Operating System Concepts:

DAY-01:
Q. Why there is a need of an OS?
- as any user cannot interacts dierctly with any hardware device, hence there is need of some interface between user & hardware.

Q. What is an OS?
- An OS is a "system software" (i.e. collection of system programs) acts as an interface between user & hardware.
- It also acts as an interface between programs & hardware.
- As an OS allocates required resources like main memory, CPU time, IO devices access etc... for running programs, hence it is also called as a "resource allocator"
- As an OS manages limited available resources among all the running programs, it is also called as a "resource manager".
- an OS controls an execution of all programs as well it also controls hardware devices which are connected to the system, and hence an OS is also called as a "control program".

Q. What is an OS from end user point view:
- an OS is a software which comes either in a CD/DVD/PD.

- An OS is a software comes either in a CD/DVD/PD in a binary format, which mainly contains following three components:
1. "Kernel": kernel is a core part/program of an OS which runs continuosly into the main memory does basic minimal functionalities of an OS.
e.g. Linux: vmlinuz
Windows: win32krnl.exe

2. Utility softwares: these are softwares comes with an OS which provides extra utility functionalities
e.g. file manager, task manager, shell etc...

3. Application softwares:
e.g. google chrome, browser application, notepad, MS Office etc....

+ Functions of an OS:
core functionalities/basic minimal functionalities/compulsory functionalities
"kernel" functionalities of an OS:

1. Process Management
2. Memory Management
3. CPU Scheduling
4. Hardware Abstraction
5. File & IO Management

- extra utility functionalities/optional functionalities
6. Protection & Security
7. User Interfacing
8. Networking

$./program.out enter -> program gets loaded into the main memory
$.\a.exe -> enter ---> executable file gets loaded into the main memory from the hdd.

Q. What is a Program?
- set of instructions given to the machine to do specific task

Q. What is a Process?
- program in execution i.e. when a program gets loaded into the main memory it becomes a process.

+ "loader": it is system program (part of an OS), which loads an executable file from hdd into the main memory.
- to starts an execution of a program --> OS (loader)
- to allocate CPU time for a process --> OS
- to give control of an IO device --> OS

- for any program to completes its execution following resources are required:
    - main memory: RAM
    - CPU time: 1
    - input device access: KBD
    - output device access: Monitor
    etc...

+ "installation of OS": it is a process to store OS programs onto the hard disk drive.

+ "booting": it is a process in which bootstrap program locates a kernel which is onto the hard disk drive and load it into the main memory.

- bootstrap program is special program which is exists already into the hard disk drive in a boot sector (first sector of HDD).
- "bootstrap program" gets loaded into the main memory by "bootstrap loader".

- in a single memory cell: we can store either 0 Or 1 -> bit
- bit: binary digit
- 1 byte = 8 bits = 8 memory cells
- smallest addressable unit = 1 byte -> if processor want to read/write data into the main memory min 8 bits/memory cells must be selected.
- there are two methods by which processor can write data into the main memory:
1. "big endian format": in this method, processor writes data into the memory in its binary equivalent format as it is.

2. "little endian format": in this method, processor writes data into the memory in the reverse order of its binary equivalent format.

int num = 300;
sizeof(int): 4 bytes = 32 bits

00000000 00000000 00000001 00101100
MSB LSB

100101100

big endian format : 00000000 00000000 00000001 00101100

little endian format: 00101100 00000001 00000000 00000000

- "pentium family processor" use/follow little endian format.

16 bit compiler: sizeof(int): 2 bytes: 00000001 00101100
32 bit compiler: sizeof(int): 4 bytes: 00000000 00000000 00000001 00101100

300: 16 bit

- components of a computer hardware system are connected and does communications via a conducting wires referred as buses.
- 3 types of buses are there:
1. data bus/lines: carries data
2. addr bus/lines: addresses
3. control bus/lines: control signals
- control signals sent by the CPU to all devices are referred as a commands:
1. TEST: this command is used to check status of the device
2. READ:
3. WRITE:
4. CONTROL: to control operations on that device

- bus is a "shared communication pathway" i.e. data which is sent by one device can be recieved by any other device which is connected to the bus.

