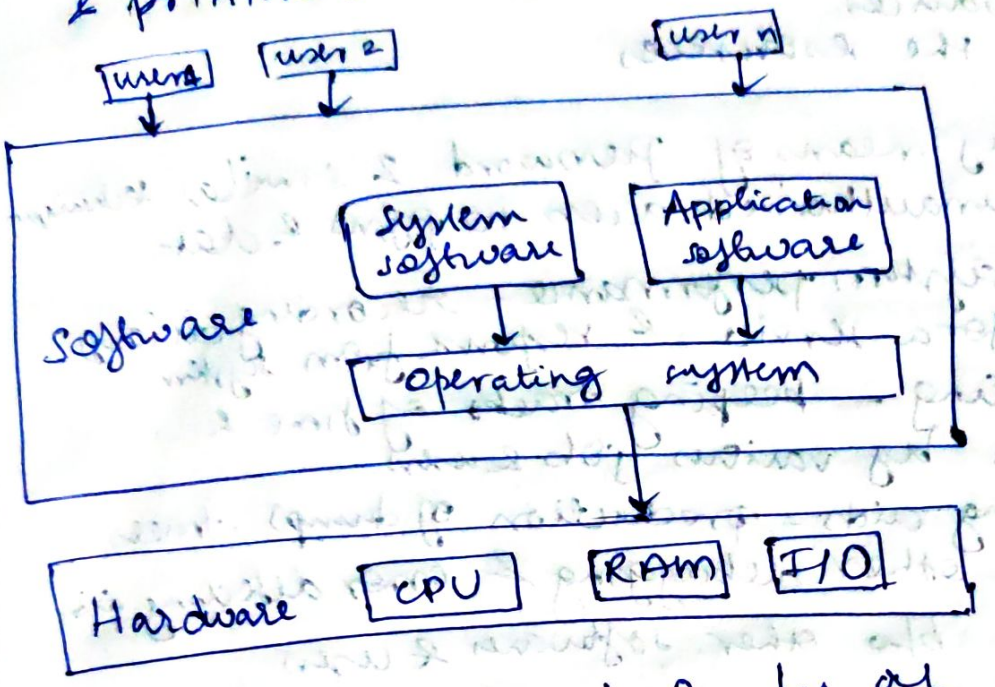
1. An OS is an interface b/w a computer user and computer hardware. An OS is a software which performs all the basic tasks, like file mgmt, memory mgt, process mgt, handling I/O & controlling peripheral devices such as disk drives & printers.

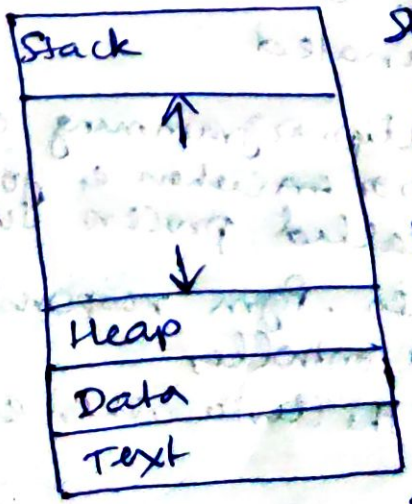


Following are some imp fns of OS

- memory mgt - refers to mgt of 1° memory or main memory. main memory provides a fast storage that can be accessed directly by CPU. For a pgm to be executed it must be in main memory. OS does following activities for memory mgt
  + keeps tracks of 1° mem. ie what part of it are in use by whom, what part are not in use
  + In multiprogramming the OS decides which process will get mem. when & how much
  + allocates mem when a process requests it to do so
  + de-allocates the mem. when a process no longer needs or has been terminated
- Processor mgt - in multiprogramming envt, the OS decides which gets processor when & for how much time. This fn is called process scheduling.
  + keeps track of all devices. Pgm responsible for this task is known as I/O controller
  + decides which process gets the device when & for how much time
  + allocates device in efficient way

- File mgt - a file system is normally organised into directories for easy navigation & usage. Following activities for file mgt.
  - keeps tracks of info, location, uses, status etc. The collective facilities are often known as file system.
  - decides who gets resources
  - allocates resources
  - de-allocates the resources
- Security - by means of password & similar techniques it prevents unauthorized access to pgms & data
- Control over system performance - recording delays b/w request for a service & response from system
- Job accounting - keeping tracks of time & resources used by various jobs & users
- Error detecting aids - production of dumps, traces, error msgs & other debugging & error detecting aids
- Coordination b/w other softwares & users

2. A process is a pgm in execution. The execution of a process must progress in a sequential fashion. We write pgms in text file & when we execute this pgm, it becomes a process which performs all tasks mentioned in pgm. When a pgm is loaded into mem and it becomes a process, it can be divided into 4 sections - stack, heap, text & data.



