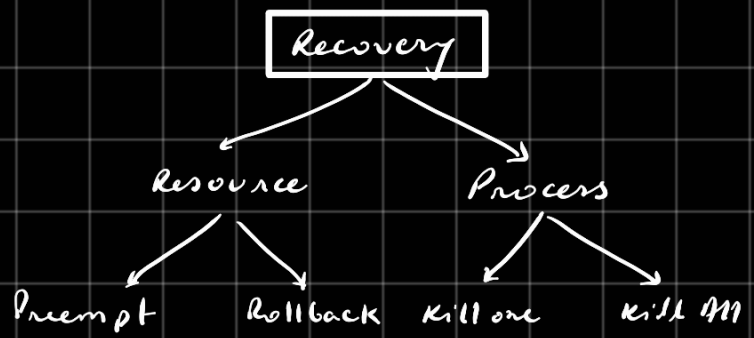
Detection

(RAG) ← use — single instanced

Multi instanced — use → (Banker's Theorem)

single instanced → Delete cycle

Multi instanced → Safety Algorithm (Banker's Theorem)



Preempt

- Taking the resource from the owner, expecting that its process will end sooner.

Rollback

- OS can rollback to previous safe state.

Banker's Theorem →

	Allocation			Max			Need More			available in system			
	A	B	C	A	B	C	A	B	C	A	B	C	after allocation
P_0	0	0	1	8	4	3	8	4	2	3	3	2	
P_1	3	2	0	6	2	0	3	0	0				
P_2	2	1	1	3	3	3	1	2	2				

Resource holding by processes

Resource required to complete the process

320
332
652
211
863
864

need ≤ available

P_0 842 ≤ 332 ✗ not

P_1 300 ≤ 332 ✓ T

new available

P_2 122 ≤ 652 ✓ T

new available

P_0 842 ≤ 863 ✓ T

new available

864

(P_1, P_2, P_0) execution orders to avoid dead lock.

	Allocated				Max				Needed				Available			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	
P_0	0	0	1	2	0	0	1	2	0	0	0	0	1	5	2	0
P_1	1	0	0	0	1	7	5	0	0	7	5	0				
P_2	1	3	5	4	2	3	5	6	1	0	0	2				
P_3	0	6	3	2	0	6	5	2	0	0	2	0				
P_4	0	0	1	4	0	6	5	6	0	6	4	2				

	need	≤	available	
P_0	0000		1520	✓ T
			new available	
			1532	
P_1	0750		1532	✗ not
P_2	1002		1532	✓ T
			new available	
			2886	
P_3	0020		2886	✓ T
			new available	
			2 14 11 8	
P_4	0642		2 14 11 8	✓ T
			new available	
P_1	0750		2 14 12 12	✓ T
			new available	
			2 14 13 16	✓ T