- bus which connects core components of a computer system is referred as a system bus.
- components of a computer system which are onto the motherboard are referred as core compeuter system.
e.g. processor, main memory, cache memory, ROM etc...
- devices which are connected to the motherboard externally through ports are referred as peripheral devices/peripherals.
e.g. io devices, hard disk drive --> external devices
==========
DAY-02:

+ Everyone must follow the standard protocols while lecture:
1. You can chat only with the host while lecture is going on, if you are having doubts ask to the faculty.
2. Post your doubts only after finishing the current topic based on current topics only, after asked to post doubts.
3. Every doubt/question will be first explained and then its answer
4. Last 20 mins are reserved for your other doubts, and if you want that time chat settings will be public.
5. Still your doubt is unanswered you can ask it onto the whatsapp group.

- "Computer Memory Heirarchy":
1. "CPU Registers": it contains currently executing data & instructions by the CPU temporarily.

    - MBR: Memory Buffer Register: main memory contents (data/instructions)
    - MAR: Memory Address Register: main memory addresses of the contents which are in MBR
    - IOBR: InputOutput Buffer Register
    - IOAR: InputOutput Address Register
    - PC: Program Counter: addr of next instruction to be executed
    etc....

2. One or more levels of cache memory i.e. L1 cache & L2 cache
3. Cache Memory
4. RAM Memory/Main Memory
5. Magnetic Disk Memory
6. Optical Disk Memory

Q. What do you mean by Internal & External memory?
- "Internal Memory": memory which is internal to the motherboard
    e.g. cpu registers, L1 cache & L2 cache, Cache Memory, RAM etc...

- "External Memory": memory which is external to the motherboard
    e.g. hard disk drive (magnetic disk memory), optical memory etc...

Q. What do you mean by Primary memory & Secondary memory?
- "Primary Memory": memory which can be accessible directly by the CPU
    e.g. cpu registers, L1 cache & L2 cache, Cache Memory, RAM etc...

- "Secondary Memory": memory which cannot be accessible directly by the CPU
    e.g. hard disk drive (magnetic disk memory), optical disk memory etc...

Q. Why RAM Memory is also called Main Memory?
- for an execution of any program RAM memory is must, and hence RAM memory is also called as main memory.
- RAM: Random Access Memory - volatile -> i.e. contents of this memory remains sustains only till power supply is there, when we switchedoff power supply main memory contents gets lossed.

Q. Why there is a need of cache memory?
- due to the speed mismatch between the CPU and main memory overall system performance gets decreased, and hence to reduce this speed mismatch cache memory can be added between the CPU & Main Memory.

Q. What is Cache Memory?
- cache memory is "faster" memory, which is type of RAM memory, in which most rencently accessed main memory contents can be kept in an "associative" manner i.e. in a key-value pairs.

- there are two types of RAM:
SRAM: Static RAM - memory cells are made up of capacitors --> e.g. main memory
DRAM: Dynamic RAM - memory cells are made up of flip-flop gates -> e.g. cache memory

- "word": it is a unit of memory from system point view.
- word length = 8 bits/16 bits/32 bits/64 bits
- on few processor, word lenght may vary or on few process it is fixed.

- "cache hit"
- "cache miss"
- one or more levels of cache memory i.e. L1 cache & L2 cache can be added between the CPU and Cache Memory, to reduce speed mismatch between them.

res = n1 + n2;

n1=10
n2=20

registers:
accumulator:

- the rate at which the CPU can execute instructions is much faster than the rate at which data can be fetched from the HDD, so there is a huge speed mismatch, and hence to reduce this speed mismatch disk chacke can be used.

- "disk cache": it is purely a software technique in which portion of the main memory can used as a cache memory in which most recently accessed disk contents can be kept in an associative manner.

- there are four methods by which data can be accessed from computer memory:
1. sequential access: e.g. magnetic tape - data used to stored in record
2. direct access: e.g. magnetic disk - hdd
3. random access: e.g. RAM Memory
4. associative access: e.g. cache memory

Book of => 1000 pages

access page no.: 755 -> sequential access

direct access:
Book of => 1000 pages => 5 blocks having same size
Block1: 1-200
Block2: 201-400
Block3: 401-600
Block4: 601-800
Block5: 801-1000

circular platter - book of 1000 pages
hundreds of concetric rings: 200 = tracks --> 5 blocks
each track is divided into thousands equal size of blocks : 1000 sectors -> each block 200 pages

usually the size of sector = 512 bytes

- if we want to access (sector no.: 455 + track no.: 125)
- this request will be sent to the "disk controller", and disk controller controls movement the "head"
- "head" is a conducting coil exists in hdd which can be used to access data from one sector at a time i.e. head has the capacity to write/to read data into/from the sector at a time.
- head can access 512 bytes of data at once - block by block
- data access can be done block by block - hdd is also called as a "block device".

