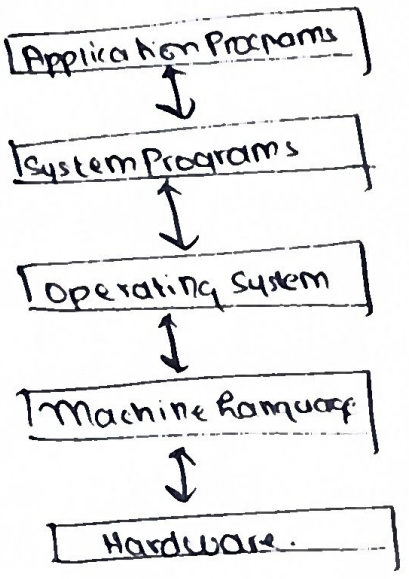
Unit 1 - Operating Systems Overview

* **Operating System** : A program that acts as an intermediary between a user of a computer and computer hardware.

Goals: → execute user programs and make solving user problems easier

→ make the computer system convenient to use

→ use the computer hardware in an efficient manner (contd. later)



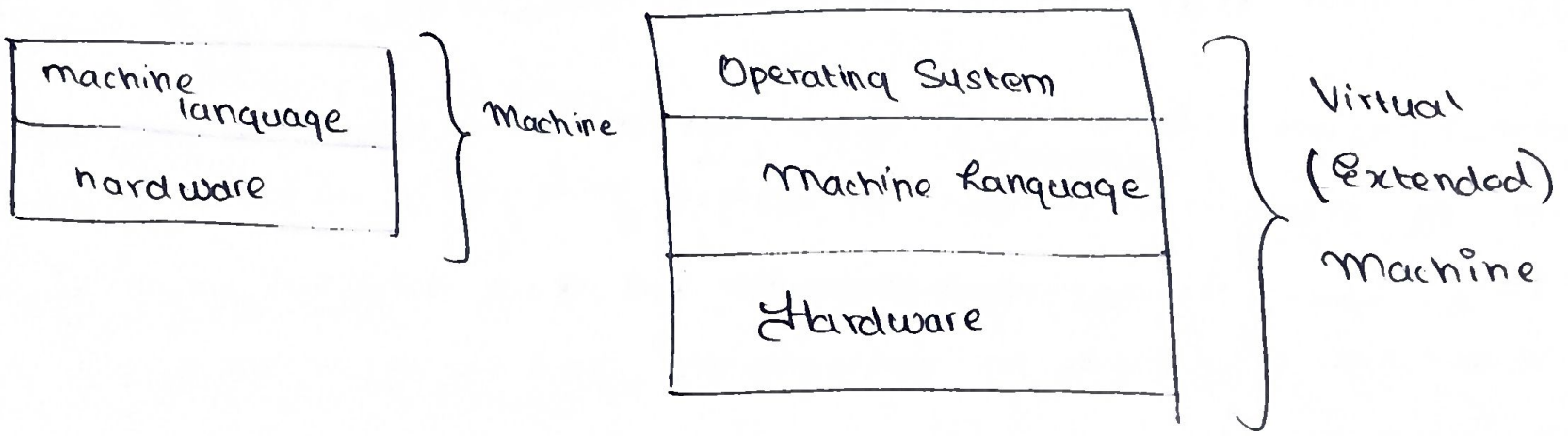
More generally, an OS can be defined in terms of the tasks that it performs

(i) Resource allocator : manages and allocates resources

(ii) Control program : controls the execution of user programs and operations of I/O devices

(iii) Kernel : the one program running at all times (all else being application programs)

* **Virtual Extended Machine** : With the advantage of easier programming provided by the OS, the hardware, its machine language and the OS constitute a new combination called the virtual (extended) machine



* Components of a Computer System

→ can be divided into 4 components: the operating system, the application programs and the users.

Hardware → the CPU, the memory and the I/O devices

Application Programs → word processors, spreadsheets, compilers & Web browsers

* The User View and the System View

User View → depends on the interface being used

A. Personal Computers – PCs have monitors, keyboards, mouse & a system unit. Such a system is designed for one user to monopolize its resources. In this case, the OS is designed more for ease of use, than for efficient resource utilization

B. Mainframe or Minicomputer : Other users also access the same computer through other terminals. These users share resources & may exchange information.