$(P_0, P_2, P_3, P_4, P_1)$

	AT	BT	TQ = 4
P_1	0	5	
P_2	1	6	
P_3	2	3	
P_4	3	1	
P_5	4	5	
P_6	5	4	

P_1 P_2 P_3 P_4 P_5 P_1 P_6 P_2 P_5

0 4 8 11 12 16 17 21 23 24

P_1 ✓
P_2 ✓
P_3 ✗
P_4 ✗
P_5 ✓
P_1 1
P_6
P_2
P_5

With Preemption

	AT	P	BT
P_1	0	1 (Low)	6
P_2	1	2	3
P_3	2	3	2
P_4	3	4	4
P_5	4	5 (High)	3

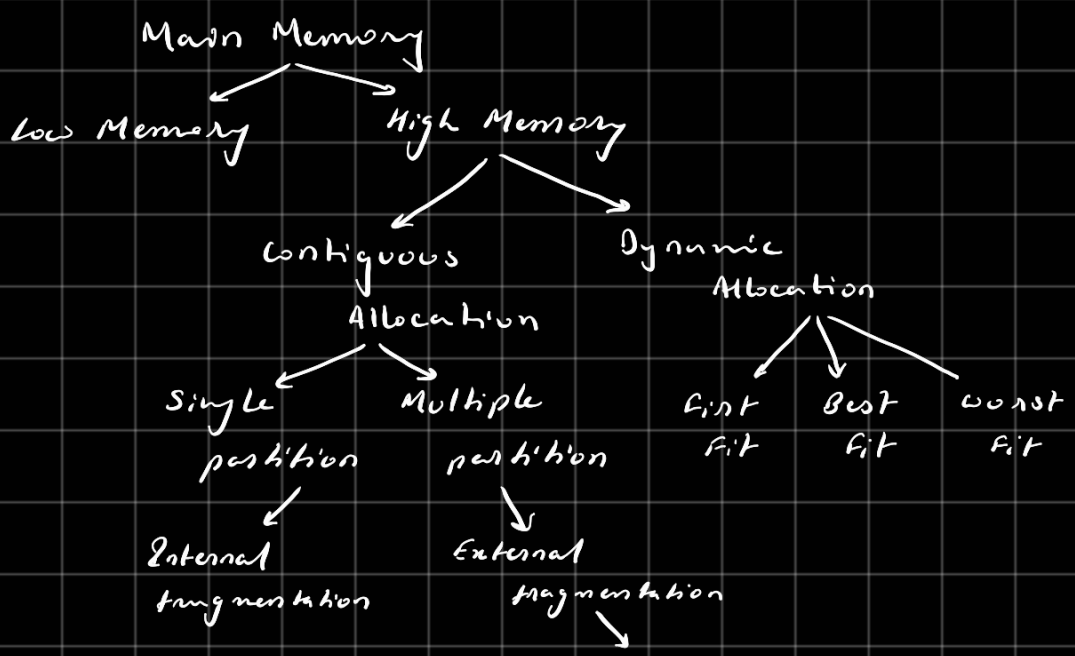
P_1 P_2 P_3 P_4 P_5 P_4 P_3 P_2 P_1

0 1 2 3 4 7 10 11 13 17

$$\frac{12 + 9 + 7 + 3 + 0}{5} = 6.2$$

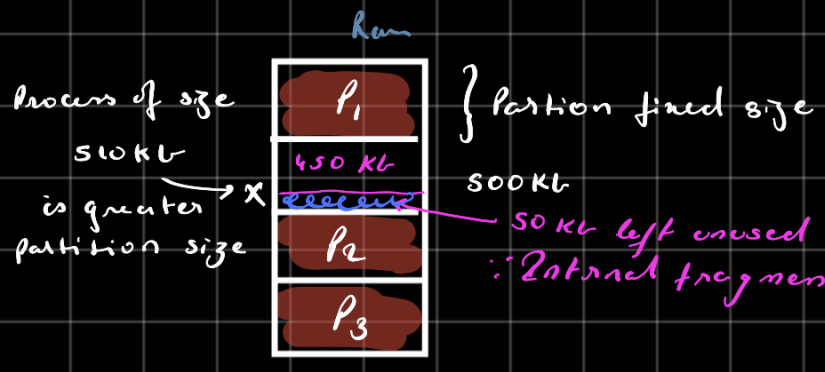
Unit - 2

Memory →



Contiguous Memory Allocation →

Static Memory Allocation Techniques →

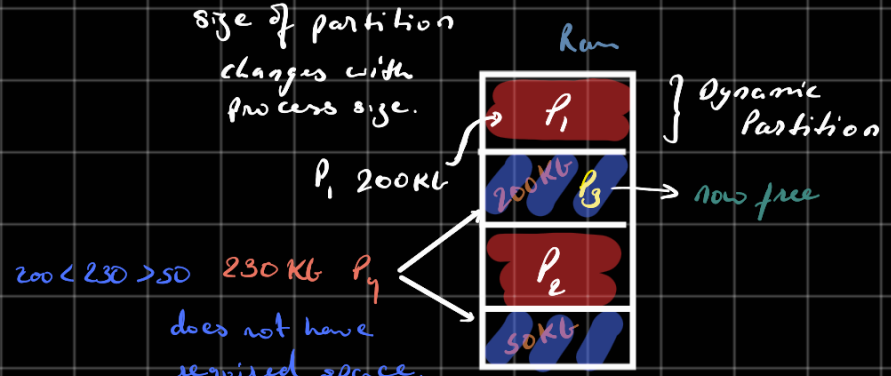


- Size of process is fixed
- degree of multiprogramig is fixed
- Internal fragmentation

- Internal fragmentation

if hole is of "500 KB" and process is of 450 KB then "50 KB" of space is left unused.