- seek time: time required for the disk controller to move head from its current position to the desired track is referred "seek time".

- time required for rotation of a platter till desired sector will not gets aligned with the head is referrred as "rotational latency".
- access time = seek time + rotational latency.

+ "IO Devices/External Device/Peripheral Devices":

Core Computer System: CPU, RAM, Cache Memory etc...
Peripheral Devices   : HDD, IO devices

Input Devices: KBD, joystick, bar code reader, scanner, webcam, mic, etc...
Output Devices: Monitor, Printer, Plotter, Projector etc...

- whenever there is a transfer of data either from core computer system to an external devices or vice-versa it is referred as an "IO", and this io can be done via "io modules" i.e. through "io ports", as core computer system cannot directly interacts with any external device.
- io modules/io ports acts as an interface between core computer system and an external devices.
- structure of external device (input device/output device/memory device):
e.g. keyboard

- each external device has got three important blocks:
1. "control logic block": "controller" -> this block controls all the operations of that device.

2. "buffer": each device has got its own memory in which data can be stored temporarily

3. "transducer": this block is used to do communication with the outside world.
- this block converts one form energy into another form of energy.
- in i/p devices transducer convert any other form of energy an electrical energy,
whereas in an o/p device transducer converts an electrical energy into any other form of energy.

- "QWERTY" keyboard -> keyboard layout

- there are three io techniques:
1. "Program driven io":
- in this type of io all the steps/logic which is required for an io is there inside one program, and by means of executing that program by the CPU io can be done.
- in this method the CPU fully remains involved in an io, there is a less utilization of the CPU, because by the time an io modules does communication with io devices, CPU has to remain idle.

2. "Interrupt io":
Q. What is an interrupt?
- an interrupt is a "signal" which is sent by any io device to the CPU, due to which it stops an execution of one job/process and starts executing another job/process.
in other words:
- an interrupt is a signal received by the CPU from any io device, due to which it stops an execution of one job/process and starts executing another job/process.

- control signals sent by the CPU to an io devices are called as "commands"
e.g. TEST, READ, WRITE, CONTROL

- by the time an io modules does communication with an io devices, the CPU can complete part of another job/process, instead of remaining idle, and it can takes part again in an io by means of sending an interrupt, and hence due to an interrupt involvement of the CPU in an io has beed reduced and its utilzation gets maximized.

3. DMA (Direct Memory Access): this is an io technique in which DMA controller can be added onto the system bus which takes cares about an io between main memory and hard disk drive.

==========
DAY-03:
- UNIX: UNICS

- "File has Space and Process has Life".
- from UNIX point of veiw whatever that can be stored is considered as a file
and whatever which is in a running state is considered as a process.

- UNIX treats all devices as a file.
- UNIX system catagoriese devices into two catagories:
1. "character device": character special device files
- devices from which data gets transfered char by char
e.g. keyboard, monitor, printer etc...

"buffer": file in a main memory area in which temporarily data can be kept.

stdin : standard input buffer which is associated with standard i/p device - KBD
stdout : standard output buffer which is associated with standard o/p device - Monitor
stderr : standard error buffer which is associated with standard o/p device - Monitor

2. "block devices": block special device files
- devices from which data gets transfered block by block i.e. sector by sector

e.g. all storage devices

$./program.out - press enter --> to execute a program
- double click executable file

- Loader/OS creates a process implicitly
- to exit a process
- to open file a file or to create a new file

- sir can we create a process from out program explictly

* kernel: program -> functions

+ "system calls": are the functions (of kernel program) defined in C, C++ & assembly language which provides interface of services made available by the kernel for programmer user.
- In UNIX 64 system calls
- In Linux more 300 system calls
- In Windows more than 3000 system calls

- there are 6 catagories of system calls:

exit() --> _exit()
fopen() --> open()
fclose() --> close()
printf(), fprintf(), fputc(), fputs(), fwrite() --> write()
scanf(), fscanf(), fgetc(), fgets(), fread() --> read()
fseek() --> lseek()
etc...

