Page No.
Date: / /

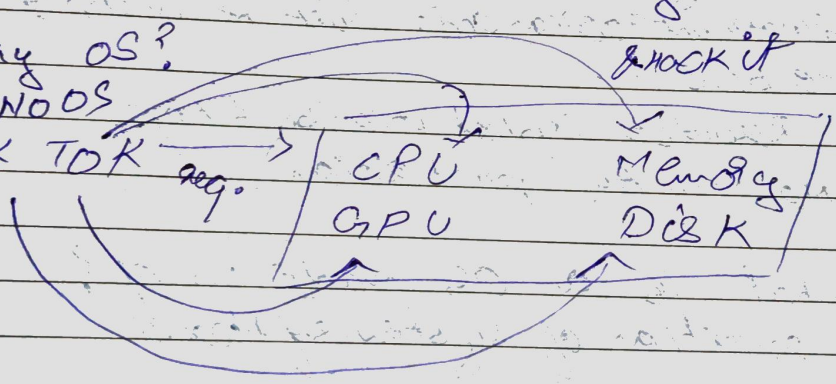
OPERATING SYSTEM

LB

Lec1 : what is operating system.

- Application software performs specific task for the user System software operates and controls the computer system and provides a platform to run application software.

- An operating system is a piece of software that manages all the resources of a computer system, both hardware and software, and provides an environment in which the user can execute his/her programs in a convenient and efficient manner by hiding underlying complexity of the hardware and acting as a resource manager.



Now TikTok is running, if now can we use another app then it won't even open as CPU, GPU, memory, Disk all are locked already.

→ To solve this we can give some % resources

- Add OS layer

Tik tok → 5% CPU
10% memory
12% GPU
& other apps bhi apni jagah.

Pubg → 50% CPU
30% memory
60% GPU.
Resource management.

* Why OS?
* What if there is no OS?
  - Bulky and complex app (hardware interaction code must be in app's code base)
  - Resource exploitation by 1 App.
  - No memory protection.

* What is an OS made up of?
  - collection of system software

* An OS function -
  - Access to the computer hardware
  - interface b/w the user & the computer hardware.

- Resource management (AKA, Abitration) (memory, device, file, security, process etc)

- Hides the underlying complexity of the hardware (AKA, abstraction)

- facilitates execution of application programs by providing isolation & protection.

- OS acts as interface.

- If there is no OS.
  In C++ ~~memory allocation~~ → malloc

TikTok
↳ Developer has to write app code & memory management logic bhi khud likhna hoga

Pubg
↳ memo mang.
↳ res. mang.

Ab dekho ek game code use ho raha Pubg wale or tik tok or extra codes bhi ek hi app hogya bulky
⇒ App → 50 MB + 700 MB → bulky ho.

This violates DRY principle

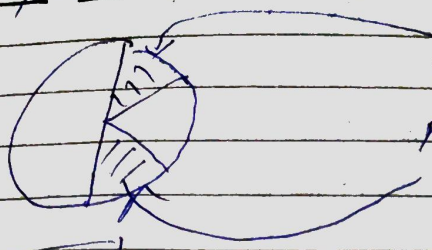
DRY: Do not repeat yourself

→ Resource management code

OS

resource res. mang. code OS apne wale lega

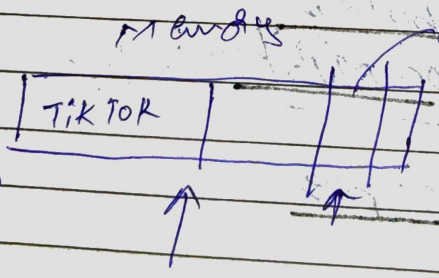
- OS provide isolation & protection.



If no OS
abhi the memory alag dikh rahi h but ye necessary nhi h
OS nhi h => isolation nhi h
But
Inko dono ki memory alag alag rakhi chahiye.

Agar OS nhi h to Sktah
Tik tok jake Pubg ke memory wale override krde
This is Security violation

If there is OS then



OS ensure both memory should be diff.

App prog.
OS
Computer hardware

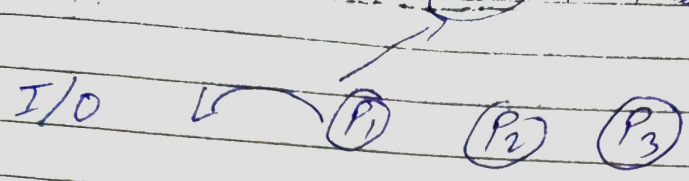
OS provides the means for proper use of the resources in the operation of computer system

LEC-2: Types of OS

OS goals

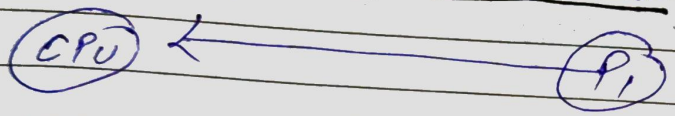
1. Maximum CPU utilization

Hum yhi chahte ki CPU kbhi ideal na ho



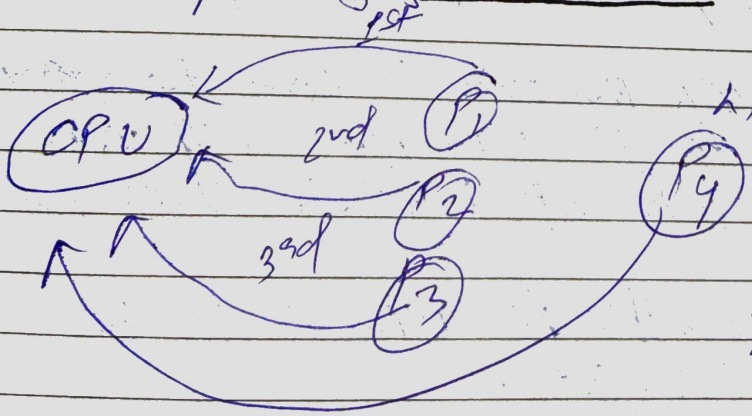
P_1 → CPU pr P_1 I/O wae gya th CPU ideal na ho th hm chahenge P_2 ko execute krle.