Dynamic Memory Allocation Techniques →



- External fragmentation

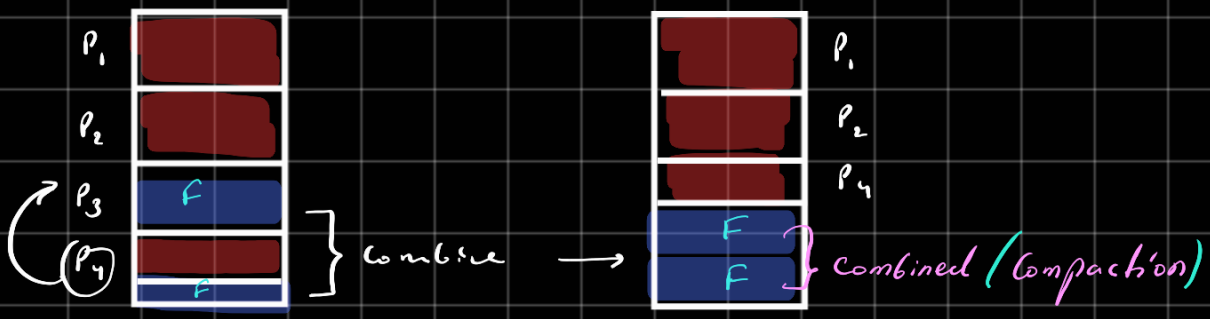
→ compaction

- External fragmentation

if process size is "560 KB" but available is "500 KB" or "60 KB". Therefore the available size is not ≥ process.

- Compaction

by using compaction we can combine the empty space to form one, combine '500 KB + 60 KB' to form a new hole of size ≥ process size.



- To combine we need to cut and paste 'P4' at different location. (location above the available space)
- With cost's time.

Dynamic Memory Allocation →

1) Best fit

If hole size is equal or very close to space required by process.

Hole/Partition − Process size = min difference

- Compare all the holes every time for every single process.

2) First fit

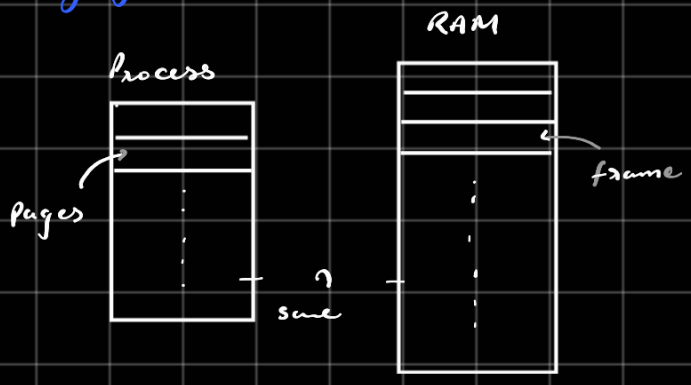
Process takes the first available space in RAM.

- Its probability of getting best fit is very low.

3) Worst fit

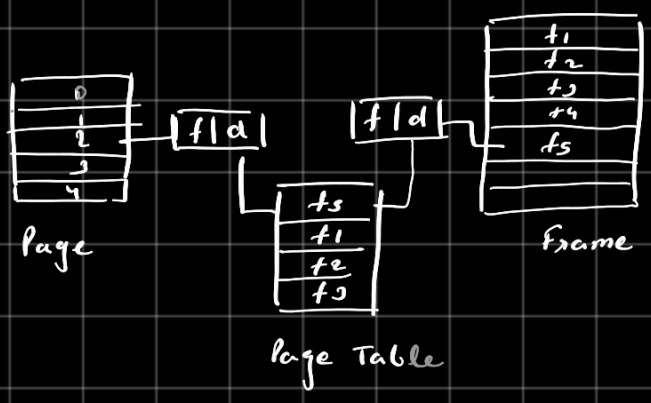
Hole/Partition − Process size = max difference

Paging (Non continuous)



N - number of fixed Partition of small size.

