Operating System

"An operating system is an imp. part of almost every computer system. It is a prog. that manages the computer h/w and act as an interface b/w user of a computer and computer h/w."

Need of O.S:- H/W ⇒ The central processing unit (CPU), m/m and peripheral devices.
S/W ⇒ All the application programs i.e spreadsheets, compilers, word processors etc.

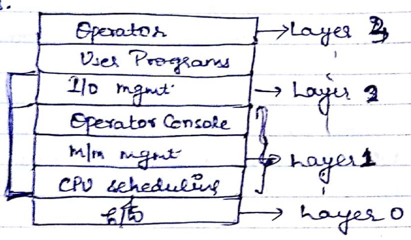
The O.S system controls and co-ordinates the use of the h/w among the various application programs for the various types of users. like "government". It provides the environment for others.

Abstract view of O.S on page no. 4 in Galvin.

Point of views of O.S:- ① - User view - It includes diff. types of user with their diff. purposes for using the computer system ex- Normal user, Other users connected with the teminals of ~~main~~ computers, mainframe comp. or connected with work stations/servers on a n/w.
② - System view - ~~It~~ includes or defines the O.S as resource allocator, and control program.

Reid & Taylor

Layered Architecture of O.S :- It was developed into 60's in this approach; the O.S is broken up into no. of layers.



Layer 0 - Dealt with allocation of the processor, switching the processes, when interrupts occures or timers expires.

Layer 1 - It allocates the m/m for processes and and manage them.

~~Layer 2 - Handled communication b/w each~~ process ~~and the operator~~ console

Layer 2 - Took care of managing the I/O devices and buffering the infn streams to and from them.

Layer 3 - It is the layer where user accessed the O.S.

Types of O.S -

① - Batch Operating System - Common I/P devices were card readers and tape drives. O/P devices were line printers, tape drives, punch cards.

"User prepared a job - which consisted of the program, the data, some control infn about the nature of the job Control cards and submitted to computer."

To speed up - operators batched together jobs with similar needs and ran them through the computer as group.
Very slow.

② - Multiprogramming O.S - It increases CPU utilization by organising jobs so that the CPU always has one to execute. In this condition CPU is never idle.

③ Time sharing O.S - (Multitasking) It is a logical extension of multiprogramming. How multi-program run simultaneously with the Interaction of O.S.

* Response time is fast.
* Reduces CPU idle time.

Virtual h/w deviced provided by the virtual machine, are mapped to real h/w on physical m/c.

ex- virtual machine's virtual hard disk is stored in a file located on your hard drive.

O.S ~~services or~~ functions components-

1. - Process Management
2. - Memory "
3. - File "
4. - I/O System "
5. - ~~Device~~ Protection system.

Services-

- Program execution
- I/O programs operations.
- File system manipulation.
- Communication
- Error Detection.
- Resource Allocation.
- Accounting
- Protection

BIOS (Basic I/O system)- It is the program a personal computer's microprocessor uses to get the computer system started after you turn it on.
It also manages data flow b/w the computer's O.S and attached devices such as the hard disk, adapter, keyboard etc.

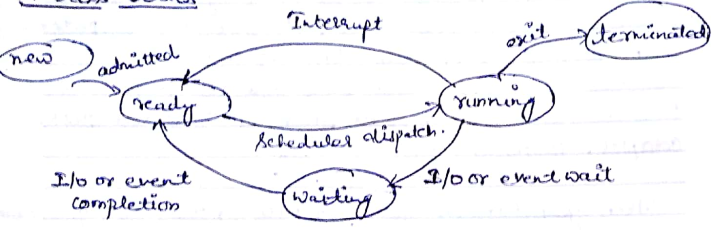
When you turn on your computer, the microprocessor passes control to the BIOS program which is always located at the same place on EPROM chip.
When BIOS boots up your computer, it first determines whether all of the attachments are in place and operational and then it loads the O.S into your computer's RAM from your hard disk.

Process Management- Program- A specific set of instructions (ordered) for a computer to perform & produce an O/P.

Process- A program in execution called process. It also includes the current activity, which is represented by the value of program counter.