2. Less process starvation



Agr P_1 bhut time P_2 ke bhad th hm yhi P_3 chahenge ki baki process ko wait kr na mile.

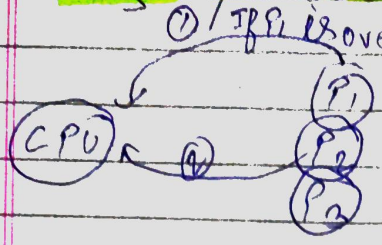
3. Higher priority job execution.



High priority then all other process dis continue & P_4 will execute

Types of OS

(1) Single process OS



⇒ No max. CPU utilization
i.e if $P_1 \rightarrow I/O$ th P_2 KO
mauka bhi nhi milega

only 1 process executes at a time from the ready queue.

⇒ process starvation

bhut time leta h then again P_2, P_3 KO mauka nhi milega

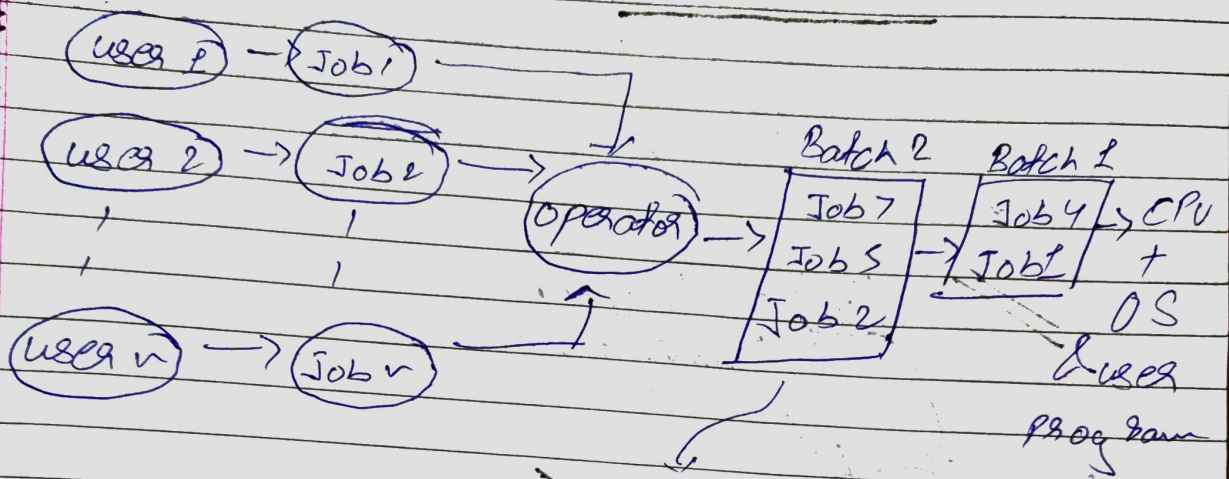
⇒ No high priority job execution.

Ex - MS DOS.

(2) Batch processing OS.

- Firstly user prepares his job using Punch cards.
- Then he submits the job to Computer operator
- operator collects the jobs from diff users & sort the jobs into batches with similar needs.
- Then operator submits the batches to the processor one by one.
- All the jobs of one batch are executed together.

- All the jobs of one batch are executed together.
  - Priorities cannot be set. of a job with some higher priority.
  - May lead to starvation. (A batch take more time to complete).
  - CPU may become idle in case of I/O operation.



& these batch ones a single process

Batch → OS → CPU

Job 1 then Job 4

=> No max. CPU utilization

- Wahi same reason agar koi batch bhut time lga diya

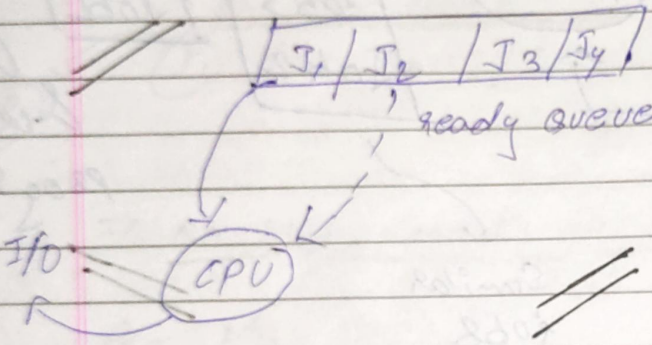
=> ~~XXX~~ process starvation & batch starvation

=> ~~batch~~ high priority jobs X

(3) Multiprogramming

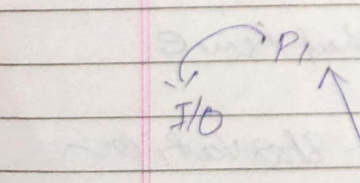
Increases CPU utilization by keeping multiple jobs (code & data) in the memory so that CPU always has one to execute in case some job gets busy with I/O.

- Single CPU
- Context switching for processes.
- switch happens when current process goes to wait state.
- CPU idle time reduced.



Jb bhi I/O k CPU, I2 le lega or I1 wait state mae chla jaega.

- context switching.



P1 jb I/O ke liye jaegi, P2 ka execution mae jana hoga. usi ko bol te h context switching

Jb bhi P1 wait state mae jaegi th P1 ka jo abhi ka context hoga usko hm save krlenge i.e abhi ki state (P1 konse address pe thi) inum sb ko PCB mae save krlenge.

small → OS

- PCB : store process ki info
(PROCESS Control block)

ab jo (P2) ki info h PCB se CPU le lega.

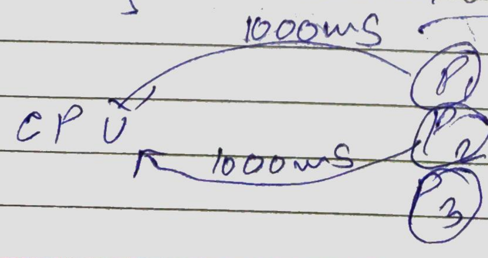
(4) **Multitasking OS.**

- more than 1 CPU on a single computer.
- increase reliability, 1 CPU fails other can work.
- better through put.
- less process starvation 'if 1 CPU is working on some process other can be executed on other CPU.

