OPERATING SYSTEM

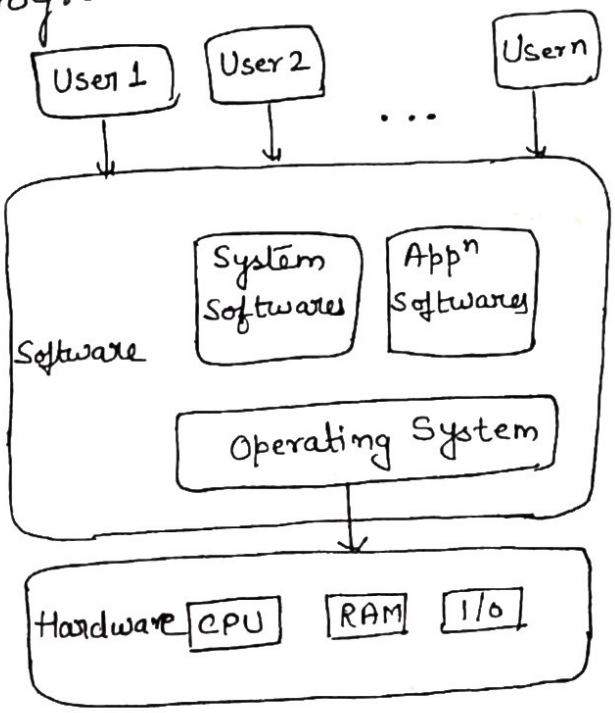
Lecture :

UNIT-1

INTRODUCTION :- An Operating System (OS) is an interface between a computer user and computer hardware. An OS is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, & controlling peripheral devices like disk drives & printers.

Some popular O.S. include :- Linux, Windows, OS X, VMS, OS/400, AIX, z/OS, etc.

Definition :- An operating system is a program that acts as an interface between the user and the computer hardware & controls the execution of all kinds of programs.



Following are some important functions of an O.S.:-

→ Memory Management
→ Processor Management
→ Device Management
→ File Management
→ Security
→ Control over system performance
→ Job accounting
→ Error detecting aids
→ Coordination between other softwares & users.

MEMORY MANAGEMENT

Memory management refers to management of Primary Memory or Main memory. Main memory is a large array of words or bytes where each word or bytes where each word or byte has its own address.

→ Keep tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.
→ In multiprogramming, the OS decides which process will get memory when and how much.
→ Allocates the memory when a process requests it to do so.
→ De-allocates the memory when a process no longer needs it or has been terminated.

PROCESSOR MANAGEMENT

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called process scheduling. An OS does the following activities for processor mgmt.

→ Keeps tracks of processor & status of process. The program responsible for this task is known as traffic controller.

→ Allocates the processor (CPU) to a process.

→ De-allocates processor when a process is no longer required.

DEVICE MANAGEMENT

An O.S. manages device communication via their respective drivers. It does the following activities for device mangement —

→ Keep tracks of all devices. Program responsible for this task is know as the I/O Controller.

→ Decides which process gets the device when and for how much time.

→ Allocates the device in the efficient way.

→ De-allocates devices.

FILE MANAGEMENT

A file management system is normally organized into directories for easy navigation and usage. These directories may contain files & other directions.

An O.S. does the following activities for file management:-

→ Keeps track of information, location, users, status etc. The collective facilities are often known as file system.

→ Decides who gets the resources.

→ Allocates the resources.

→ De-allocates the resources.

Other Important Activities.

→ Security :- By means of password & similar other techniques, it prevents unauthorized access to programs & data.

→ Control over system performance :- Recording delays between request for a service & response from the system.

→ Job accounting :- Keep track of time & resources used by various jobs & users.

→ Error detecting aids :- Production of dumps, traces, error messages, and other debugging & error detecting aids.

→ Coordination between other softwares & users :- Coordination & assignment of compilers, interpreters, assemblers & other software to the various users of the comp^r system.

CLASSIFICATION OF OPERATING SYSTEM

Operating systems are there from the very first comp^r generation and they keep evolving with time.

1. Batch Operating System :- The users of a batch operating system do not interact with the computer directly. Each user prepares his job on an offline device like punch cards & submits it to the computer operator. To speed up processing, jobs with similar needs are batched together an ·oun as a group. The programers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.

The problems with Batch Systems are as follows-

→ Lack of interaction between the user & the job.
→ CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.
→ Difficult to provide the desired priority.

2. Interactive Operating System :- An interactive operating system is one that allows the user to directly interact with the operating system whilst one or more programs are running.