Page fault →
Page hit / Page miss
if page does not exists in RAM



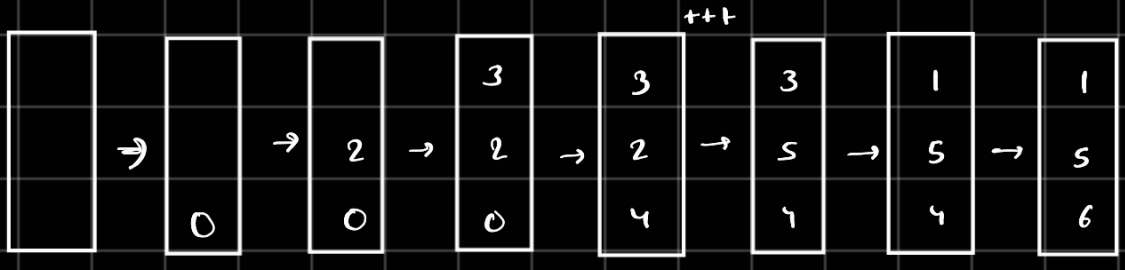
Virtual Memory →

System Crash →
When CP perform swap in / swap out

① FIFO →

hit → +
fault → −

P_1	0	2	3	4	3	2	4	5	1	6
	−	−	−	−	+	+	+	−	−	−



Swap space in disk

Virtual Memory = Swap space + RAM

②

Page	7	0	1	2	0	3	0	4	2	3	0	3	2	3
	−	−	−	−	+	−	+	−	+	+	−	+	+	+

frame size 4

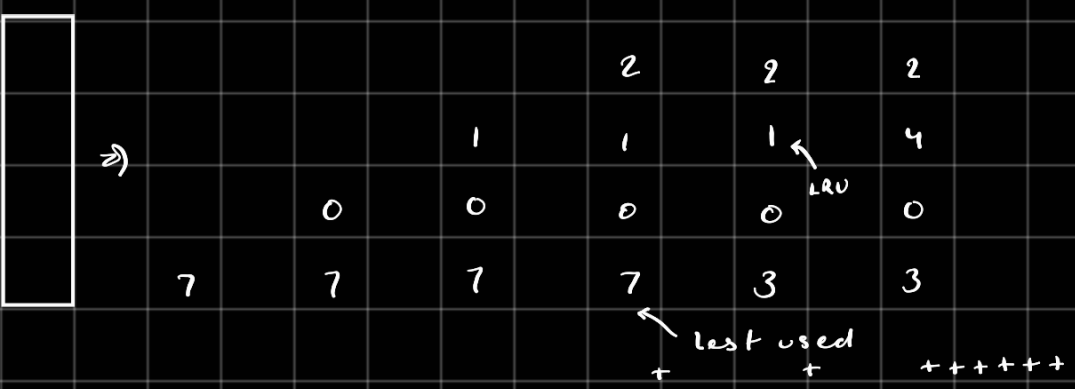
Hit → 7
Total → 14

$\frac{7}{14} \times 100 \Rightarrow$ 50% hit and 50% miss

Least Recently Used →

Page 7 0 1 2 0 3 0 4 2 3 0 3 2 3