- Single CPU
- Able to run more than one task simultaneous
- Context switching and time sharing used.
- Increases responsiveness.
- CPU idle time is further reduced.

If one process take alot of time th isse overcome karne ke liye, th isko I/O ke liye wait nhi karte, hm karte h ek **time quantum** le lete h

TQ → 1000 ms
Agar koi job 1000ms ko mil jaenge th wo uski ek quantum of execution hogya h ab dusri job ko mauka do.



→ 1000 ms complete then next process.

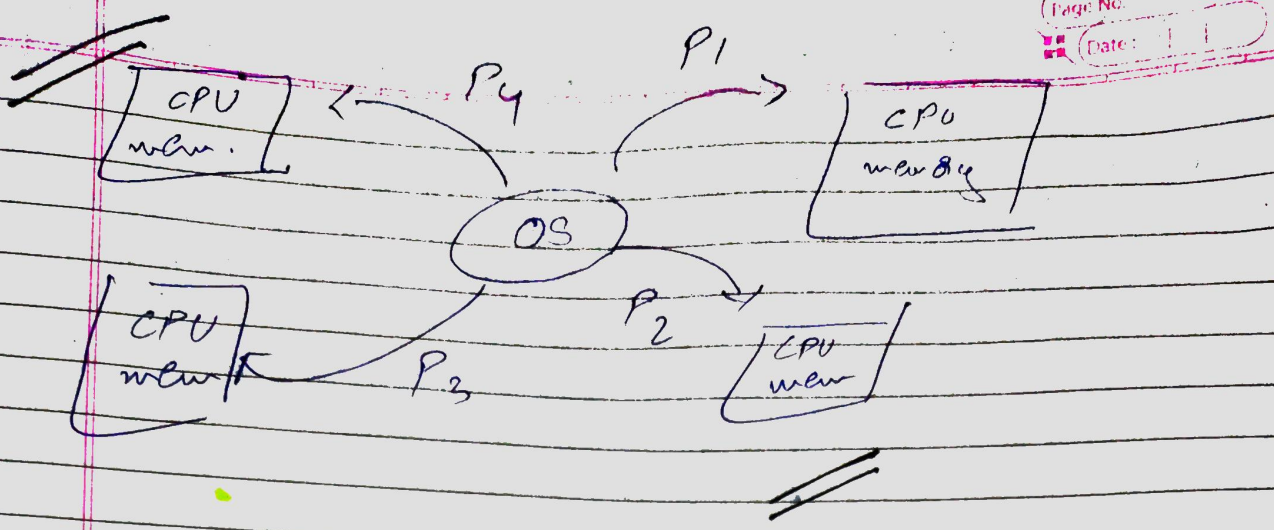
i.e time sharing.

=> CPU utilization ↑↑
=> process starvation XX

=> context switching is invisible then if any high priority job come then it will be executed immediately.

(5) Multiprocessing OS
- more than 1 CPU in a single computer.
- increases reliability, 1 CPU fails other can work.
- Better throughput
- Lesser process starvation - if 1 CPU is working on some process, other can be executed on other CPU.

(6) Distributed OS
- OS manages many bunch of resources.
  >= 1 CPUs, >= 1 memory, >= 1 GPU, etc.
- Loosely connected autonomous, interconnected computer nodes.
- Collection of independent, networked, communicating and physically sep. computational nodes.



(7) RTOS :- Real time OS
- error chance XX

- Ex- Air traffic control system, Robots, Nuclear plants,

LEC 3 | Multi-Tasking vs Multithreading.

* Program: A program is an executable file which contains a certain set of instructions written to complete the specific job & operation on your computer.
  - It's a complied code. Ready to be executed.
  - stored in Disk.

Process: program under execution. Resides in computer's primary memory (RAM).

Code → C++.

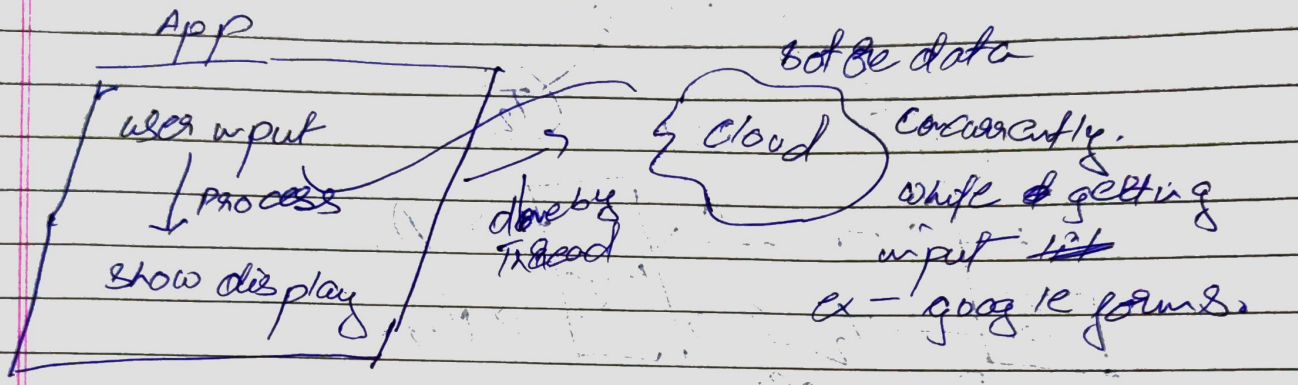
.txt → .cpp → Compile → executable (.exe) (or .x) in window

Executable is OS dependent
mac mac alag, windows mac alag.

