Week 1:

OS is an interface between user and hardware. Its job is to: execute and make solving user programs easier, make systems more convenient to use, use hardware efficiently. Its job is to control and co-ordinate the use of hardware among application programs. Using OS allows the user to code in high-level language as it converts the language to instruct the hardware.

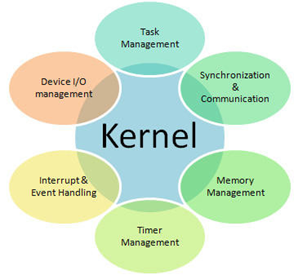
**User view of OS**: OS should offer ease of use and performance.

**System view of OS**: OS act as resource manager. OS should be able to manage all hardware resources in system. Hardware used in most efficient manner, process kept waiting for minimum amount of time if requires resources.

**Resource allocator:** Allocates resources (software & hardware) of computer system and manage them efficiently.

**Control Program**: Controls execution of user programs and operations of I/O devices.

**Kernel**: Program that executes forever (everything is an application compared to kernel), The internal part of the OS is often called the kernel.



* It’s a program that represents the core of computer.
* Controls everything occurring in system.
* Interacts with hardware on demand from applications.
* Doesn’t interact directly with user, interacts with shell, other programs, and hardware devices on system (including CPU, memory, disk drives).
* Kernel code is loaded into protected area of memory, so it isn’t overwritten by less frequently used parts of OS or application programs.

**Shell:** is the outermost part of an OS and a program that interacts with user commands. It’s a program that provides text-only user interface for OS. Term shell is from the fact it is outer layer of OS. Program gives user ability to communicate with OS. Shell is a program layer which understands and executes command user enters. Shell is also called command interpreter.

**OS File Manager:** Maintains information about files on system. E.g.(located in mass storage, size, type, protection, space available). Files allowed to group E.g. (directories/folders – hierarchical organisation).

**OS Device Drivers:** Software to communicate with peripheral devices or controllers. Each driver is unique. Translates general requests into specific steps for devices.

OS Memory Manager: Responsible for coordinating use of machine’s main memory. Decides what area of memory to allocate to program and its data. Allocates and deallocates memory for programs and knows what areas are free.

OS Schedular: Maintains record for present & new process, removes completed. Memory area(s) assigned, priority, state readiness to execute (ready/wait).

OS Dispatcher: Ensures process ready to run are executed. Time divided into small segments called time slice. When time slice over, dispatcher allows schedular to update process state for each process, select next process to run.

Week 2:

1950 Serial OS: entire program instructions in punched card, translated into card reader, submitted to the OS. Drawbacks: No user & computer system interactions, less memory, lot of time for execution, 1 program executed at a time. **1960 Simple Batch System:** High level language magnetic tapes, Jobs batched together based on language. Operations: place batched jobs on input. Program monitor manages execution of each program. Monitor utilities loaded when needed. ‘Resident monitor’ in main memory & available for execution. **Idea of SBS**: Reduce setup time by batching similar jobs, Alternate execution between user & monitor program, Rely on available hardware to effectively alternate execution from various parts of memory. Uses automatic Job sequencing (moves to next job when one is done). Job Control Language (JCL): provides instructions to monitor: What complier to user, what data to use. Resident Monitor/ Job Sequencer: First basic OS, – initial control in monitor, loads next program transfers control, job complete - control passed back to monitor. Desirable Hardware Features: Memory Protection memory area not altered by user program. Privileged Instructions – executed by resident monitor, traps occur if instructors tried. Interrupts –prevent job monopolising system. **Uniprogramming** (batch processing): Jobs are submitted /executed 1 by 1. Entire system used by 1 process at a time. Disadvantage: Wastage of CPU time, No User Interaction, No mechanism to prioritise processes. **1970 Multiprogrammed batch systems**: Memory Layout: several jobs in main memory at once, CPU is multiplexed among them. When process goes to I/O, CPU allocated another process. Uses Interrupts to run multiple programs simultaneously: When program performs I/O, execute another program until interrupt is received. Requires secure memory, I/O for each program, intervention if program loops indefinitely, CPU scheduling chooses next job. Advantages: CPU utilised all times, Processes finished in less time. Disadvantage: No user interactions.