//user defined code
#include<stdio.h>

//entry point function
int main(void)

```
{
	int n1, n2, res;

	printf("enter the values of n1 & n2: ");//write() - system defined code
	scanf("%d %d", &n1, &n2);//read() - system defined code

	res = n1 + n2;

	printf("res: %d\n", res);//write() - system defined code

	return 0;//successful termination
}
```

- whenever system call ocurres the CPU switched from user defined code to the system defined code, and hence system calls are also called as "software interrupts".

- throughout an execution of any program the CPU keeps switched in between user definded code and system defined code, and system runs in two modes
1. "kernel mode"/system mode/monitor mode/previledged mode: when the CPU executes system defined code instructions

2. "user mode"/previledged mode: when the CPU executes user defined code instructions

- there is one bit onto the CPU referred as a "mode bit" which is maintained by an OS/kernel.

mode bit = 0 --> kernel mode
mode bit = 1 --> user mode

- "dual mode operation" of a system.

==========

+ Process Management:

Q. What is Program:
- from OS point of view the program is nothing but an executable file (.exe, .out) which has got different sections like exe header, bss section, data section, rodata section, code/text section & symbol table.

- What Process?

+ features of an OS:

- "multi-programming": system in which more than one processes can be submitted at a time.
- "degree of multi-programming": no. of processes that can be submitted into the system at a time.

- "multi-tasking": system in which it seems that the CPU can execute multiple processes concurrently or simultaneously.

- "the CPU can execute only one process at a time".

Q. What is a thread?
- thread is the smallest execution unit of a process
- thread is the smallest indivisible part of a process
- thread is a lightweight process

- "multi-threading": system in which it seems that the CPU can execute multiple threads which are of either same process or different processes concurrently or simultaneously.

"the CPU can execute only one thread of any one process at a time".

- "multi-processor": system can run on such a machine in which more than one CPU's are connected in a closed circuit.
- "uni-processor"

- "multi-user": system in which it seems that the CPU can execute processes of more than one users concurrenlty.

- to keep control on all running programs kernel maintains few data structures:
1. "job queue": it contains list of PCB's of all submitted processes

2. "ready queue": it contains list of PCB's of proceses which are in the main memory and waiting for the CPU time.

- an OS maintains device queue/waiting queue per device
3. "waiting queue"/"device queue": it contains list of PCB's of processes which are requesting for that device.

- "job scheduler": it is a system program which schedules jobs/processes from job queue to load them onto the ready queue.

- "cpu scheduler": it is a system program which schedules job/process from ready queue to load it onto the CPU.

- "dispatcher": it is a system program which copies an execution context of process which is scheduled by cpu scheduler from its PCB into the CPU registers.

- in following four cases CPU scheduler must gets called:
1. running --> terminated
2. running --> waiting

3. runnig --> ready
4. waiting --> ready

- there are two types of CPU scheduling:

1. "non-preemptive":
- control of the CPU released by the process by its own, i.e. voluntarily
e.g. above case 1 & case 2

2. "preemptive":
- control of the CPU taken away forcefully from the process
e.g. above case 3 & case 4

+ CPU scheduling algorithms:
1. FCFS (First Come First Served) CPU Scheduling
2. SJF (Shortest Job First) CPU Scheduling
3. Priority CPU Scheduling
4. Round Robin CPU Scheduling

- there is need to decide which algo is efficicent and which algo is best suited at which sitution, there are certain criterias, called as "CPU scheduling criterias":
1. "CPU ultilization": one need to select such an algo in which utilization of the CPU must be as max as possible.

2. "throughput": total work done per unit time
one need to select such an algo in which throughput must be as max as possible.

3. "waiting time": total amount of time spent by the process into the ready queue for waiting to get control of the CPU from its time of submission.
- one need to select such an algo in which waiting time must be as min as possible.

4. "respose time": time required for the process to get first response from the CPU from its time of submission.
- one need to select such an algo in which response time must be as min as possible.

5. "turn-around-time": total amount of time required for the process to complete its execution from its time of submission.
- it is sum of periods spent by the process into the ready queue for waiting and onto the CPU for execution.
turn-around-time = waiting time + execution time.
- one need to select such an algo in which turn-around-time must be as min as possible.