Operating system Services

Services to programs and to the user of those programs

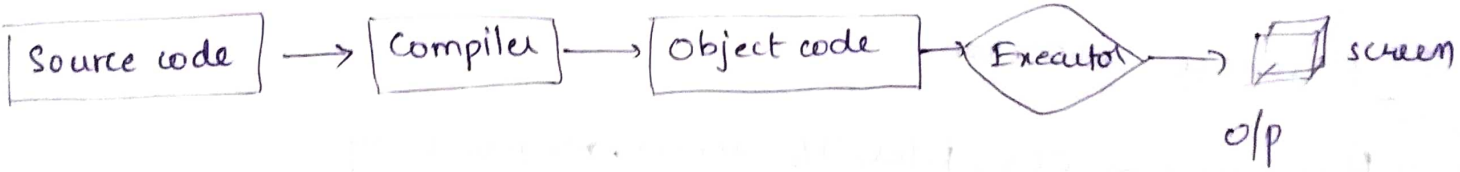
1) **user interface**

↳ allows the user to Interact with the OS

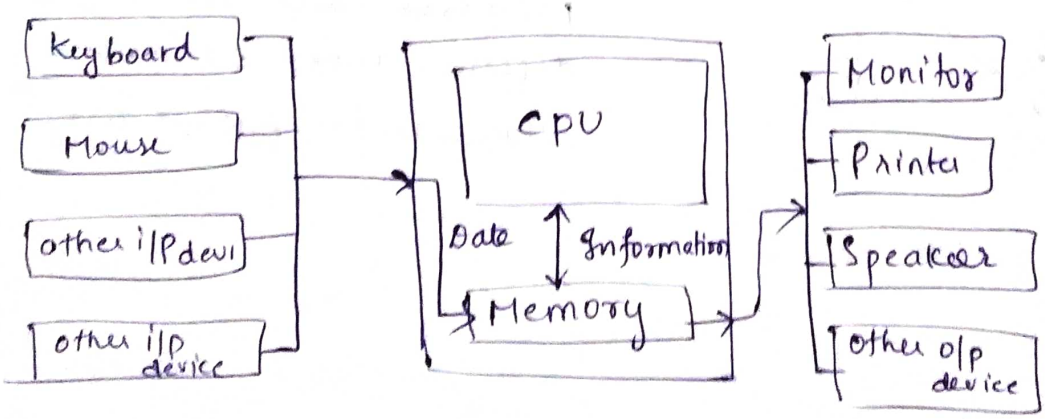
- Command line interface (CLI)
  - e.g:- Command prompt (windows)
  - Terminal (Ubuntu based)
- Graphical user interface (GUI)
  - ↳ user friendly

2) **program Execution**

loading program into the memory & running



3) **I/o operations:-**



4) **file system management:**

Create, search, delete, manipulate files & directories and access restriction to files all are controlled by OS

5) Communications:

Some times processes need to communicate, i.e synchronisation should be done to execute in an efficient way

that may be processes on same Computer or different Computer connected through network

⟶ This all are controlled by the OS

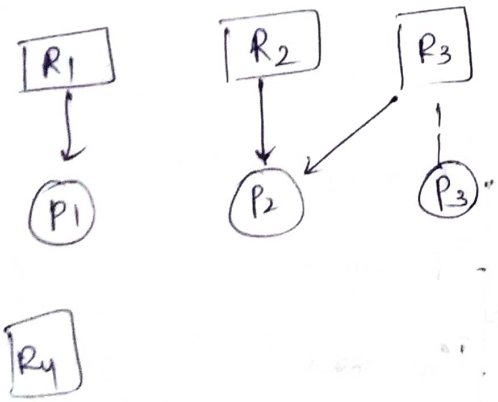
6) Error detection:-

Os must have a way in which it handles different kinds of errors so that your computing is consistent and carried on even it encounters errors

eg: debugging

7) Resource Allocation:-

Resources – CPU, files, I/O devices, main memory --



processes need resources at different times Os must allocate in an efficient way

8) Accounting

Which user use how much & what what kind of Computer resources, the statistics of this usage is valuable for researchers who wish to reconfigure System or to improve the computing services