Program	→ Passive Entity
Process	→ Active Entity

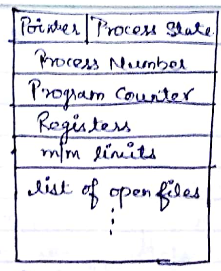
Process States -



New - The process is being created.
Ready - The process is waiting to be assigned to a processor.
Running - Instructions are being executed
Waiting - The process is waiting for some event to occur (such as I/O completion or reception of signal)

Termination - Finished execution and exit.

Process Control Block - Each Process is represented in the O.S by a PCB also called task control block. It contains many pieces of infⁿ associated with a specific process. They are :



(PCB)

Process State - The state may be new, ready, running, waiting, halted and so on.

Program Counter - The counter indicates the address of the next instruction to be executed for this process.

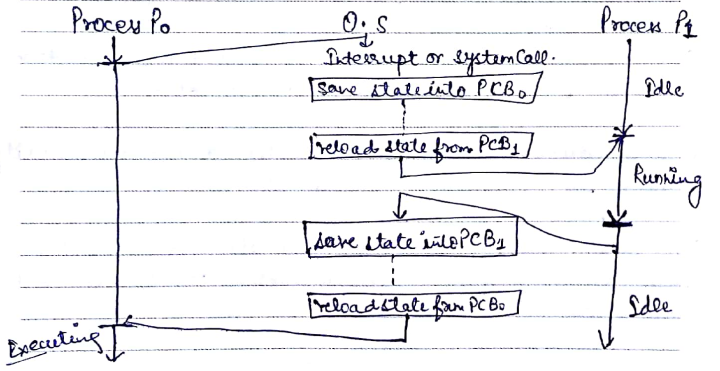
CPU Registers - It includes accumulators, index-registers, general purpose registers etc.

CPU scheduling Infⁿ - It includes a process priority, pointers to scheduling queues etc.

M/m mgmt' Infⁿ - This includes the value of base and limit registers, page tables or the segment tables etc.

Accounting Infⁿ & I/O Status Infⁿ → Includes the amount of CPU and real time, process no., which devices allocated to this process, list of open files related to this process etc.
~~Diagram~~

Context Switching Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process. This task is called "context switch."
The context switch of a process is represented by PCB of a process.



Assign. Q. -
Is LINUX A Kernel or An O.S?
Diff. b/w Types of Kernel.
System Calls in Unix & Windows.
O.S is a resource allocator. Justify

Kernel - A Kernel is the part of the O.S that mediates access to system resources. It is responsible for enabling multiple applications to effectively share the H/W by controlling access to CPU, m/m, disk I/O, and networking.
It acts as the interface b/w user applications and the H/W.

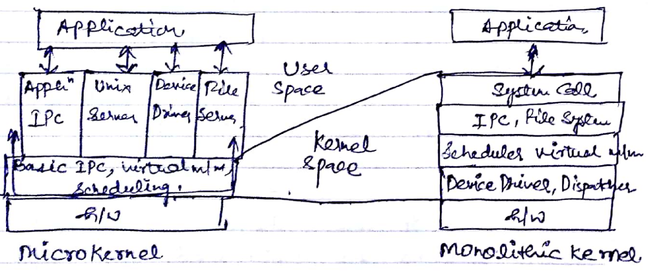
Types of Kernel - Two categories -
① - Monolithic
② - Micro Kernel.

Monolithic → It manages applⁿ and H/W. The user services and Kernel services are implemented under same address space. This increases the size of the kernel further increases the size of O.S.
All services like CPU scheduling, m/m mgmt., file mgmt. are done by system calls. As user services and kernel services both resides in same address space, this results in fast executing O.S.
* If any one service fails. entire system is crashed.

Micro Kernels → In this, the user services and the Kernel services are Implemented in different address space. The user services are kept in

user address space and the kernel services are kept in kernel address space. This reduces the size of the kernel and then O.S.
Microkernel provides minimal services of process and m/m mgmt. The commn b/w the client program/ application and services running in user space is established through message passing. They never interact directly. It reduces the speed of execution of microkernel.