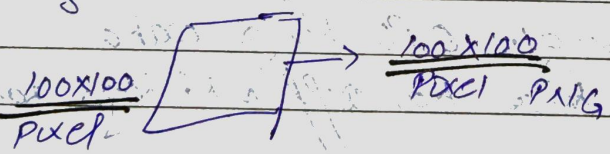
(.exe) → double click becomes Process

* Thread:

- Single sequence stream within a process.
- An independent path of execution in a process.
- Light weight process.
- used to achieve parallelism by dividing a process's task which are independent path of execution.
- Eg.- Multiple tabs in a browser, text editor (when you are typing in an editor, spell-checking, formatting of text & saving the text are done concurrently by multiple threads.)



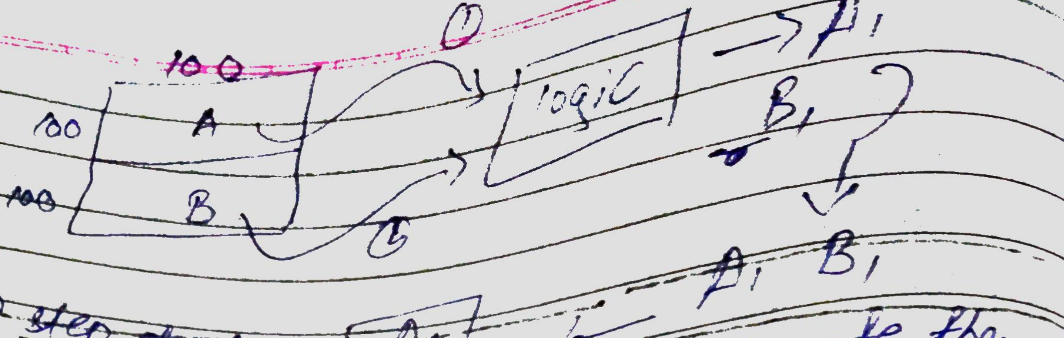
ex- JPG → PNG



→ Input → Image file 100 X 200
width height

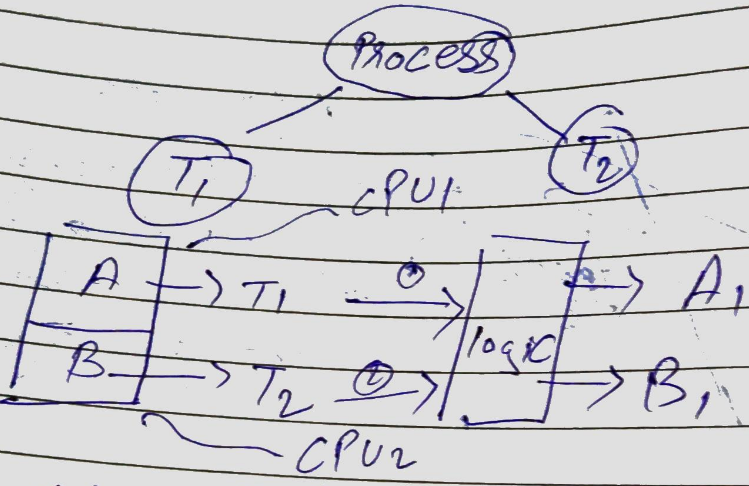
But limitation is 100X100
but iss kaam ko kaise karna

① Sequentially:



1 & 2 step alag ho raha h th time lel IOS + IOS hoga

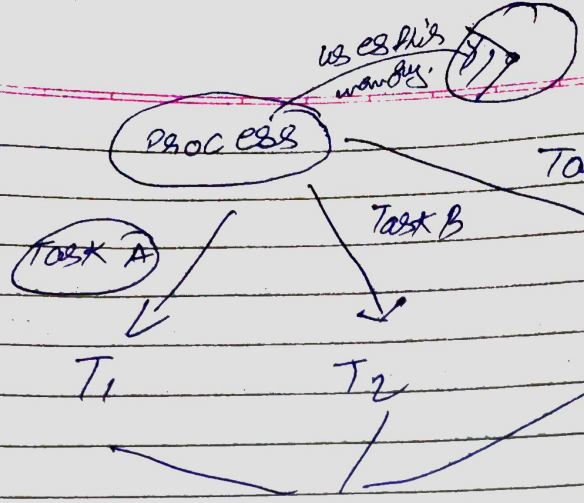
2



Yaha 1 & 2 dono ek saath hoga. 11aly th IOS mae hojaega.

Jaha pe single CPU hoga waha multithreading mae koi diff nhi hoga same reh jaega

Ye tech waha hoga jaha multi processing environment hoga.



uses the same memory as process

No isolation in threads.

Multi tasking	Multi threading
(1) The execution of more than one task simultaneously	(1) A process is divided into several different sub tasks called as threads.
(2) Concept of more than 1 process being context switched.	(2) Concept of more than 1 thread. Threads are context switched
(3) No. of CPU = 1	(3) No. of CPU >= 1 (Cores have more than 1)
(4) Isolation & memory protection exists. OS must allocate separate memory and resources to each program that CPU is executing	(4) No isolation & memory ~~allocation~~ protection, resources are shared among threads of that process. OS allocates memory to a process; multiple threads of that process share the same memory.

* Thread Scheduling

Threads are scheduled for execution based on their priority. Even though threads are executing within the runtime, all threads are assigned processor time slices by OS.

* Thread context switching	Process context switching
- OS saves current state of thread & switches to another thread of same process.	- OS saves current state of process & switches to another process by restoring it's state
- Doesn't include switching of memory address space. (but program counter, registers & stack are included)	- Includes switching of memory address.
- Fast switching	- Slow switching
- CPU's cache state is preserved	- CPU's cache state is flushed.

LEC 4 | Components of operating system

* Components of OS

1. User Space

where application software runs, apps don't have privileged access to underlying hardware. It interacts with kernel.