- - - - + - + - + + - + + +

frame size 4



$\frac{8}{100} \times 50$

1.1 × 50

⇒ 55 % hit

45 % miss

Optimal →

Page 7 0 1 2 0 (3) 0 4 2 3 0 3 2 3

frame size 4

Logical Address → Page Table

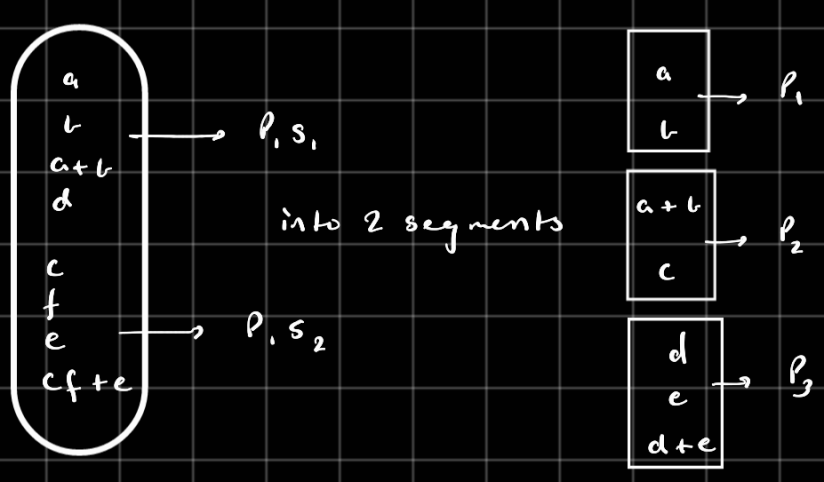
↓

Physical Address

Segmentation →

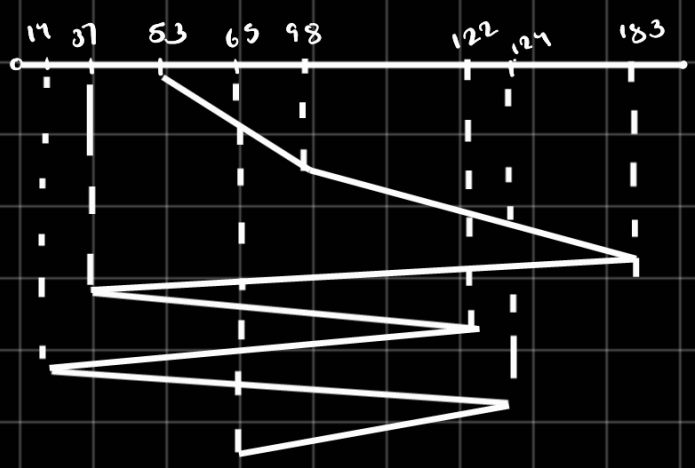
dividing the process into segments, such that program

Process →



Disk Management →

98, 183, 37, 122, 14, 124, 65, 67

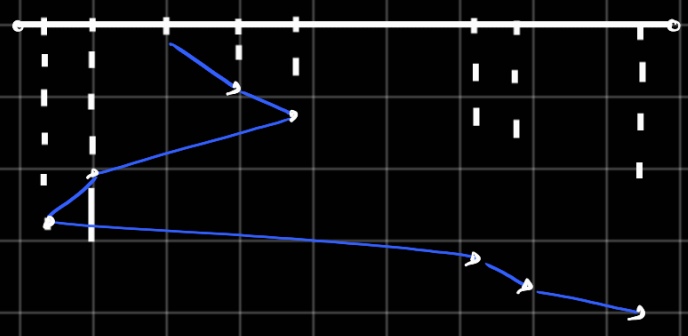


① FCFS

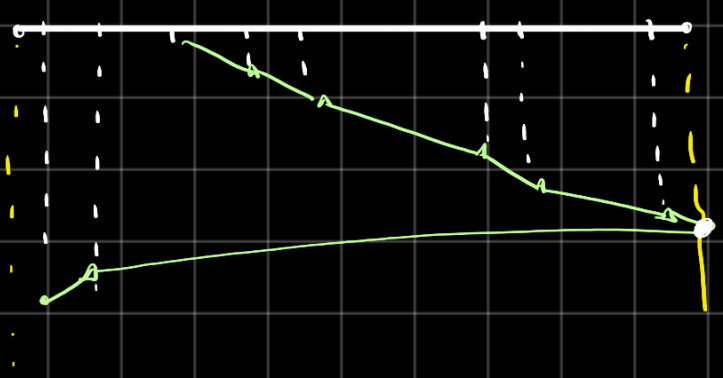
Track moment so high

(i) (98-53) + (183-98) + ...