Stack - process stack contains the temporary data such as method/fn parameters, return address & local variables

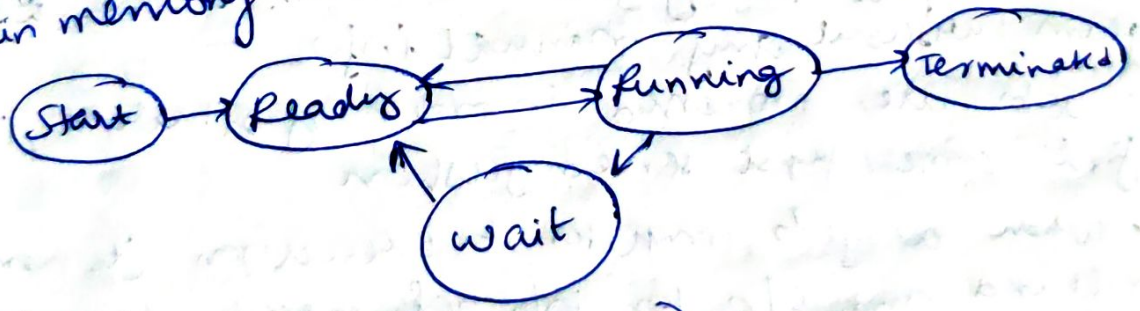
Heap → this is dynamically mem to a process during its run time

Text → this includes the current activity represented by value of PC & contents of processor's registers

Data → this section contains the global & static variables

When a procen executes, it passes through diff states. These stages may differ in diff OS.

1. Start - initial state when a procen is first started/created
2. Ready - procen is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by OS so they can run. Procen may come into this state after state start or while running it by but interrupted by scheduler to assign CPU to some other procen.
3. Running - Once the procen has been assigned to a processor by OS scheduler, the procen state is set to running & processor executes its instr
4. Waiting - procen moves into waiting state if it needs to wait for a resoura, such as waiting for user i/p or waiting for a file to become available
5. Terminated or exit - once the procen finishes its execution or it is termined by OS it is moved to terminated state where it waits to be removed from main memory



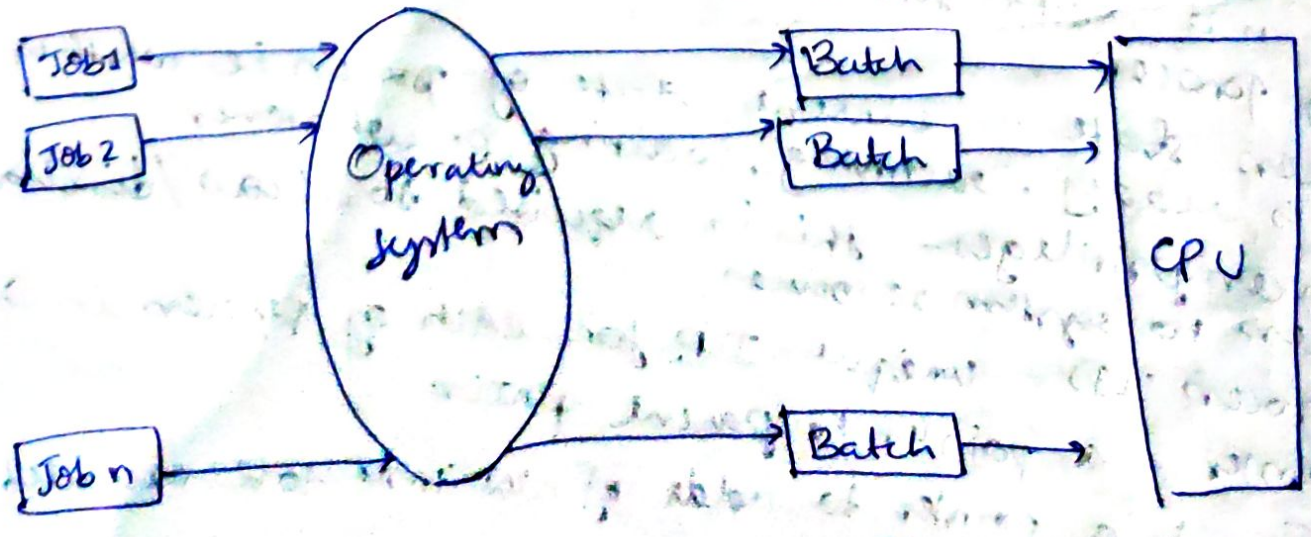
Procen Control Block (PCB)