(a) GUI
Graphical user interface
- just the interface

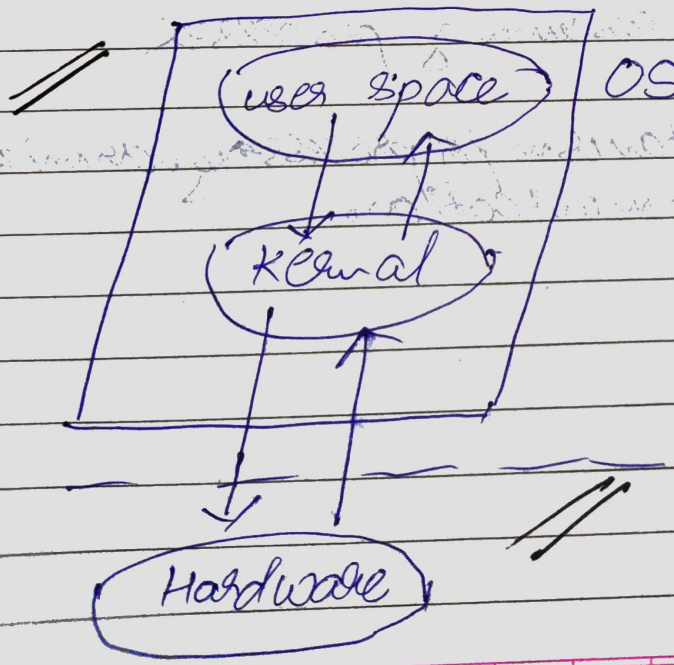
(b) CLI
Command Line interface
- writes command here

2. Kernal

Kernal is that part of OS which interacts directly with the hardware and performs most crucial tasks.

(a) Heart of OS / core component
(b) very first part of OS to load on startup.

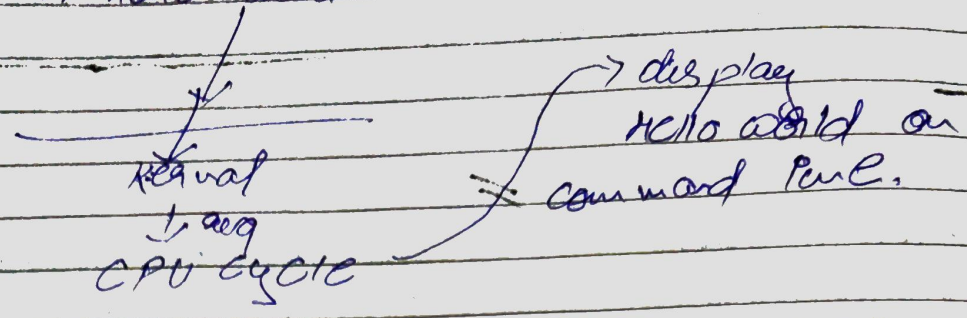
- Interacts with hardware



ex-

Helloworld.sh/-

./ Helloworld.sh



A shell, also known as a command interpreter, is that part of OS that receives commands from user & gets them executed.

Functions of Kernel:

* Process management.
  - scheduling process & threads on the CPUs
  - creating & deleting both user & system process.
  - suspending & resuming processes.
  - providing mechanism for process synchronization & process communication.

* Memory management
- Allocating & deallocating memory space as per need.
- Keeping track of which part of memory are currently being used and by which process.

* File management
- Creating & deleting files & directories to organize files
- Mapping files into secondary storage
- Back up support on a stable storage media.

* I/O management
- To manage & control I/O operation & I/O devices.
(a) Buffering (data copy b/w two devices), Caching & Spooling.
(i) Spooling → within differing speed two jobs
→ Eg print spooling, mail spooling.

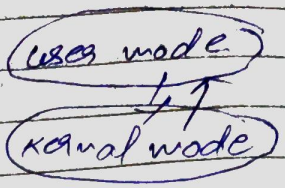
(ii) Buffering → within one job
→ Eg yt video buffering

(iii) Caching → Memory caching, web caching etc.

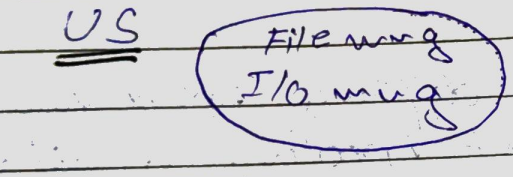
Types of Kernels:

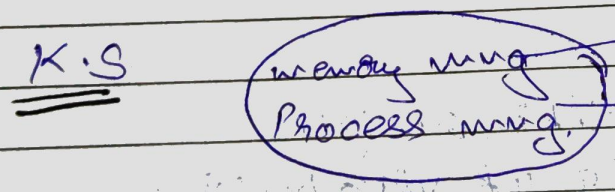
Monolithic Kernel

- All functions are in kernel itself.
- Bulky in size.
- Memory req. to run is high
- Less reliable, one module crashes
  ↳ whole kernel is down
- High performance as communication is fast (less user mode, kernel mode over heads)
- Ex- Linux, Unix, MS DOS.



Micro Kernel

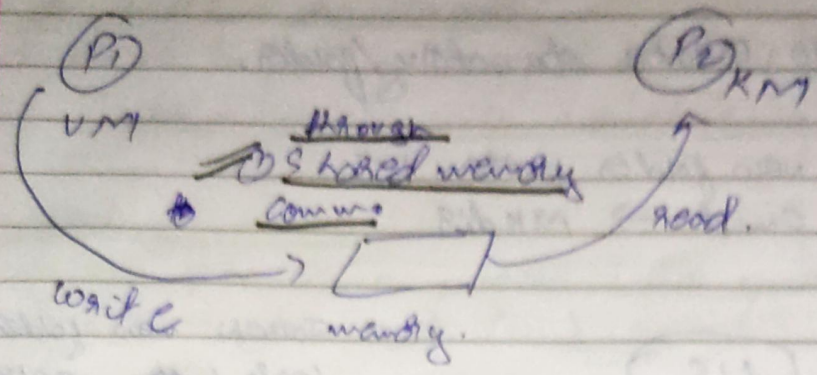




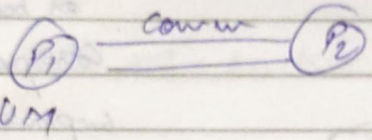
- Smaller in size
- More reliable
- More stable
- Performance is slow
- overhead switching b/co user mode & Kernel mode.
- Ex- L4 Linux, Symbian OS, MINIX etc.