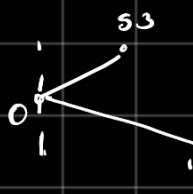
Track Moment



② SSTF (Shortest Seek Time First)

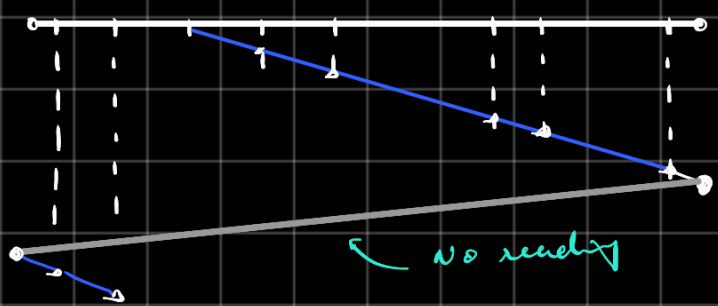


③ SCAN



Not zig-zag ≥

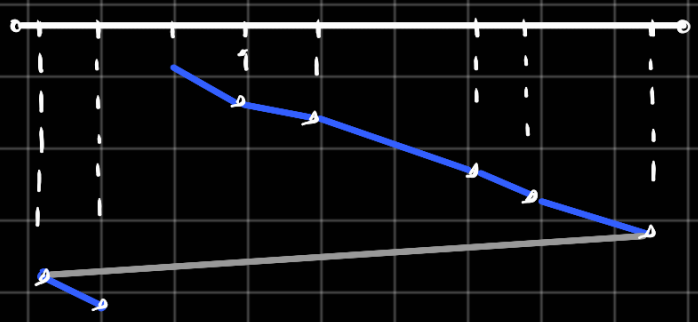
either forward or backward till not finished is one way.



④ C-SCAN

won't scan while returning back.

- Gets till the end.

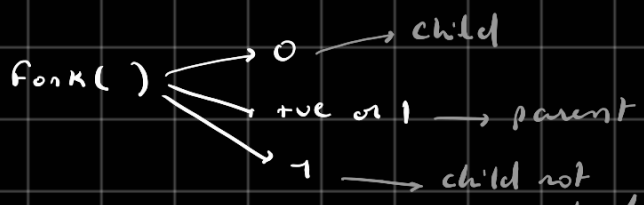


(S) C-look

- Only till required region.

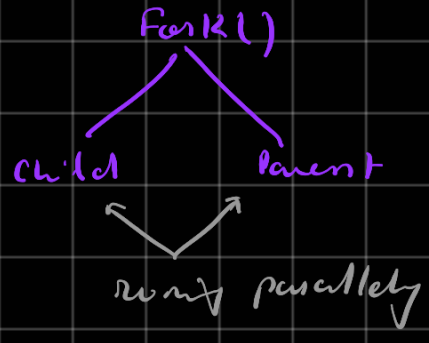
fork() system call →

To create a child process.




```
main() {
  fork();
  printf("Hello World");
}
```

→ Hello WorldHello World



Hacker →

- Hacker is one with good knowledge of Operating system and programming language.
- Hacks into system for the sake of finding loop holes in the system.

Cracker →

- Crackers also has good knowledge of Operating system and programming lang.
- He breaks into the system for illegal activity reasons or for personal gain.

Raid →

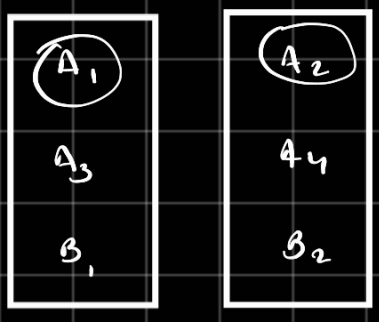
Redundant Array of Independent Disk.