9) protection & security:-

→ protection of data

Access to the System resources must be controlled

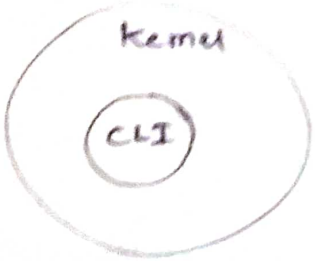
→ System should not be accessed by outsiders

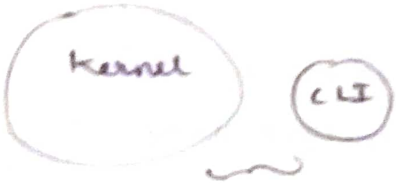
• authentication

User os interface

extension of point ① in services (user interface)

In some OS , In some others like Windows XP, Unix





CLI is treated as special program

→ On systems, multiple command interpreters exists

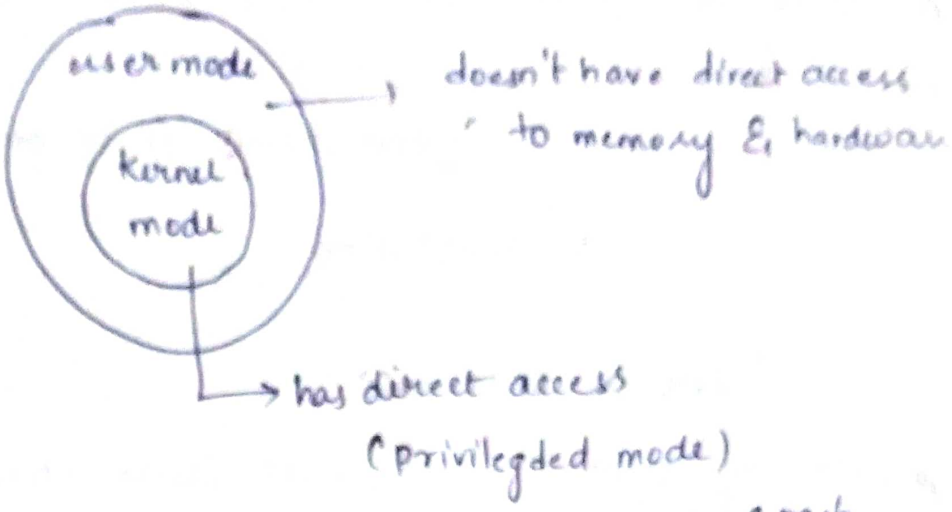
shells

eg:- Bourne shell
Bourne-Again shell (BASH) git-bash
Korn shell
C shell

if you enter command the program behind the command runs and work is done

System calls

System calls provides interfaces for the services of OS



→ If a program runs in user mode if program crash it won't effect the hardware whereas kernal mode does

user mode → safer mode
Kernel mode → privileged mode

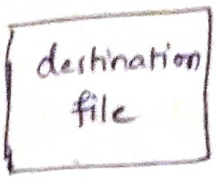
most of the programs run in user mode but when the program needs access to the memory or hardware it makes a System call so that it can shift from user mode to kernel mode

Context switching

↳ during shifting/ switching blw these modes a program is needed SYSTEM CALL

→ These calls are generally available as routines written in C and C++

ex: Task: Copy contents from source file to destination file



Acquire input filename

write prompt to screen

Accept i/p

Acquire o/p filename

write prompt to screen

Accept i/p

open i/p file

If file doesn't exist, ABORT

create o/p file

If exists, ABORT

Read from i/p file

write to o/p file

close o/p file

write completion msg on screen

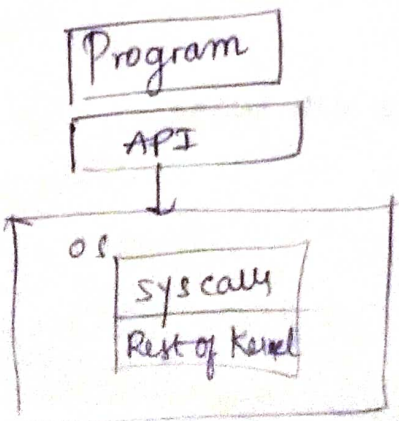
Terminate normally

loop until Read fails

System calls

each micro sequence is related to either memory or Hardware that's why they became a system call