Q. User mode & Kernel mode how they communicate.

→ IPC → Inter process communication.



② Message passing



* Hybrid Kernel:
- Advantages of both worlds (File mgt in US & rest in KS)
- Combined approach.
- Speed & design of mono.
- modularity & stability of micro.

- Ex - MacOS, Windows NT/7/10.

- IPC also happens but lesser overheads.

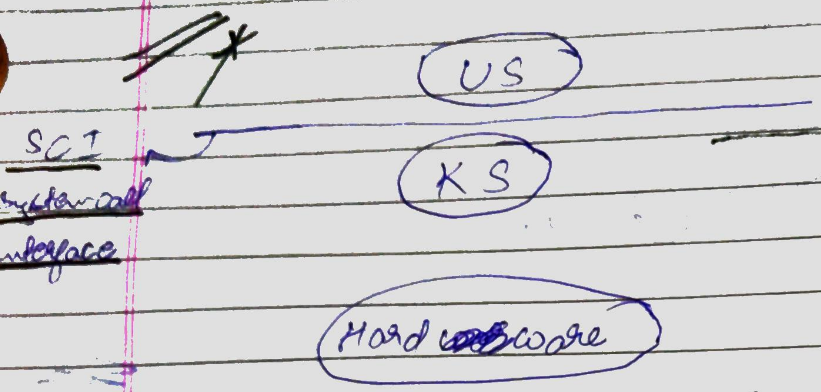
* Nano / EXO Kernel

LEC-51 System Calls in OS.

* How do apps interacts with kernel? → using system calls.

* ex – Aim to create directory/folder.

GUI → new folder button ✓
CLI → cmd → mkdir



Jab bhi new folder wale button pe click kiya tb US → SCI or bolega mkdir ke corresponding jo bhi implementation h kernel pe uspe mujhe le jao.

mkdir ke corresponding KS mae implementation hogi { = }

SCI bolega uss implementation ko find karo find karne ke baad KS pe run kardo run karne pe uska tree hoga are node add karega. ab mobile node movie folder dikhega.

Actual implementation mkdir → C.

⇒ system calls → C

In NUTSHELL

— MK dir 1&2.

- mkdir directly calls Kernel & asked the file mgmt. module to create a new directory.
- MKdir is just a wrapper of actual system calls.
- MKdir interacts with Kernel using system calls

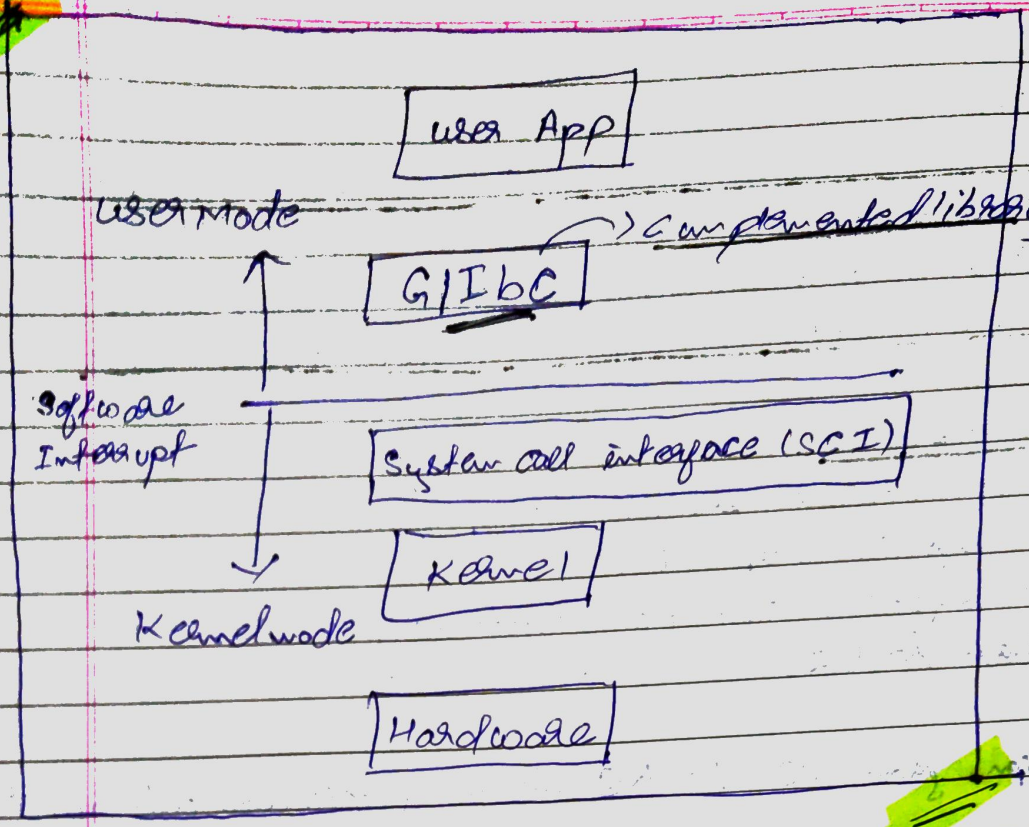
— Creating a process

- user executes a process. (user space)
- Gets system call. (US)
- exec system call to create a process. (KS)
- Return to US.

Transition from US to KS done by software interrupts.

- A system call is a mechanism using which a user program can request a service from Kernel for which it does not have the permission to perform.
  user programs typically do not have permission to perform operations like accessing I/O devices & communicating other programs.

System calls are the only way through which a process can go into Kernel mode from user mode.



* Types of System calls

1) Process Control
   - end, abort
   - load, execute
   - create process, terminate process
   - get process attributes, set process attributes
   - wait for time
   - wait event, signal event
   - allocate & free memory.

Windows	Unix
CreateProcess()	fork()
ExitProcess()	exit()
WaitForSingleObject()	wait()

2) File management

- Create file, delete file.
- Open, close
- Read, write, reposition.
- get file attributes, set file attributes.