Duplicate

① Performance
② Security
③ Available 24x7

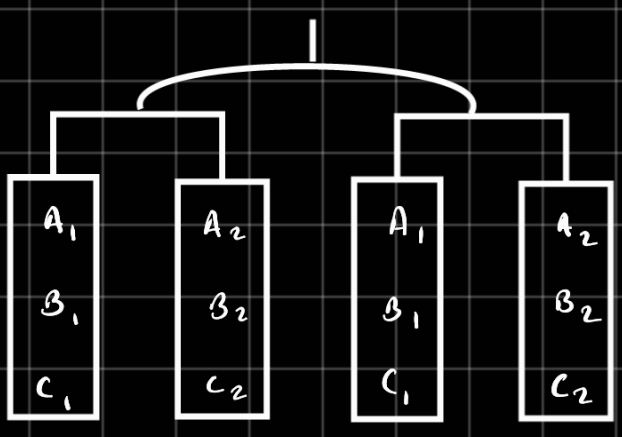
- Stripping

Raid '0' → Task → A → A_1, A_2; B; C



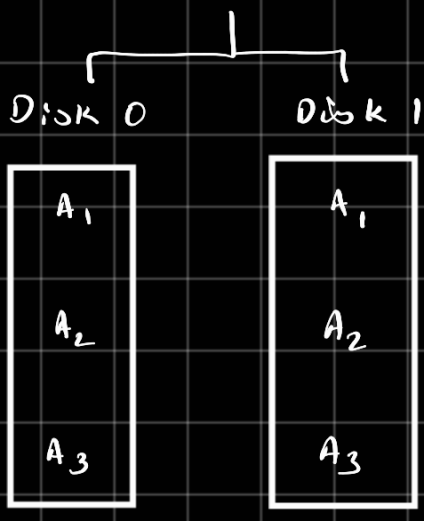
Data A can be accessed at the same time.

Raid '1+0'

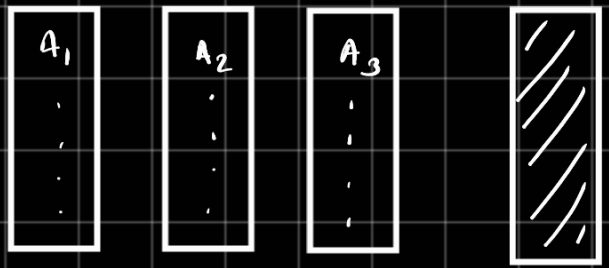


Raid "1+0"

Raid '1' → Data Mirroring



Raid '3' →



Parity Bit

Raid '4'

multiple parity

Raid '5' ⟶

A_1	C_1	P_3
P_1	A_3	C_3
B_1	P_2	B_2
A_2	C_2	B_3

- only one parity bit in one line.

P — parity Bit

distribution parity

Unit - 1

- Define OS and its types?
- function of the operating system
- Symple and layer structured of OS.
- CPU scheduling "SRJF, Round Robin" ① Question
- Deadlock and its conditions
- Deadlock handling techniques.
- Numerical based on Bankers algorithm

Unit - 2

- Difference between internal, external fragmentation
- Paging with example / diagram.
- segmentation with diagram.
- Page fault, 1 numerical
- Disk scheduling, 1 numerical

Unit - 3

Domain Structure →

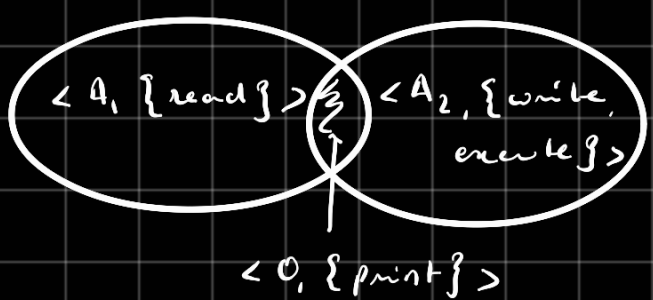
Access Rights →

< object name, rights >

Domain →

set of access rights.

< O_1, {read} >
< O_2, {write} >
< O_3, {read, write} >
< O_4, {execution} >



Access Matrix →

1. Global table →

stores tuples < domain, object name, rights >

Domain \ Object	F_1	F_2	F_3	F_4
D_1	r/w		r	
D_2		r		
D_3	r			r/w

2. Attribute list for objects →

F_1 → r/w
F_2 → r
F_3 → r
F_4 → r/w

3. Capability list for domain →

$D_1 \rightarrow$ r/w

$D_2 \rightarrow$ r

$D_3 \rightarrow$ r/w

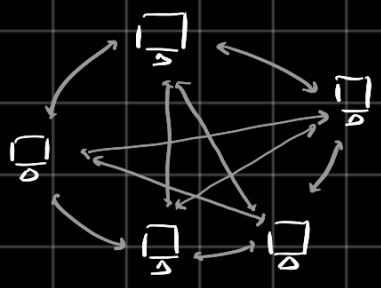
4. Lock - Key Mechanism →

combination of Attribute list and Capability list.