System calls typically not accessed directly by programs, instead they use high-level API (application user interface)



Types of system calls

5 Categories

1) process control

2) File manupulation

3) Device management

4) Infolmation maintainance

5) Communications

6) protection

process control:- System calls are used for controlling of proces running in our system

- end, abort ⟶ terminated due to errors
  - ⟶ terminated normally
- load, execute ⟶ execute a process
  - ⟶ load a process
- Create process, terminate process
- get process attribute, Set process attributes
- wait for time (Scheduling Algos)
- wait sievent, signal event
  - S = S+1
  - S = S+1
- allocate and free memory.

System calls under proces

File manipulation of Management:

- Create, delete file
- open, close
- read, write, reposition
- get file attributes, Set file attributes

System calls groupe under file manipula

Device Manipulation / Management:

↳ I/o devices

- Request device, release device
- Read, write, reposition
- get device attributes, set device attributes
- logically attach or detach devices
  ↳ not physically through wires
  eg:- when pendrive is plugged in by clicking remove/eject device we can remove it logically though it's physically plugged in

Information Maintainace:-

get time or date , set time or date

get system data , set system data

get process, file, or device attributes

set process, file or device attributes

Communications

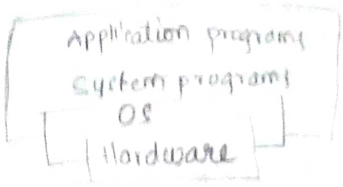
processes need to communicate either same device or different

- create, delete communication connection
  ↳ after completion of communication deleting the connection
- Send, recieve messages
- transfer status information ↳ saying the status of its task
- attach or detach remote devices

protection

- Control access to resources
- Get and set permissions
- Allow and deny user access

System programs: go to the ①ct page



→ System programs will provide a environment for convinient execution and development of program

→ Some are simply UI to system calls others are more complex

1) File management :-

System programs which help for managing of files fall under this category

- Creat
- Delete
- copy
- Rename
- print
- Dump
- List

} manipulating files and directories

2) Status Information:

System programs which deal with status of the system

Ask the system for

- Date, time
- Amount of available memory or disk space
- Number of users
- Detailed performance
- Logging and debugging info

3) file modification:-

- deals with the modification of inner content of the file
  - text editors eg: Notepad, word, vs code --
  - Special commands to search the contents & perform transformations

4) Programming-language Support:-

Os provides system programs to compile any programming language

C c++, Java, python, Visual Basic, PERL)

- Compilers
- Assemblers
- Debuggers
- Interpeters

5) program loading and execution:-

once a program is assembled or compiled, it must be loaded into the memory to get executed

The system may provide

- absolute loaders
- Relocatable loaders
- Overlay loaders
- Linkage editors

→ Debugging systems for machine language or high level language are needed

6) Communications:-

→ Creating virtual connection among processes, users & computer systems.

→ Sending messages to another screen

→ To Browse webpages

→ To Send electronic mail messages

→ To transfer files from one system to another

apart from system programs other programs are provided to perform some common problems or common operations called

7) "Application programs"

↳ web browsers, word processors, Spreadsheets, Database systems, Games

firefox, chrome, edge — MS word — Mserced — mysQL, oracle — purble palace

Operating System Design and Implementation

Designing

Defining goals & specification

↳ This is very difficult task to satisfy all requirements & specifications

Requirements
- choice of hardware
- Type of system (eg: multiprocessing system, multitasking system, time sharing system)

↳ Beyond these, requirements become harder to specify

Requirements from user — User Goals
- convinient to use, Easy to learn & use, Reliable, Safe & fast

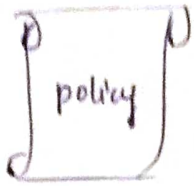
Requirements from System — System Goals
↳ developer goals
- Easy to design, implement, maintain, operate
- flexible, reliable, error free, efficient

not complete but some principle are used to make efficient OS

Mechanisms and policies:-

- How to do Something
  - eg:- How you can maintain that speed is mechanism
- what will be done
  - eg:- dont exceed speed of 50 km/hr