→ Here, the OS is designed to maximize resource utilization, to assure that all available CPU time, memory and I/O are used efficiently and that no user takes more than their fair share

C. Workstations and Servers: Users sit at workstations connected to networks of other workstations & servers

→ These users have dedicated resources at their disposal, but they also share resources such as networking and servers, like file & print servers

→ OS designed to compromise between individual usability & resource utilization.

System View

A. As a resource allocator : → The OS can be viewed as a resource allocator.

→ The OS has many resources that may be required to solve a problem- CPU time, memory space, file-storage space, I/O devices.

→ The OS acts as the manager of all these devices

→ It may face numerous & possibly conflicting requests for resources, and must decide how to allocate them to specific programs & users so that it can operate the computer system efficiently & fairly.

B. As a control program: → manages the execution of user programs to prevent errors and improper use of the computer.

→ concerned with the operation & control of I/O devices.

C. Kernel : → the OS is the one programming running at all times on the computer.

→ The OS software file is copied into the RAM, from the hard disk drive, during the boot-up.

→ The kernel remains in the RAM, while the computer is on, and is in charge of the overall operation of the operating system.

→ The kernel contains internal programs for the most often used operations like copying files.

→ responsible for

(a) file management
(b) memory management (RAM)
(c) security

* Types of Software → System software
                    → Application Software

Application Software - programs for performing a specific task

→ includes word processing, spreadsheets, gaming, web page design, graphic design

System Software → Operating software
                → Utility Software

Operating software - controls the overall operation of the computer

Utility software - software that extends or customizes the capabilities of the operating system.

includes → formatting
         → compress / decompress data
         → network communications

* Moore's Law : The number of transistors on an integrated circuit would double every 18 months

* Middleware → Mobile operating systems not only include a core kernel, but also middleware

→ A set of software frameworks that provide additional services to application developers.

→ Apple's iOS & Google's Android feature a core kernel along with middleware that supports databases, multimedia & graphics

* Goals of an Operating System (contd.)

→ Simplify the execution of user programs and makes solving user problems easier

→ Use computer hardware efficiently - allows for the sharing of hardware and software resources

→ Make application software portable and versatile.

→ Provide isolation, security and protection among user programs

→ Improve overall system reliability - error confinement, fault tolerance, reconfiguration

A. Operation
B. Storage
C. I/O structure

* Computer System Organization & Operation

Bootstrap Program - when a computer is powered up or rebooted, it needs an initial program to run.

→ Bootstrap program stored within the computer hardware in the ROM or EEPROM (Firmware)

→ Initializes all aspects of the system, from CPU registers to device controllers to memory contents

→ The bootstrap program must locate the operating system kernel and load it into the memory.

System Processes / System Daemons : Some services are provided outside of the kernel, by system programs that are loaded into the memory at boot time to become system processes or system daemons.

→ These run the entire time the kernel is running.

→ On UNIX, the first system process is init.

* Interrupts

→ The occurence of an event is usually signalled by an interrupt from either the hardware or software.

Hardware interrupts → through system bus
Software interrupts → through system calls

→ When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location. (Fixed location contains the starting address where the service routine for the interrupt is located.

→ The interrupt service routine executes, and on completion, the CPU resumes the interrupted computation.

Handling interrupts

→ Only a predefined no. of interrupts is possible. A table of pointers to interrupt routines can be used instead to provide the necessary speed.

→ These locations hold the addresses of the interrupt service routines for the various devices.

→ This array, called the | interrupt vector | is indexed by a unique device no for the given interrupt request.

* Storage Structures

A. General purpose computers run most of their programs from rewritable memory — RAM, specifically DRAM

B. ROM — cannot be changed, only static programs such as the bootstrap program is stored there.

· c. EEPROM - can be changed but not frequently, eg. smartphones have EEPROM to store factory installed programs.

D. Secondary Storage Devices

(i) magnetic disk → provides storage for both programs and data.
→ most programs, system and application programs are stored on a disc, until they are loaded onto the memory.

(ii) cache rom, CD-ROM

* Speed vs. Cost

→ As the cost of the memory decreases, the time taken to access it increases.

registers
↓↑
cache
↓↑
main memory } volatile
↓↑
solid state disk
↓↑
magnetic disk
↓↑
optical disk } non-volatile
↓↑
magnetic tapes

* Instruction-Execution Cycle - von Neumann Architecture

1. Fetch instruction from memory
2. Store instruction in instruction register.
3. Decode instruction
4. Fetch operands from memory and store in register.