Windows	Unix
Creat file ()	open ()
Read file ()	read ()
write file ()	write ()
close handle ()	close ()
Set file security ()	Ch mod ()
Initialize security descriptor ()	Umask ()
set security descriptor Group ()	Ch own ()

3) Device management

- Request device, release device
- read, write, reposition.
- get device attributes, set device attributes
- logically attach or detach devices.

Windows	Unix
set console Mode ()	ioctl ()
Read Console ()	read ()
Write Console ()	write ()

4) Information management

- get time & date, set time & date
- get system data, set system data
- get process, file & device attributes.
- set process, file & device attributes.

Windows	Unix
Get Current Process ID()	get pid()
Set Timer()	alarm()
Sleep()	sleep()

5) Communication Management

- Create, delete comm. connection
- send, receive messages
- transfer status information.
- attach & detach remote devices.

windows	unix
create pipe()	pipe()
create file mapping()	shmget()
Map view of file()	mmap()

fork → kissi bhi process ki child banati h

terminal → run exe

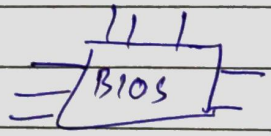
Parent process → child process

Lec-6 | what happens when you turn on computer?

I. PC on

II. CPU initializes itself & looks for a firmware program (BIOS) stored in BIOS chip (Basic input output system chip is a ROM chip found on motherboard that allows to access & setup computer system at most basic level.)
- In modern PCs, CPU loads UEFI (unified extensible firmware interface).

CPU loads → BIOS & UEFI



- CPU goes to chip BIOS
- BIOS & UEFI run test & initi. hardware.
  - BIOS loads some setting from a memory area
    backed by CMOS battery
- Program loads with setting.
- load ke baad pata karenge power on self test ki essential hardware hai ya nhi

(iii) CPU runs BIOS which test & initializes system hardware. BIOS loads configuration settings. If something is not appropriate (like missing RAM) error is thrown & boot process is stopped.

This is called POST (Power on self test) process.

(UEFI can do a lot more than just initialize hardware; it's really a tiny OS. ex - Intel CPUs have Intel Management Engine. This provides a variety of features, including powering Intel's active management technology, which allows for remote management of business PCs)

iv) BIOS will hand off responsibility for booting your PC to your OS's boot loader.

— BIOS looked at MBR (master boot record), a special boot sector at the beginning of a disk. The MBR contains code that loads the rest of OS, known as boot loader.

- The BIOS executes the boot loader, which takes it from there & begins booting the actual OS windows & Linux,
ex - In other words,
The BIOS & UEFI examine a storage device on your system to look for a small program either in MBR or on an EFI system partition & runs it.

BIOS hands off to boot device
boot device — disk (HDD or SSD
CD
USB.

Now OS need prog. to boot.
Boot loader → when executes → ON actual OS.
is where
BIOS → MBR → at disk's zeroth index.
used old system

UEFI → EFI (extensible format interface)
↓
Partition in disk
↓
~~this~~ boot loader.

V) The boot loader is small program that has a large task of booting the rest of the OS (boots Kernel then, user space). Windows uses boot loader named windows boot Manager (bootmgr.exe), most Linux system use GRUB & Macs use something called boot.efi

bootloader loads full OS

Lec-7 | Difference b/w 32 bit & 64 bit OS.

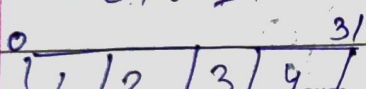
~~boot loader loads full OS~~

* 32 bit OS 32 bit registers & it can access 2^{32} unique memory address i.e 4GB of Physical memory.

* 64 bit OS has 64 bit registers & it can access 2^{64} unique memory addresses i.e 17, 179, 869, 184 GB of physical memory.

* 32 bit CPU architecture can process 32 bit of data & info
  64 bit CPU archit can process 64 bit of data & info.

CPU 1.

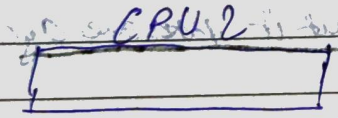


4 bytes registers → can hold 4 byte data at a time.

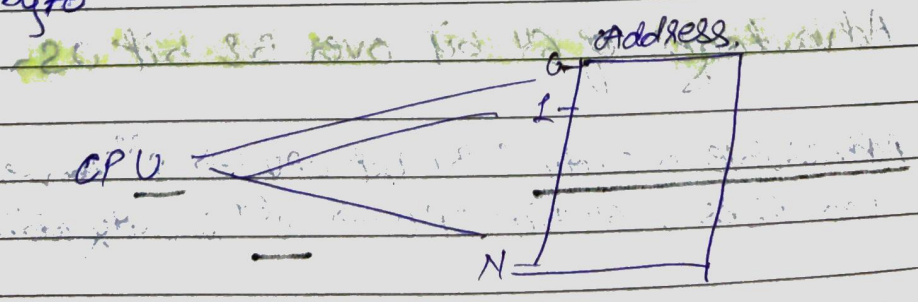
Reg. 1

00000000
8 bit

CPU 2



8 byte



32 bit CPU

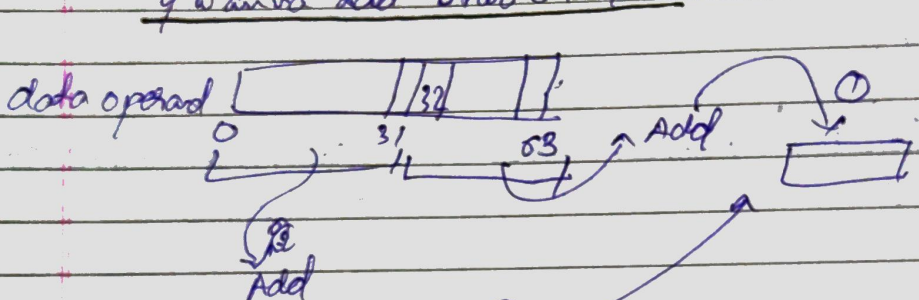
2^{32} → unique address

64 bit CPU.