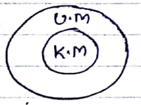
*. If any user service fails it does not effect kernel services hence O.S remains uneffected.
* Extendable. New service can be added to user space



System Calls -

Kernel Mode → when CPU is in kernel mode, the code is being executed can access any m/m address and any h/w resource. If a program crashes in kernel mode, the entire system will be halted.

User Mode → In this mode, the programs don't have direct access to memory and h/w resources. If any program crashes, only that particular program is halted. Hence most of the programs in an OS run in user mode.



System Call - When a program in user mode requires access to RAM or a h/w resource, it must ask the kernel to provide access to that resource. This is done via something called a system call.

When a program makes a system call, the mode is switched from user mode to kernel mode. This is called a context switch. Then kernel provides the resource which the program requested.

Generally system Calls made by the user level programs in following situations:

→ Creating, opening, closing and deleting files in the file system.
→ Creating and managing new processes.
→ Requesting access to a h/w device, like a mouse or a printer etc.

UNIX system, there are around 300 system calls.
Some imp. are –
fork() – To create a child process by parent process.
Exec() – It replaces the address space, text segment, data segment etc. of the current process with the new process.
→ The parent & child process has diff. address space.

Types of System Calls – Page No. 66

① – Process Control. ← End, Abort, load, execute / Create, terminate / Allocate and free m/m.
② – File mgmt. ← Create file, delete file / open, close. / Read, Write, reposition.
③ – Device mgmt. ← request device, release device / read, write, reposition. / logically attach or detach devices.

④ – Infⁿ Maintenance ← Get time or date, set date or time. / Get process, file, or device attributes.

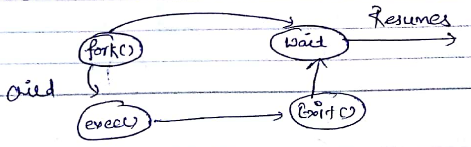
⑤ – Communications ← Create, delete commⁿ connection. / send, Recieve msg. / Transfer status infⁿ.

Process Creation –

Process may create other processes through appropriate system calls, such as fork or spawn.
The process which does the creating is termed the parent of the other processes, which is termed its child. Each process is given an integer identifier, termed its process identifier PID.

Resource Sharing –
→ Parent process and children share all resources.
→ Children share subset of parent's resources.
→

Execution –
→ Parent and children execute concurrently.
→ Parent waits until children terminate.



Address Space - → Child & parent share same address space.

→ fork() calls create new process.
→ exec() system call used after a fork to replace the process' memory space with a new program.

Process Termination -

Process executes last statement and asks the O.S to delete it (exit)

→ O/P data from child to parent
→ Process resources are deallocated by OS.
Resource deallocation -
→ Parent may terminate execution of children processes (abort)

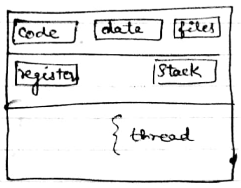
→ If parent is exiting -

Some O.S do not allow child to continue if its parent terminates.

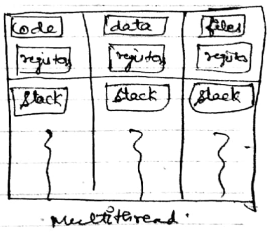
ex - The parent (the command that you typed) - "find name somefile - print" kicks off subprocesses that actually go out and search for the file that you specified. When they are done they report back to the find command.

Thread - A thread sometimes called a "lightweight process (LWP), is a basic unit of CPU utilization; it comparises a thread ID, a program counter, a register set and a stack.

A traditional (or heivyweight) process has a single thread of control. If the process has multiple threads of control, it can do more than one task at a time.



Single thread



multithread.

Benefits - 4 benefits -

1. Responsiveness - Multithreading an interactive application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.

2. Resource Sharing - By default, threads share the m/m and the resources of the process to which they belong, within the same address space.

3. Economy - Allocating m/m and resources for process creation is costly. Creating a process is about 30 times slower than is creating a another ~~process~~ thread and context switching is about five times slower.

4. Utilization of multiprocessor architectures - In single-processor architecture, the CPU generally moves b/w each thread so quickly as to create an illusion of parallelism, but in reality only one thread is running at a time.