Types of Multiprogramming OS: **Multitasking OS**: Ability to execute many programs at once. OS swaps every program I/o memory. Program fetched from memory then kept store in secondary memory until required. **Multiuser OS**: OS allows many users to connect with single system running the same OS. Difference between Multiprogramming and Multitasking OS: (Multiprogramming) context switching implemented. Allows enhancing CPU utilisation by using jobs to decrease the CPU idle time, using single CPU. (Multitasking) context switching, time sharing implemented. **1970 Time-Sharing:** Programs queued in FIFO order. Timer device interrupts after quantum (time slice), interrupted program returned to end of queue , next program is head of FIFO. Interactive – OS finishes execution of command, seeks next statement from user. Online filesystem. Several processes loaded into main memory simultaneously and several users share system. Aim - reduce overall process response time. CPU could execute several processes by providing equal time to each. Benefits: Multiple processes and user requests responded to simultaneously. Better response time. CPU not idle due to regular switching. **1970 Real-Time Systems:** Correct system function depends on timeliness. Feedback/control loops. Sensors and actuators. Hard real-time systems - Failure if response time too long, Secondary storage is limited. Soft real-time systems - Less accurate if response time is too long. Useful for application like VR. **1980 Multiprocessor Systems:** Multiple CPU in close communication. Improved Throughput, economical, increased reliability. Kinds: Vector, pipelined, Symmetric, asymmetric multiprocessing, Distributed memory/shared memory. Allows multiple processors connected with physical memory, computer buses, clocks, peripheral devices. Aim - consume high computing power, increase speed of system. Advantages: Great Reliability, Improve Throughput, Cost Effective System, Parallel Processing. Disadvantages: Expensive, Large memory, Speed degrades, Time delay - processor receives message takes appropriate action. Context switching. **Distributed/Loosely coupled systems** : Distribute among many processors. Loosely coupled - no shared memory, various communication lines. Client/server architectures. Advantages: Resource sharing. Computation speed-up. Reliability. Communication - e.g. email. Allows distributing system on processors, serves on multiple real time products, multiple users. Processors connected by communication medium; every processor contains own local memory along with other local processor. OS involves multiple computers, nodes, and sites, linked together with LAN/WAN lines. Capable for sharing computational capacity and I/O files, allowing virtual machine abstraction to users. Done within 3 areas: Client-Server System: “Tightly Coupled OS”. For multiprocessors and same kind multicomputer. Works as centralized server as it provides approval to all requests generated by client systems. Server systems divided into two segments: Computer Server System: Allows interface, client sends own all requests for executing. It sends to back response after executing action, transfer result to client. File Server System: Allows file system interface for clients to perform various tasks e.g. creating, updating, deleting files, etc. Objective – Hide, manage hardware resources. Peer-to-Peer System: “Loosely Couple System”. Contains bunch of processors no shareable memories/clocks. Every processors consist own local memory, make communication with each other through communication medium. Objective – provides local services to remote clients. Middleware: allows the interoperability in-between applications running on other OS. Using these services those applications are capable for transferring data to each other. Objective – Allows distribution transparency. Advantages: can share all resources between sites - increases data availability. enhances speed of data exchange. reduces probability of data corruption - data replicated on all sites. provides excellent services. decrease load of jobs on one host. scaled easily. more reliable to single system. excellent performance. Better portability. Better re-usability of existing hardware. decrease duration time in data processing. high fault tolerance system. Better flexibility. openness system because system can be accessed from local, remote sites. works independently. Well protective system due to Unique ID. Limitations: centre hub fails, entire network will halt. designed with language not defined till now. more costly. security issues arise while sharing data on networks. data packet corrupted due to following in large networks. maintenance costly. site gets overload, creates big challenges. same time multiple users try to access same data from database then its performance can degrade. Administration is difficult task. only supports few software’s.

**Week 3:The Process Concept**

An operating system executes a variety of programs: **Batch system** – jobs. **Time-shared systems** – user programs or tasks. **Process** – a program in execution; Process is dynamic/active entity that actions the purpose of the application. When we write and compile the code, we get an executable file. File created is known as a program. File created has all the instructions or the code within it. File is inactive unless executed so program is passive/static entity. A program is system activity that has a set of instructions, performs a specific task. **Batch processing systems** - executing jobs. **Real-time operating system** - program. User can run many programs simultaneously. Process is executing program. The process executes all code line in program. A process can create, delete, schedule another process. After you write a program in any language, two steps follow: (1.Compiling. 2. Running/Executing - makes that program a process.)Process is an instance of the program, has a shorter and minimal lifespan, many resources, considerable overhead. **Program can exist independently but a process cannot exist without a program.** A program's lifespan is longer - stored in secondary memory until erased, process' lifespan is shorter and limited - terminated after completed. **Process structure in RAM: (Text – program code. Data – contains global variables. Both fixed size because neither code/variables is going to change. Stack – contains temporary data, function parameters, return addresses, Local variables. Dynamic - difficult to determine the number of function calls required. Heap – memory allocated dynamically during run time - cannot determine the memory required.)**

**Process states: Process change state during lifetime (5 states):( (New –process is created. Ready –process in RAM, waiting for processor/CPU allocation. Running –process gets CPU and is executing. Waiting –process waiting for some event to occur or an I/O device. Terminated – the process finishes its execution normally/forcefully.) Step 1 − new process created, admitted into ready state. Step 2 − no other process present at running state, is dispatched to running based on scheduler dispatcher. Step 3 − higher priority process ready, uncompleted process sent to waiting state from running state. Step 4 − I/O or event is completed, process send back to ready state based on the interrupt signal**

given by running state. Step 5 − execution of process is completed in running state, will exit to terminate state, completion of process. Waiting, the process occupies main memory, constraint with limited memory, I/O/event might take time to complete.