There will be an user interface in place to allow this to happen. It could be a command line style of interface or it could be a graphical interface

3. Time-sharing operating systems :-

Time sharing is a technique which enables many people, located at various terminals, to a particular computer system at the same time. Time-sharing or multi tasking is a logical extention of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing.

⇒ The main difference between multiprogrammed Batch systems and Time sharing systems is that in case of Multiprogrammed batch systems, the objective is to maximize processor use, whereas in Time Sharing, the objective is to minimize response time.

Advantages :-

→ Provides the advantage of quick response
→ Avoids duplication of software
→ Reduces CPU idle time

Disadvantages :-

→ Problem of reliability
→ Question of security & integrity of user programs & data
→ Problem of data comm^n.

4. Real time System:-

A real time system is defined as the data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. The time taken by the system to respond to an ~~system~~ input & display of required updated information is termed as the response time. So in this method, the response time is very less as compared to the online processing.

2 types of real time systems.

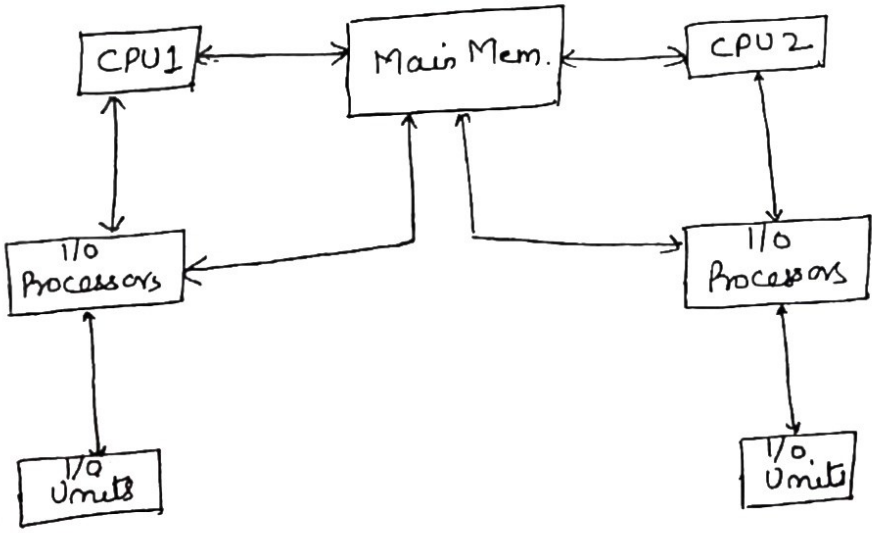
(a) Hard real time systems:- Hard r.t.s. guarantee that critical tasks complete on time. In hard rts, secondary storage is limited or missing & the data is stored in ROM. In these systems, virtual memory is almost never found.

(b) Soft real time systems:- Soft real time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects like undersea exploration & planetary rovers, etc.

5. Multiprocessor Systems :-

Multiprocessing is the use of two or more CPUs within a single computer system. The term also refers to the ability of a system to support more than one processor and/or the ability of a system to support more than one processor to allocate tasks between them. These multiple processors are in close comm^{n} sharing the computer bus, memory & other peripheral devices. These systems are referred to as tightly coupled systems.

These types of systems are used when very high speed is required to process a large volume of data. These systems are generally used in environment like satellite control, weather forecasting etc. The basic organization of multiprocessing system –



6. Multiuser Systems :-

A multi user operating system is an O.S. that allows multiple users on different computers or terminals to access a single system with one OS on it.

Timesharing systems are multiuser systems. Most batch processing systems for mainframe computers may also be considered "multi-user", to avoid leaving the CPU idle while it waits for I/O operations to complete.

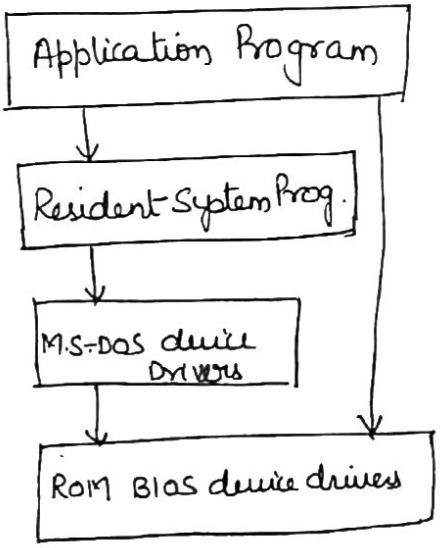
7. Multiprocess Systems :- Multiprocesser O.S. refers to

8. Multithreaded System :- Multithreading is the ability of a program or an O.S. process to manage its use by more than one user at a time and to even manage multiple requests by the same user without having to have multiple copies of the programing running in the computer.