☞ It is always good to seperate your policy from mechanism



Good & flexible

Not good

Car — policy on ①st road
40 km/hr max speed

policy on ②nd road
50 km/hr max speed

mechanism should remain same

mechanism should also change when speed varies

policy → Whether to allocate that resource to process or not

Mechanism → How of allocation ⇒ always remains same

Implementation:

Traditionally (previously) OS implementation is done using assembly language, but now it's implemented using high level language like C, C++

Advantages of high-level language over assembly language: (difficult)

→ The code can be written faster

→ It is more compact

→ It is easier to understand & debug.

→ It is easier to portable

↓ eg: MS-DOS was written in Intel 8088 assembly language so it's available only on Intel family of CPU's

• where as LINUX is mostly written in C and is available on different

CPUs, including Intel 80x86, Motorola 680x0, SPARC, MIPS Pro

Structures of OS:

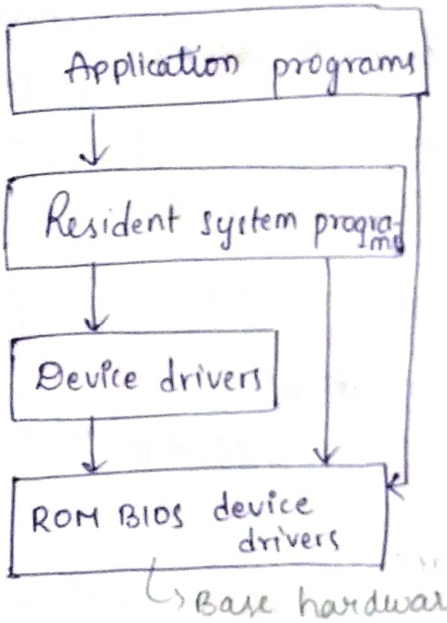
Simple structure:

Used in the beginning and in older OS's

→ not very well defined structure

MOS-DOS followed this structure (written on Intel8088

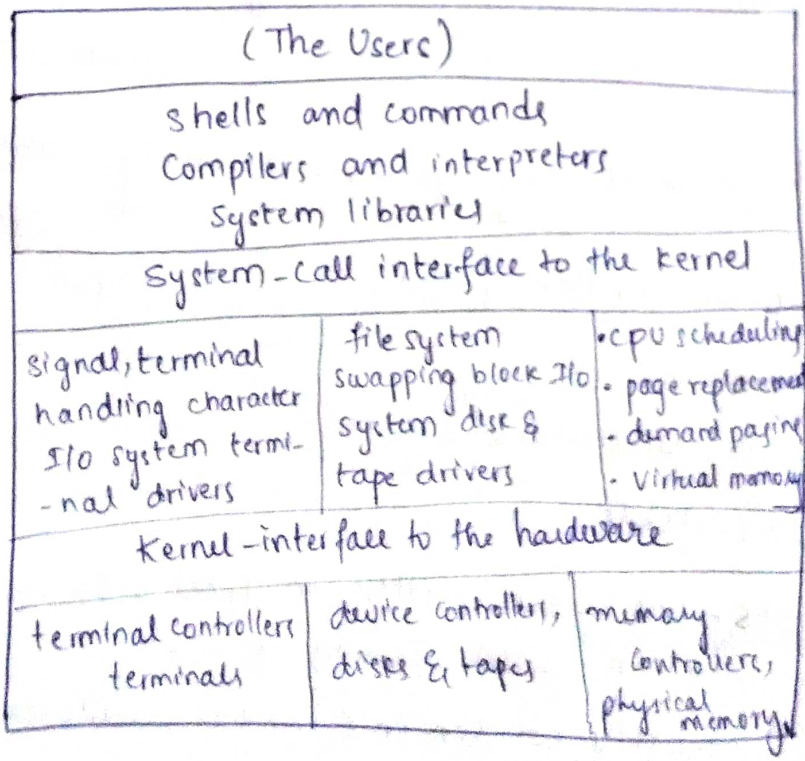
it doesn't provide hardware protection



every layer can acce
basic hardware direct
so in case of any fa
System will crash