Process vs. Thread -

	Process	Thread
Definition	An executing instance of a program is called a process.	Thread is a subset of the process.
Process	It has its own copy of the data segment of the parent process.	It has direct access to the data segment of its process.
Comm^n	Processes must use inter-process comm^n to communicate with sibling processes.	Threads ~~are~~ can directly communicate with other threads of its process
Overheads	More overhead	Low or No overhead
Memory	Run in separate m/m space	Run in shared m/m space.
Dependence	Not Dependent	Dependent.
Controlled By	Controlled by O.S	Controlled by programmes in a program.

Reid Taylor

Types of Thread → There are two types of threads-
Level

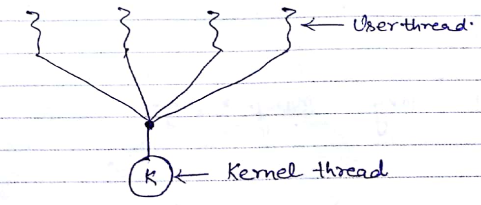
①- User Thread – User threads are supported above the Kernel and are implemented by a thread library at the ~~tast~~ user level. The library provides support for thread creation, scheduling, and mgmt. with no support for ~~thread~~. from kernel.
It is fast to create and manage.
Drawback – If the Kernel is single-threaded, then any user-level thread performing block system call will cause the entire process to block, even if other threads are available to run within the application.
User-thread libraries – POSIX Pthreads, UI-threads.

②- Kernel level thread → They are supported directly by the O.S, the kernel performs thread creation, scheduling and mgmt. in Kernel space. Kernel thread are slower than user level. In kernel thread if any thread performing blocking system call then no appli will be hanged and other threads are available.
ex- Windows NT, Windows 2000.

BOND WITH THE BEST

Multithreading Models -

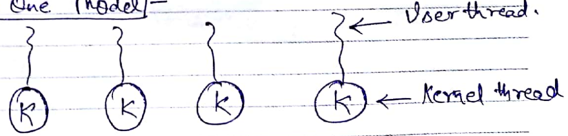
①- Many to One Model -



Many to One model.

It maps many user level threads to one kernel thread. Thread mgmt. is done in user space, so it is efficient; but entire process may block.

②- One-to-One Model -

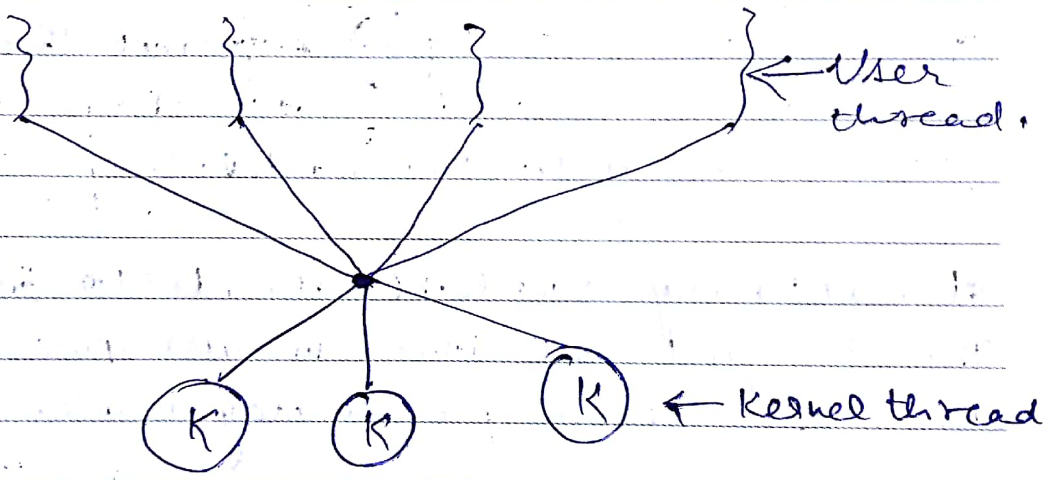


Maps each user thread to a kernel thread. It provides more concurrency than the many-to-one model by allowing another thread to run

Reid & Taylor

when a thread makes a blocking a system call.
Drawback - For each User thread, kernel thread is created which create burden the performance of an application.

③- Many to Many Model -


User thread.
K
K
K
Kernel thread