OPERATING SYSTEM STRUCTURE

The design of an operating system architecture traditionally follows the separation of concerns principle. This principle suggest structuring the O.S. into relatively independent parts that provide simple individual features, thus keeping the complexity of the design manageable.

1. Simple Structure :-



→ In MS DOS, applications may bypass the O.S.
→ O.S. such as MS-DOS and the original UNIX did not have well defined structures
→ There was no CPU Execution Mode (user & kernal), and so errors in applications could cause the whole system to crash.

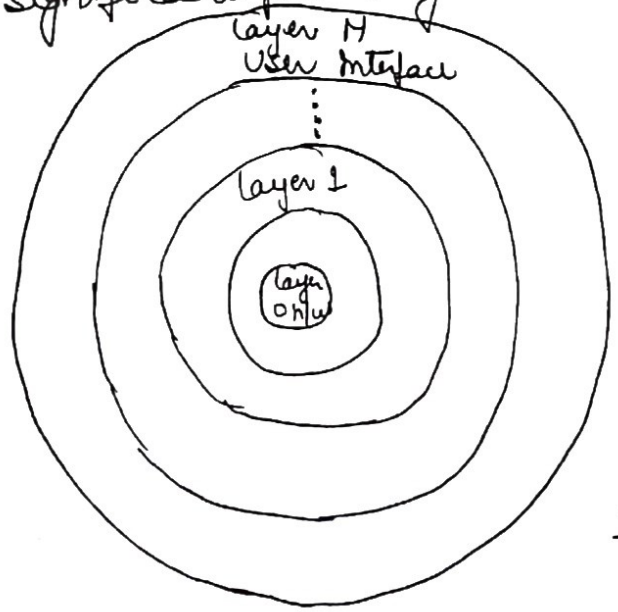
2. Layered Approach

→ Another approach is to break the OS into a number layers, each of which rests on the layer below it, and relies solely on the services provided by the lower layer.

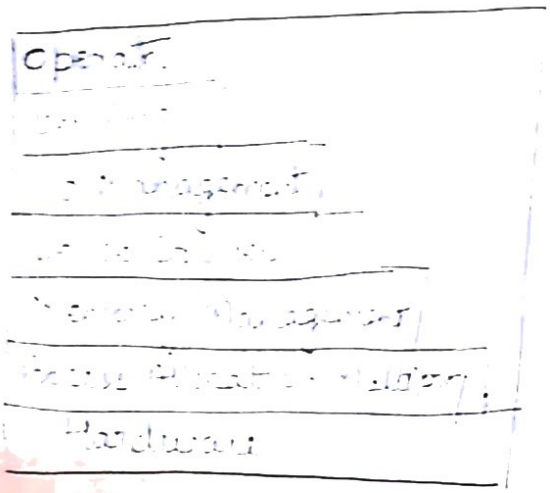
→ This approach allows each layer to be developed & debugged independently, with the assumption that all lower layers have already been debugged & are trusted to deliver proper services.

→ The problem is deciding what order in which to place the layers, as no layer can call upon the services of any higher layer. & so many

→ Layered approaches can also be less efficient, as a request for service from a higher layer has to filter through all lower has to filter through all lower layers before it reaches the HW, possibly with significant processing at each step.



A layered O.S.