• Not well protected, struct

Monolithic structure: (used in UNIX based OS)



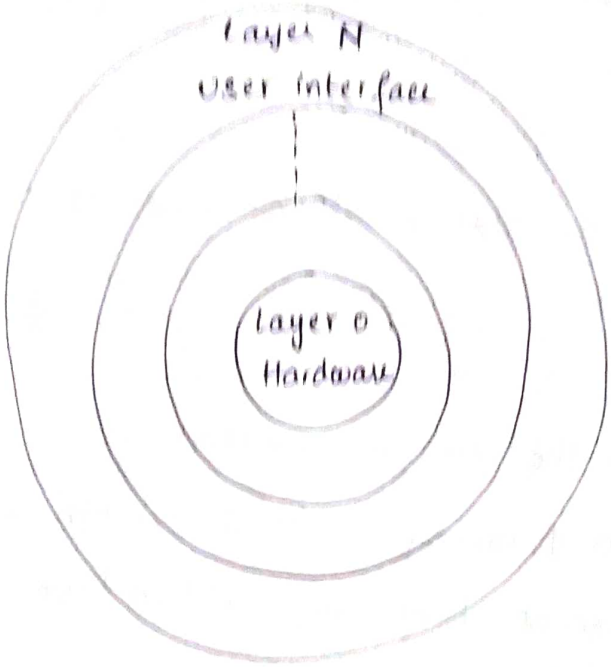
every thing is packed into 1 level so it is called monoly structure

making implem
tation & maint
Very diff

→ if we want to change 1 thing entire level has to be changed

"I"

Layered structure:



→ Hardware is highly protected through layers

Broken down the functionalities into different layers, making easy to implement & debug

→ If any error debug only that layer

→ only below layer services are available to above layer, which is very difficult we have to be more careful while designing

→ It has to dig through deeper into the layers to reach its destination

→ time taking

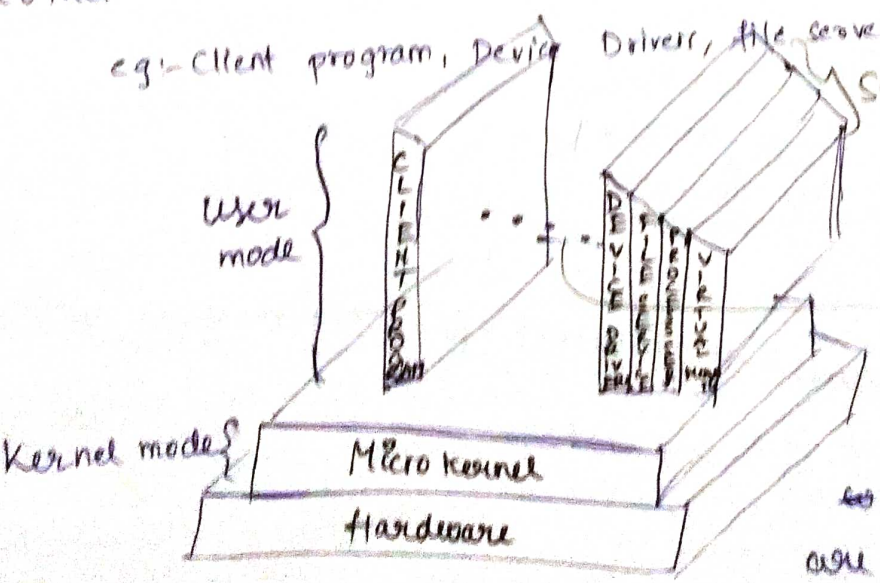
Micro kernels:

Splitting of kernel from monolythic structure

Micro kernel → takes all core responsibilities of kernel

all other works are divided as user level & system level programs

eg:- Client program, Device Drivers, file server, process servers, System program, virtual memory.