* Why can't programs and data reside in the main memory permanently?

A. Main memory is too small to store all needed programs and data permanently.

B. Main memory is a volatile storage device that loses its contents when the power is turned off.

* I/O Structure

→ A general purpose computer system consists of CPUs and multiple device controllers, that are connected through a common bus.

→ There are device drivers for each device controller.

→ To start an I/O operation, the device driver loads the appropriate registers within the device controller.

→ The device controller examines the contents of these registers to determine what action to take (say, read from keyboard).

→ The controller starts the transfer of data

→ Once the data transfer is complete, the device controller informs the device driver via an interrupt that it has finished the operation.

→ The device then returns control to the operating system.

* Computer System Architecture on the basis of no. of processors

A. Single-Processor Systems

→ have only a single processor

→ one main CPU capable of executing a general-purpose instruction set - including instructions from users.

→ The single processor systems have other special-purpose processors as well.

→ These are device specific processors - such as for disk, keyboard or graphics.

→ These special purpose processors run on a limited instruction set and do not run user processes.

B. Multiprocessor Systems

→ also called **parallel systems** or multicore systems

→ have 2 or more processors in close communication, sharing the computer bus, and sometimes the clock, memory & peripheral devices.
↳ called tightly coupled system since processors share memory & clock

Advantages of Multiprocessor Systems

(1) Increased throughput - more work gets done in less time.
The speed up ratio w/ N processors is not N, but rather less than N.

(2) Economy of scale - cost less than equivalent multiple single-processor systems, because they can share peripherals, mass storage and power supplies

(3) Increased reliability: If fns. can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down.

* Graceful Degradation : The ability to continue providing service proportional to the level of surviving hardware is called graceful degradation

* Fault Tolerant : Some systems go beyond graceful degradation and are called fault tolerant, because they can suffer a failure of any single component and still continue operation. Fault tolerance requires a mechanism to allow the failure to be detected, diagnosed and if possible corrected.

* Types of Multiple-Processor Systems

A. Asymmetric Multiprocessing

→ Each processor is assigned a specific task

→ A boss process controls the system.

→ The other processors look at the boss for instructions, or have pre-defined tasks.

→ This scheme defines a boss-worker relationship. The boss processor, schedules and allocates work to the worker processes

B. Symmetric Multiprocessing (SMP)

→ Each process performs all tasks within an operating system.

→ SMP means that all processes are peers, no boss-worker relationship exists.

→ eg. AIX, a commercial version of UNIX designed by IBM.

* Difference between symmetric and asymmetric multiprocessing

Hardware differences: Special hardware can differentiate the multiple processors, or the software can be written to allow only one boss and multiple workers.

eg. Sun Microsystems' OS SunOS V4 provides asymmetric multiprocessing, but Version 5 (Solaris) is symmetric on the same hardware

* Memory Access Model in Multiprocessor Systems

Multiprocessing can cause a system to change its memory access model - from uniform memory access (UMA) to non-uniform memory access (NUMA).

UMA - defined as the situation where access to any RAM from any CPU takes the same amount of time.

NUMA - some parts of memory may take longer to access than other parts, creating a performance penalty.

* Multicore Systems

→ Multiple computing cores can be included in a single chip.

→ Such multiprocessor systems are termed multicore

→ They can be more efficient than multiple chips with single cores because one-chip communication is faster than between chip communication.

→ One chip w/ multiple cores uses significantly less power than multiple single-core chips.

→ All multicore systems are multiprocessor, but not all multiprocessor systems are multicore.

* Blade Servers → multiple processor boards, I/O boards and networking boards are placed in the same chassis.

→ The difference between these and traditional multiprocessor systems is that each blade-processor board boots independently and it runs its own operating system.

c. Clustered Systems

→ gather together multiple CPUs.

→ different from the multiprocessor systems, in that they are composed of 2 or more individual systems - or nodes - joined together. (said to be loosely coupled)

→ Clustering provides high-availability service, the service will continue even if one or more systems in the cluster fail.

* Types of Clustering

a. Asymmetric Clustering - One machine is in hot standby mode while the other is running the applications.

→ The hot standby host machine does nothing but monitor the active server.

→ If that server fails, the hot-standby host becomes the active server.

b. Symmetric Clustering - 2 or more hosts are running applications and are monitoring each other.