Layered Architecture

3. Microkernals

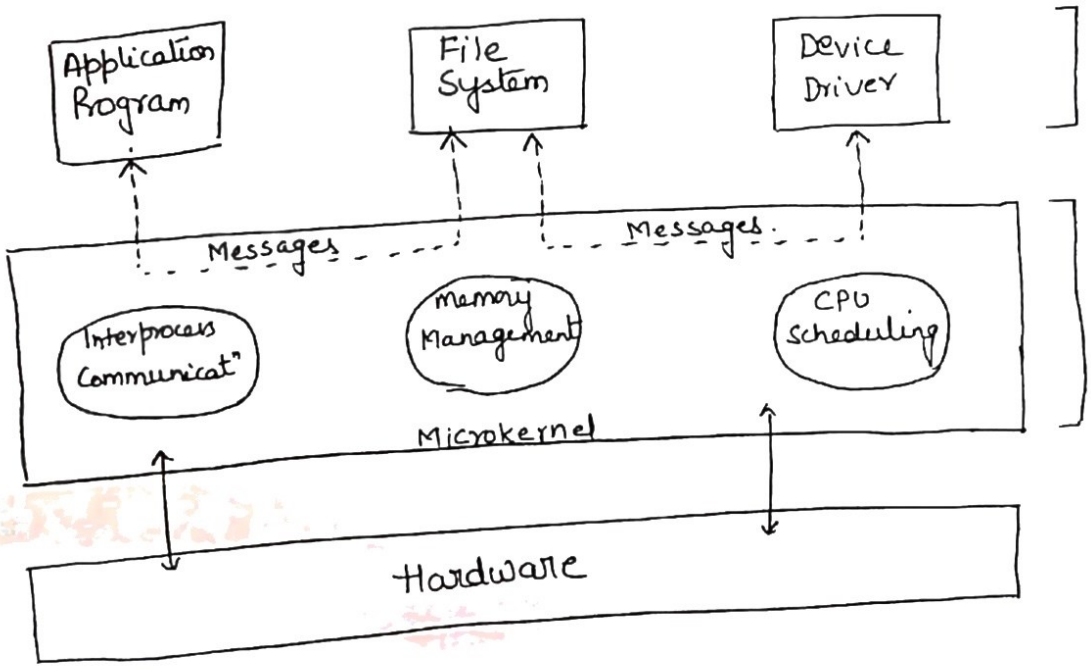
The basic idea behind micro kernals is to remove all non-essential services from the kernal, and implement them as system applications instead, thereby making the kernal as small and efficient as possible.

→ Most micro kernels provide basic process & memory management & message passing between other services & not much more.

→ Security & protection can be enhanced, as most services as performed in user mode not kernal mode.

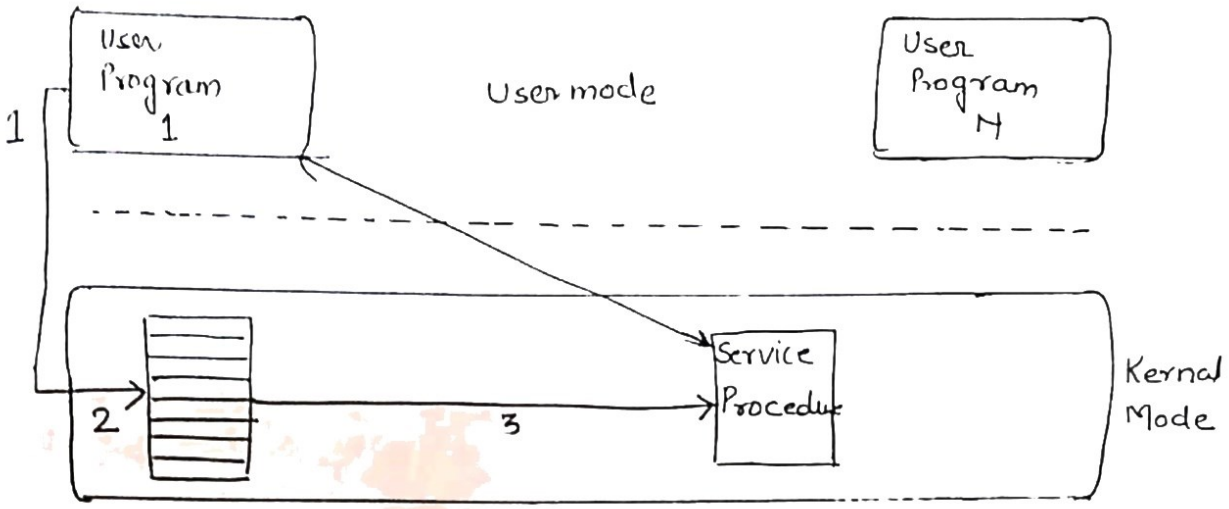
→ System expansion can also be easier as it involves adding more system applications, not rebuilding a new kernal.

Eg. of micro kernal. QNX, a real time O.S. for embedded system.



Architecture of a typical microkernal

4. Monolithic Architecture of OS



Monolithic Structure of OS

It is the oldest architecture used for developing OS Operating system resides on kernal for anyone to execute. System call is involved i.e. switching from user mode to kernal mode and transfer control to O.S. [Event 1]

Many CPUs have 2 modes have 2 modes,

Kernal Mode for O.S. in which all instructions are allowed & user mode for user program in which I/O devices and certain other instruction are not allowed.. Then OS then examine the parameter of the call to determine which system call is to be carried out [Event 2]. Next, the OS index's into a table that contains procedure that carries out system call [Event 3] Finally, it is called when the work has been completed & the system call is finished, control is given back to the user mode [Event 4].

SYSTEM COMPONENTS

Even though, not all systems have the same structure many modern OS share the same goal of supporting the following types of system components.