PCB is a data structure maintained by OS for every procen

1. Procen state - current state of procen ie whether it is ready, running, waiting or whatever
2. Procen privileges - this is required to allow/disallo access to system resources
3. Procen ID - unique ID for each of procen in O.
4. Pointer - a pointer to parent procen
5. PC - its a pointer to addr of next instr to be executed for this procen

6. Register - various CPU register where proces needs to be stored for execution for running state
7. CPU scheduling info - proces priority & other scheduling info which is req to schedule proces
8. mem. mgt info - includes info of page table, mem limits, segment table depending on mem by OS
9. Accounting info - includes amt of CPU used for proces execution, time limits, execution ID etc
10. IO status info - this includes a list of I/O devices allocated to proces

3. Batch processing is a technique in which an OS collects the programs & data together in a batch before processing starts. An OS does following activities related to batch processing.
   - OS defines which has predefined sequence of commands, pgms & data as a single unit.
   - OS keeps a no of jobs in mem. & executes them without any manual info.
   - Jobs are processed in order of situation, i.e first come first served fashion.
   - When a job completes its execution, its mem. is released and O/p for job gets copied into an O/p spool for later processing or printing.



Advantages:
- Batch processing takes much of work of operator to computer
- Increased performance as a new job get started as soon as previous job is finished, without any manual intervention

Disadvantages
- Difficult to debug program
- A job could enter an infinite loop
- Due to lack of protection scheme, one batch job can affect pending jobs

4. Application software is capable of dealing with user inputs & helps the user to complete the task. It resides above system software.

Types of application software

1. Presentation software - Presentation pgm is a pgm to show the info in the form of slides. We can add text, graphics video and images to slides to make them more informative.
   software has 3 components
   - Text editor for inputting & formatting text
   - Inserting graphics, text, video & other multimedia files
   - Slideshow to display the info

   Presentation software helps the presenter to present their ideas with ease & visual info easy to understand

   Eg :- PowerPoint, Keynote

2. Spread sheet software - Spreadsheet software is used to perform manipulate and calculations. In spreadsheet software data is stored in intersection row & column. The intersection of row & column is known as cell. The cell labelled with row & column like A1, A2 etc while entering data into cell, we can also also define the data value like text, date, time, no. It provides many formula & fn to perform calculations like arithmetic operations,

logical operations, text operation etc. It provides charts, graphs to display data graphically.
Eg:- Excel

3. Data base software - database is a collection of data related to any application. Every application has some database where data regarding users stored. For this purpose, we used database software. Whenever operate the application data is accessed from the database & other manipulation, it gets back stored in database

4. Multimedia software - multimedia is a combination of text, graphics, audio & multimedia software used in editing of video, audio & text. Multimedia software are used in growth of business, education, info, remote system & entertainment

5. Simulation software - simulation is an imitation of real world & environment. The simulation creates a physical environment of real world to represent the similar behaviour, In and key nature of selected topic. Simulation is technology for education, engineering, testing, training, video games & for scientific modelling of natural systems to gain insight into their functioning. The simulation used in area of real world where the real system can't be accessible or may be dangerous or unacceptable.

6. Word processing software - its used to manipulate, format the text, to create memos, letters, documents. Processing software is used to format & beautify text It provides a list of features.

Q.