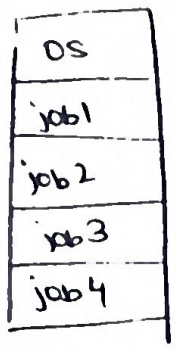
→ more efficient, since it uses all the available hardware.

* Parallelization - Divides a program into separate components that run in parallel on individual computers in the cluster.

* SANs : Storage area networks, which allow many systems to attach to a pool of storage

→ If applications and their data are stored on the SAN, then the cluster software can assign the application to run on any host attached to the SAN.

* Multiprogramming Systems (Batch Systems)



→ Multiprogramming increases CPU utilization by organizing jobs, so that the CPU always has one to execute.

→ Since the main memory is too small to accommodate all jobs, the jobs are initially kept on the disk in the job pool.

→ The OS picks and begins to execute one of the jobs in the memory.

→ Eventually, the job may have to wait for some task, such as an I/O operation to complete.

→ In a non-multiprogrammed system, the CPU would sit idle. In a multiprogrammed system, the operating system switches to and executes another job.

→ As long as there is at least one job that needs to execute, the CPU is never idle.

* Time Sharing

→ In time-sharing systems, the CPU executes multiple jobs by switching among them, but the switching occurs so frequently that the users can interact with each program while it is running.

→ A time-shared operating system allows many users to share the computer simultaneously.

→ Time sharing requires an interactive computer system, which provides direct communication between the user and the system.

→ The user gives instructions to the OS through a program, or through the keyboard/mouse/touch pad.

→ When the OS finishes the operation of one command, it seeks the next control statement from the user's keyboard.

→ In a time-sharing system, the OS must ensure reasonable response time. This is achieved through swapping whereby processes are swapped in and out of the main memory to the disk.

* Traps / Exceptions

→ a software-generated interrupt caused either by an error or by a specific request from a user program that an operating system service be performed.

→ For each type of interrupt, separate segments of code in the OS determine what action should be taken.

* Dual and Multimode Operation

A. Dual Mode

→ There is a need for 2 separate modes of operation: the user mode and Kernel mode (also called the supervisor mode, system mode or priveleged mode).

→ A bit, called the mode bit is added to the hardware of the computer to indicate the current mode: kernel(0) or user(1)

→ When the computer system is operating on behalf of a user application, the system is in user mode. However, when a user application requests a service from the OS, via a system call, the system transitions to kernel mode to fulfil the request.

→ At system boot time, the hardware starts in kernel mode and starts user applications in user mode.

→ The dual mode of operation provides us with a means for protecting the operating system from errant users - This is done by designating some of the machine instructions that may cause harm as priveleged instructions

B'. Multimedia Operations

→ CPUs that support virtualization have a separate mode to indicate when the virtual machine manager (VMM) is in control of the system.

→ The VMM has more privileges than the user, but less than the kernel.

* Life Cycle of Instruction Execution

→ Initial control w/ OS

→ when control is given to a user application, the mode is set to user mode.

→ System calls provide the means for a user program to ask the OS to perform tasks reserved for the OS on the user program's behalf.

→ When a system call is executed, it is treated by the hardware as a software interrupt.

→ Control passes through the interrupt vector to a service routine in the OS, and the mode bit is set to kernel mode.

→ The kernel examines the interrupting instruction to determine what system call has occurred.

* Timers

→ should not allow user program to get stuck in an infinite loop or fail to return control to the operating system.

→ A timer is used to interrupt the computer after a specific period.

→ Usually incorporated using a fixed clock rate and counter.

→ Every time the clock ticks, the counter is decremented, and when the counter reaches 0, an interrupt occurs.

* Caching?

→ Info is usually kept in some storage system (such as the main memory)

→ As it is used, it is copied into a faster storage system, i.e the cache, on a temporary basis

→ When we need a piece of information, we check if it is in the cache or not.

→ If it is we use the information directly from the cache, otherwise the information is taken from the source, putting a copy into the cache under the assumption that we will need it again soon.

→ Internal programmable registers, such as index registers, provide a high speed cache for the main memory.

→ There are other caches like the instruction cache, to hold the instructions expected to be executed next.

→ The main memory can be viewed as a cache for secondary storage since data in the secondary storage must be copied into the main memory for use.