1. Process Management :-

→ Creation & deletion of user & system processes.
→ Suspension & resumption of processes.
→ A mechanism for process synchronization
→ A mechanism for process communication.
→ A mechanism for deadlock handling.

2. Main Memory Management

→ Keep track of which part of memory are currently being used & by whom
→ Decide which process are loaded into memory when memory space becomes available.
→ Allocate & deallocate memory space as needed.

3. File Management

→ The creation & deletion of files.
→ The creation & deletion of directories
→ The support of primitives for manipulating files & directories.
→ The mapping of files onto secondary storage.
→ The backup of files on stable storage media.

4. I/O System Management

I/O subsystem hides the peculiarities of specific hardware devices from the user. Only the device drivers know the complexities of the specific device to whom it is assigned.

5. Secondary Storage Management :-

→ Managing the free space available.
→ Allocation of storage space when new files are written.
→ Scheduling the requests for memory access.

6. Networking

A distributed systems is a collection of processors that do not share memory, peripheral devices, or a clock. The processors communicate with one another through comm^n lines called n/w. The comm^n-n/w design must consider routing & connection strategies, and the problems of contention & security.

7. Security :-

Security refers to the mechanism for controlling the access of programs, processes or users to the resources defined by a computer system.

OPERATING SYSTEM - SERVICES

An O.S. provides services to both the users & to the programs.

→ It provides programs an environment to execute

→ It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system.

→ Program Execution
→ I/O operations
→ File system manipulation
→ Communication
→ Error Detection
→ Resource Allocation
→ Protection.

1. Program Execution

→ Loads a program into memory
→ Executes the program.
→ Handles program's execution
→ Provides a mechanism for process synchronizat^{n}
→ Provides a mechanism for process comm^{n}
→ Provides mechanism for deadlock handling.

2. I/O Operation

An I/O subsystem comprises of I/O devices & their corresponding drivers softwares. Drivers hide the peculiarities of specific hardware devices from the users.

An OS manages comm^n b/w user & device drivers.

→ I/O operation means read or write operation with any file or any specific I/O device.

→ O.S. provides the access to the required I/O device when required.

3. File System Manipulation

→ Program needs to read a file or write a file.

→ The O.S. gives the permission to the program for operation on file.

→ Permission varies from read-only, read-write, denied and so on.

→ O.S. provides an interface to the user to create/delete files.

→ O.S. provides an interface to the user to create/delete directories.

→ O.S. provides an interface to create the backup of file system.

3. Communication

→ Two process often require data to be transferred between them.

→ Both the processes can be on one computer or on different computers, but are connected through a computer network.

→ Comm^n may be implemented by two methods, either by shared or by message passing.

→ Comm^n.

4. Error Handling

Error may occur in CPU, in I/O devices or in memory h/w.

→ The OS constantly checks for possible errors.

→ The OS takes an appropriate action to ensure correct & consistent computing.

5. Resource Management

→ The O.S. manages all kinds of resources using schedulers.

→ CPU scheduling algorithms are used for better utilization of CPU.

- The OS provides authentication because
authentication is carried out by
trusted.

RE-ENTRANT KERNEL

A reentrant kernel enables processes to give away the CPU while in kernel mode, not hindering other processes from also entering kernal mode. Thus in re-entrant kernal several processes may be executing in Kernal Mode at the same time.

A typical use is IO wait. The process want to read a file. It calls a kernel fun" for this side. Inside the kernel fun", the disk controller is asked for the data. Getting the data will take some time & the fun' is blocked during that time. With a reentrant Kernel, the scheduler will assign CPU to another process untill an interrupt from the disk controller indicates that the data is available & the earlier thread can be resumed. This process can still access IO, like user input.

How to acheive re-entrancy ?

First way of providing re-entrancy is to write functions so that they modify only local variables and do not alter global data structures. Such

functions are referred to as "Re-entrant functions". A Kernel that implements 'reentrant routines" are reffered to as "Re-entrant Kernels"

Other than using 're-entrant functions", a Kernel can also use "Locking" methods to ensure that only one process can execute a non-re-entrant function at a time.

Difference between Non-reentrant & Re-entrant Kernel

In Non-reentrant kernal mode, every single process acts on its own set of memory locations & thus cannot interfere with the others.

On a 're-entrant kernel', when a h/w interrupt occurs, it is able to suspend the current running process, even if the process is in Kernel Mode. This improves the throughput of the device controllers that issue interrupts. What exactly happens is this: When a device issues an interrupt, it waits for the CPU to acknowledge. If the Kernel answers quickly, the device controller will be to perform other tasks while the CPU handles the interrupt.