→ Message passing

Advantage:- most of the basic functionalities are available in user mode

if program fails in user mode, system won't crash, where as in kernel mode system may crash

disadvantage:-

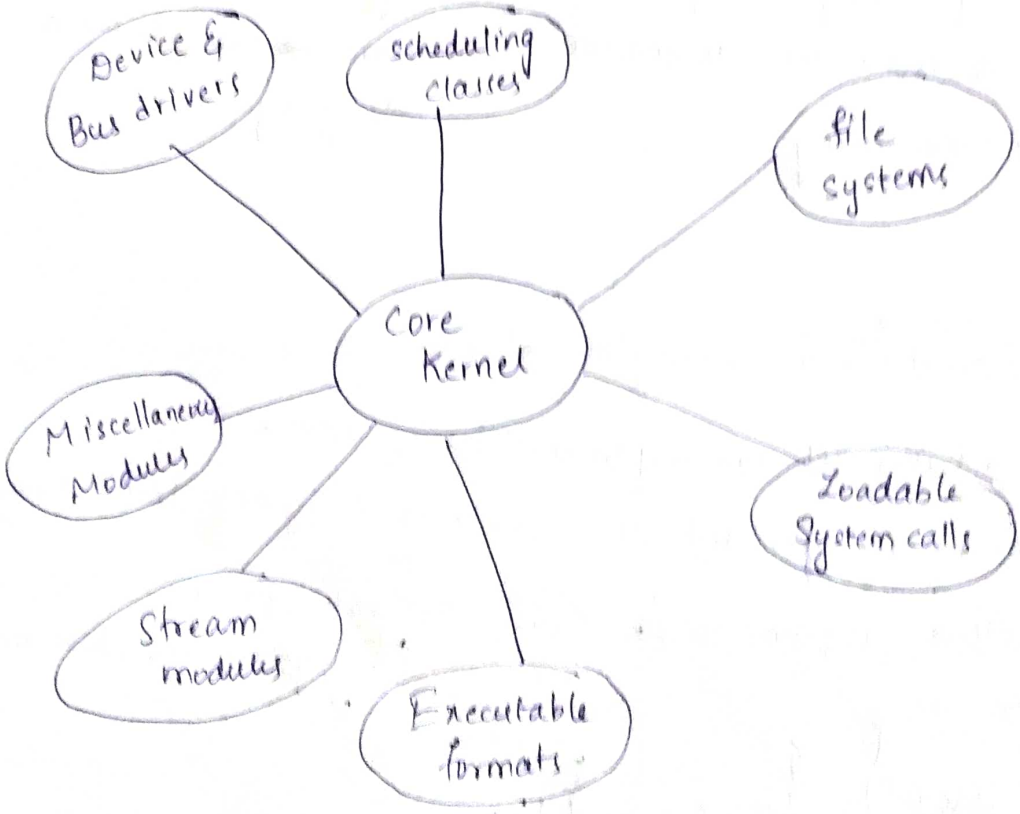
there will be a system overhead due to decreased performance

Modules:-

Best current methodology for os design which involves oops to create modular kernel

core kernel – will only have the core functionalities

other functionalities are in form of modules which will be loaded into the kernel either at boot time or run time



Layered approach + microkernel approach = Modular approach

↓ Better.

Operating System Generation & system Boot

Designing OS

- for one machine at one site
- for any of a class of machines at a variety of sites, with a variety of peripheral configurations (efficient)

SYSGEN — Helps in generating OS in such a way that it System generation is compatible or work for that specific machine hardware

SYSGEN must know this ⟶ depending on these factors OS is generated

- What CPU is used?
- How much memory is Available?
- what OS options are desired? → what kind of fxns system want
- what devices are available? → devices connected to machine

System Boot

→ The procedure of loading the kernel while starting PC is known as booting the system

managed by Bootstrap loader / program

↓ it locates the kernel & load it into memory

it is in the form of ROM

Firmware

↳ kind of ROM in small devices

(Boot strap loader + OS) Same address

in this if any thing wants to be updated entire firmware has to be updated

EEPROM is meant for Read only but by explicitly giving a command contents can also be edited

to make this comfortable

full bootstrap program is loaded

↓

It traverses the file system to find kernel

↓

if found loads it into memory

↓

Execution starts (System is Running now :))