* Computing Environments

A. Traditional Computing

→ PCs connected to a network, w/ servers providing file & print services

→ usually single computers for personal use

B. Mobile Computing

→ computing on handheld smartphones & tablets

→ are portable and lightweight

→ Apple iOS and Google Android

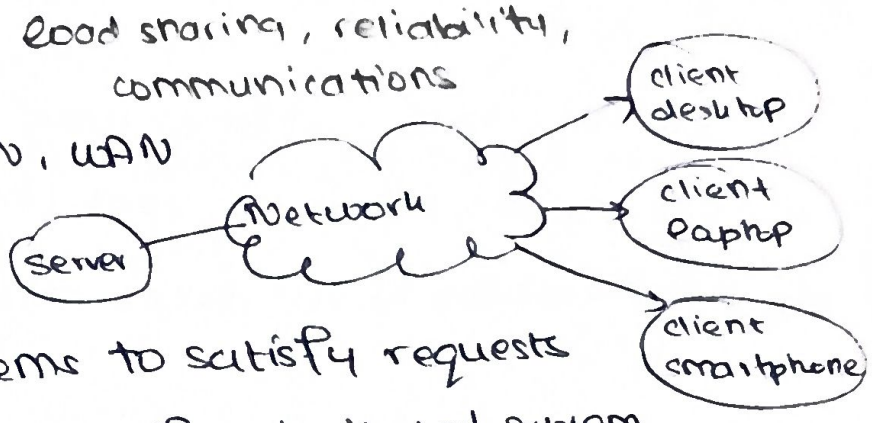
C. Distributed Systems

→ a collection of physically separate, heterogenous computer systems, that are networked to provide users with access to the various resources that the system maintains.

→ Access to a shared resource increases computation speed, functionality, data availability & reliability.
→ loosely coupled - each processor has its own local memory

→ Distributed systems depend on networking for their functionality
- use the TCP/IP network protocol
— can write about PAN, MAN, WAN

advantages - resource sharing, load sharing, reliability, communications



D. Client-Server Computing

→ some systems act as server systems to satisfy requests generated by client systems, a special kind of distributed system.

(i) Computer-Server System - provides an interface to which a client can send a request to perform an action

(ii) File-Server System - clients can create, update, read and delete files. eg, a web server that delivers files to clients running web browsers.

Advantages - central location, cheaper.
Disadvantages - a single source for all problems

E. Peer-to-Peer Computing

→ Clients and servers are not distinguished from one another

→ All nodes are considered peers, and can act as a client or server, depending on whether it is requesting or providing a service.

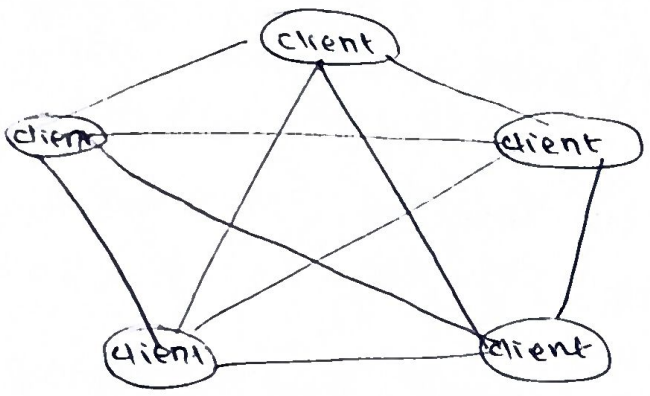
→ Advantageous over client-server systems, since in that case, the server is a bottleneck.

→ To participate in a peer-to-peer system, a node must first join the network of peers. Then it can begin requesting and providing services from the other nodes.

Determining which services are available is done in 2 ways:

① when a node joins a network, it registers its service w/ a centralized lookup service on the network. Any node in need of a service contacts this centralized lookup service to determine which node provides the service.

② An alternate scheme uses no centralized lookup service. Instead a peer must discover what node provides the service it wants by broadcasting to all other nodes (called discovery protocol)



F. Virtualization

→ allows operating systems to run as applications within other operating systems

eg. emulation, when source & target CPU are different

comes at a heavy price though - since every machine level instruction that runs natively on the source system must be translated to the equivalent fn. on the target system

G. Cloud Computing

→ a type of computing that delivers computing, storage & applications as a ~~service~~ service across a network

- Types:

(i) public cloud - available via the Internet to anyone willing to pay for the services

(ii) private cloud - run by company for their own private use

(iii) hybrid cloud - has both public & private components

(iv) Software as a Service - SaaS - one or more applications such as word processors or spreadsheets

(v) Platform as a Service - PaaS - a software stack ready for application use via the Internet (a database server)

(vi) Infrastructure as a service - IaaS - servers or storage available over the Internet

11. Real-Time Embedded Systems

→ have very specific tasks

→ embedded systems almost always run real-time operating system. A real-time system is used when rigid requirements have been placed on the operation of a processor or the flow or data.