Object / domain } has unique bit pattern known as → lock, → key

Networking Architecture →

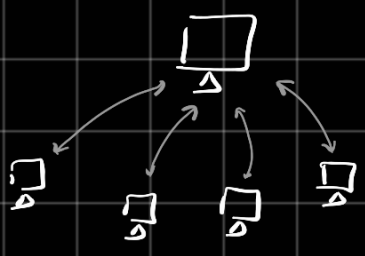
1. Peer to Peer →



- All system's are connected to each other
- Every system has equal rights.
- If a system goes offline, no effect on other system.
- Cost effective

- No data backup
- Less secure
- Can't handle multiple devices, as the system increases it becomes slower.

2. Client - Server



- There is mainly only one server, which manages all the important tasks.
- Data backup can be done.
- Client can ask for service and it's upto server to solve or discard the request.

• If server goes down then no function can be performed.

Network Topologies →

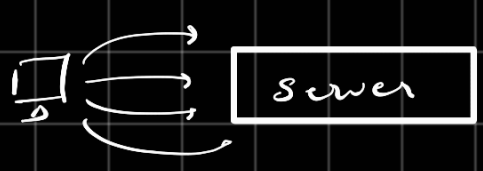
• Bus • Ring • Star • Mesh • Hybrid

Network and denial of service attack →

1. Denial of service Attack → DOS

Attack is an attempt to make network or a server resource unavailable to user.

- TCP-SYN flooding
- PING of DEATH

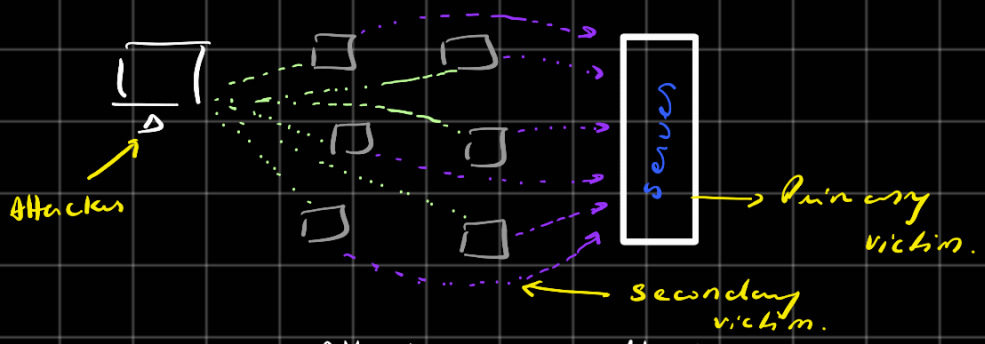


- Multiple attacks from single PC (normally)

2. Distributed Denial of service Attack → DDOS

Attack is an attempt to make network or services unavailable for users.

• The attacks are send from numerous hacked/zombie systems.



• Attacks from multiple zombie system's

- Use of unsecure internet channels
- Huge traffic
- Hide attacker's Id
- Security defence of victim

Plain Text →

A simple message or information which makes sence to a normal human.

Cipher Text →

It encreaps a sencesable message/information into to secure format such that it doesn't makes sence to a normal human being.

"Hello"

→ "elolh" → ciphers

Cryptography →

Plain text → [Encryption] —Ciphers→ [Decryption] → Plain text

Threats →

1. Program Threat →

When a program created by a user is used by another user then misuse of the the program may occur.

Prog. created by 'A' → X → B using Program 'X' created by 'A'

eg → Trojan Horse (wiki is wooden horse example).

2. System Threat →

OS files or resources being misused.

- Worm
  - Duplication
  - Malacious files cases may cause damage.
  - Consumes a lot of resource, which causes denying of services to users.

- Viruses →
  - Duplicates
  - can execute without user knowing about it.
  - stages →
    1. Sitting
    2. infecting host files
    3. triggering
    4. damage