2^{64} → unique addresses.

Reg. ka size double kr diya isse jada memory bhi ka sakte h, jad address ka bhi kr sakte h

32 bit → 32 bit data.
if wanna add two 32 bit no.
if wanna add two 64 bit no.



it takes 2 cycles for 32 bit OS but it take one cycle for 64 bit OS

* Advantages of 64 bit over 32 bit OS.

(a) Addressable memory : 32 bit CPU → 2^{32} memory addresses, 64 bit CPU → 2^{64} memory addresses.

(b) Resource usage : Installing more RAM on a system with a 32 bit OS doesn't impact performance. However, upgrade that system with excess RAM to 64 bit version of Windows & you'll notice a difference.

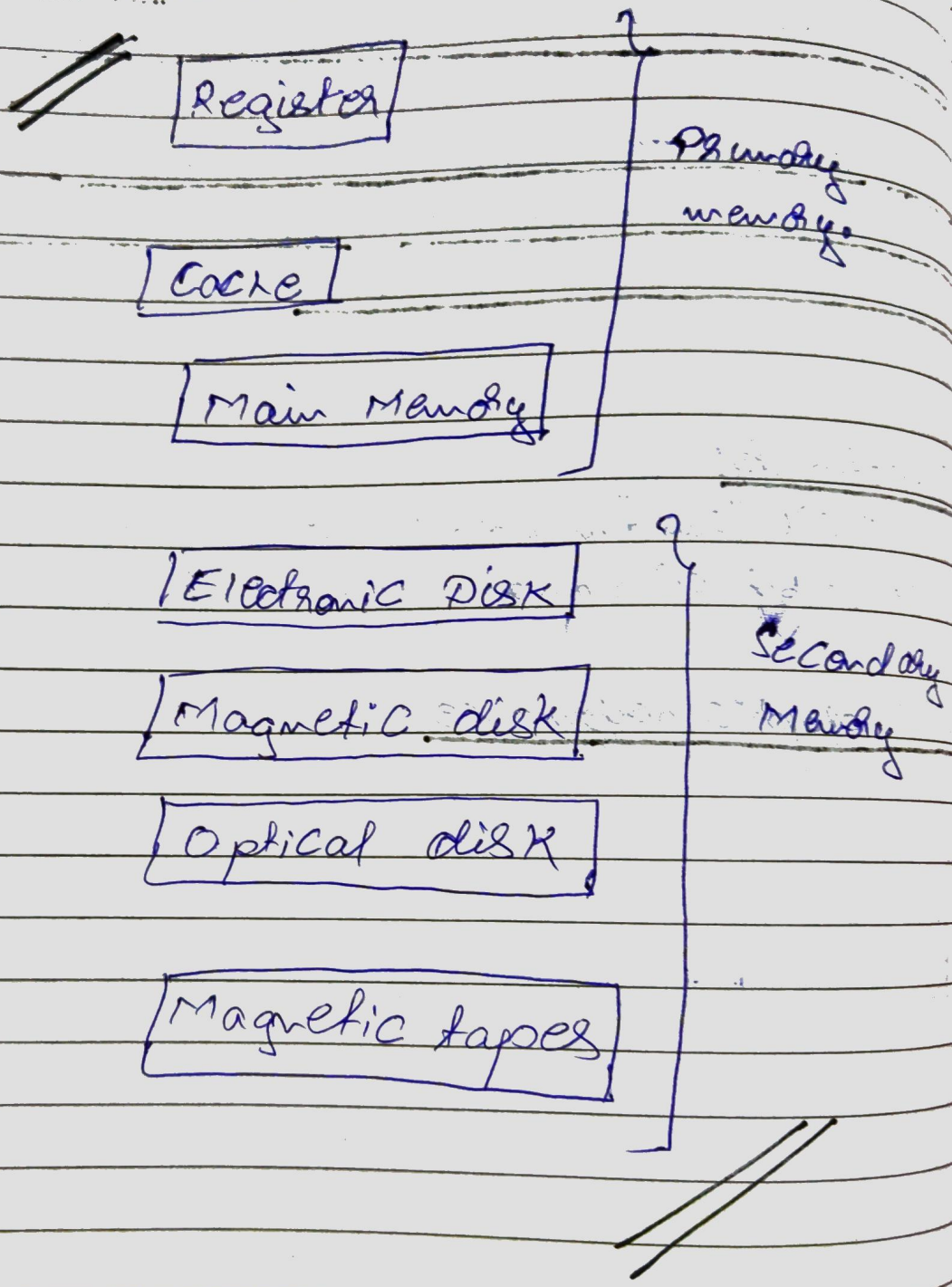
(c) Performance: All calculations take place in registers. When you're performing math in your code, operands are loaded from memory into registers. So having larger registers allow you to perform larger calculations at the same time.

32 bit processor can execute 4 byte of data in 1 instruction cycle while 64 bit means that processor can execute 8 bytes of data in one instruction cycle.

(d) Compatibility:
64 bit CPU can run both 32 bit & 64 bit OS.
While 32 bit CP can only run 32 bit OS.

(e) better graphic performance

Lec-81 Difference b/w different Storages used in computer.



* Register: smallest unit of storage. part of CPU.

*) AReg. may hold an instruction, a storage address, a any data(such as a bit seq. & individual characters).
Reg. are a type of computer memory used to quickly accept, store, transfer data & instructions that are being used immediately by the CPU.

* Cache:
Additional memory system that temporarily stores frequently used instruction & data for quicker processing by the CPU.

* Main memory: RAM

* Secondary memory: storage media, which computer can store data & prog.

* Comparision.
① Cost
ⓐ Primary storage are costly
ⓑ Reg. are most expensive due to expensive semiconductors & labour.
ⓒ secondary storage are cheaper than primary.

② Access speed.
ⓐ primary has higher access speed than secondary memory.
ⓑ Reg. are highest access speed. then comes cache then main memory.

(3) Storage size
   (a) secondary has more space.

(4) Volatility:
   (a) primary memory is volatile.
   (b) secondary is non volatile.