→ Sensors bring data to the computer.

(Real time systems have well-defined, fixed time constraints).

hard

(i) limited 2° storage

(ii) not supported by general OS

soft

(i) limited utility in industrial control of robs

(ii) useful in applications requiring advanced OS features

* Enhancing an OS

--> provides service not included with OS

A. Backup Software

(i) archives files onto removable media

(ii) ensures data integrity

B. Antivirus Software

(i) finds, blocks & removes viruses

(ii) must be updated regularly

(iii) McAffee & Norton

C. Firewall

(i) a crucial utility

(ii) protects computer from intruders

* Virtual Memory → ability of the CPU & the OS to use the hard disk as additional RAM, when needed

Advantage - no longer have to worry about insufficient memory

Disadvantage - performance is slow when accessing VM

* History of Operating Systems

→ started w/ computer hardware in the 1940s, initially used vacuum tube technology

→ programs were loaded manually into the memory using switches punched cards, or paper tapes

Structure

→ large machines run from console

→ very ineffective, single operator

→ used paper tape or punched cards

Software

→ assemblers, compilers, libraries of common subroutines

Inefficient use of expensive resource

Simple batch systems (1960s)

→ Reduce setup time by batching jobs w/ similar requirements

→ uses a card reader, and an operator was hired

→ automatic job sequencing

↓

user submits card deck
card pull on tap
tape processed by operator
output written to tape
tape printed

Problems

→ long turnaround time
(upto 2 days)

→ low CPU utilization

Unit - 1

Evolution of the Operating System

Phase1: hardware is expensive, humans are cheap

- mainframe systems
- human operators
- vacuum tubes
- punch cards

Phase2: cheap hardware, expensive humans

- batch programming
- time sharing systems

Phase 3 : cheap hardware , expensive humans

- multitasking
- multiprocessing
- distributed systems

Phase4: cloud computing

- Saas
- Paas
- Private & Public cloud

Operating System Services

(i) User interface

(ii) program execution

(iii) I/O operations

(iv) File system manipulation

(v) communications

(vi) Error Detection

(vii) Resource Allocation

(viii) Accounting

(ix) Protection & Security

* System Calls

→ a programming interface to the services provided by the OS

→ written in a high level programming language

→ implementation : each sys call associated with a no

→ system call interface maintains a table indexed according to these numbers

→ system call interface invokes the intended system call in the OS kernel & returns status value

→ most implementation details hidden from user.

Passing Parameters : (i) in registers

(ii) parameters stored in a block in mem, pass address

(iii) params pushed onto a stack

Types of System calls

(i) creation, termination & process

(ii) wait, signal

(iii) end, abort

(iv) file management

(v) device management

fork()
exit()
wait()

open()
read()
write()

getpid()

shmget()
shmat()

chmod()
chown()

* System Programs

→ provide a convenient environment for program development and execution

(i) File management

(ii) Status management - date, time, performance

(iii) Programming Language Support

(iv) Programming Loading & Execution

(v) Communication

(vi) Background Services (daemons)

(vii) Application programs

* Operating System Structure

simple structure - MSDOS

more complex - UNIX

layered -

microkernel - Mac

Simple Structure MS-DOS

→ designed to provide the most functionality in the least space

→ no clear structure

Non-Simple Structure : UNIX

has 2 parts : Kernel
System programs } elaborate

Layered Approach

→ has a no. of layers each built on top of other layers

→ bottom layer = hardware layer

→ highest layer = UI

→ has modularity, each layer can only use fns. of lower layers

Microkernel

→ moves as much of the kernel as possible into user space

→ communication through message passing

→ benefits : (i) easy to extend a microkernel
(ii) more reliable & secure

→ can sometimes have overhead of user space to kernel space communication

Hybrid Systems

→ combine multiple models
→ Linux = monolithic
→ windows - mostly monolithic + microkernel for personalities