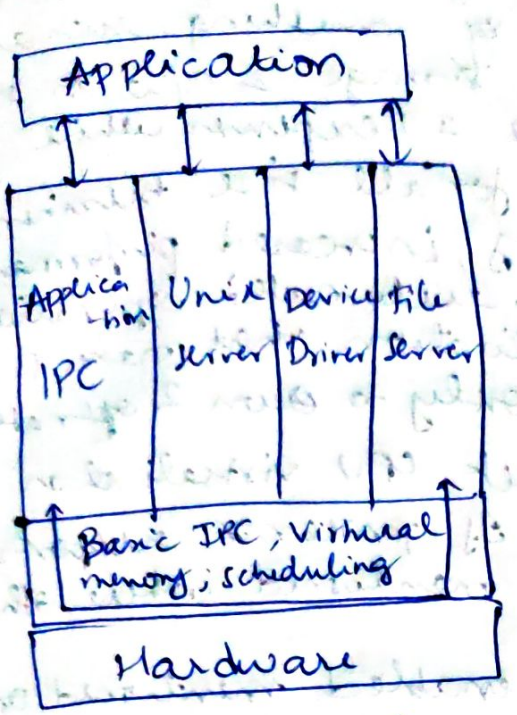


Fig. microkernel

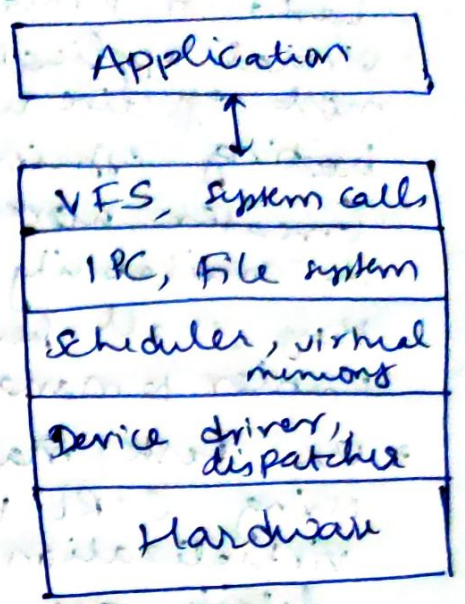


Fig: Monolithic kernel

	Microkernel	Monolithic
1. Basic	In kernel, user services & kernel services are kept in separate addr space	In monolithic kernel, both user services & kernel services are kept in same addr space
2. Size	Microkernel are smaller in size	Monolithic kernel is larger than microkernel
3. Execution	Slow execution	Fast execution
4. Extensible	Microkernel is easily extensible	monolithic kernel is hard to extend
5. Security	If a service crashes, it does effect on working of microkernel.	If a service crashes, the whole system crashes in monolithic kernel
6. Code	To write a microkernel more code is required	To write a monolithic kernel, less code is required
7. Example	QNX, symbian, L4Linux, HURD, Coyotos	Linux, BSDs, Solaris, AIX, HP-UX, DOS

9. Virtualization is the process of creating a software based or virtual representation of something, such as virtual applications, servers, storage & n/w. It is the most effective way to reduce IT expenses while boosting efficiency & agility for all size businesses. Greater workload mobility, increased performance & availability of resources, automated operations, they're all benefit of virtualization that make IT simpler to manage & less costly to own & operate.

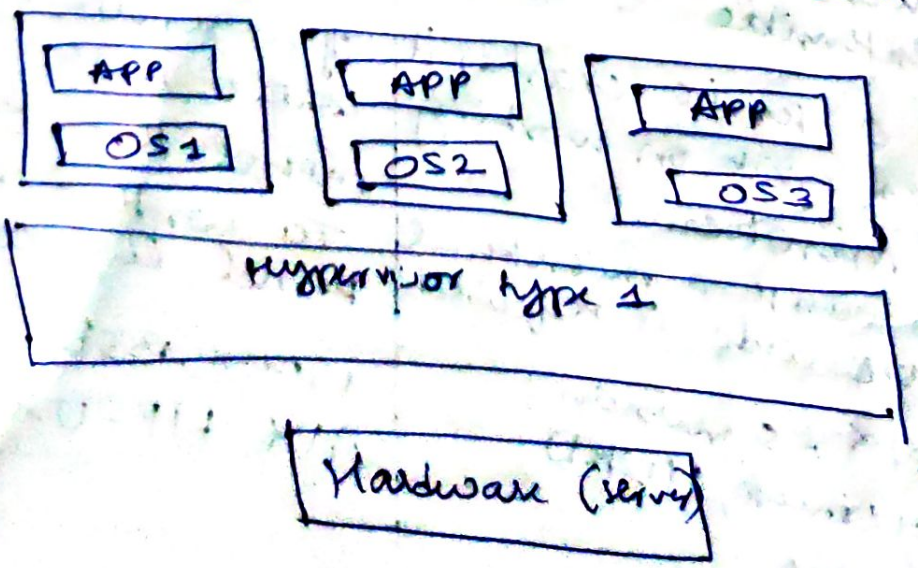
- Use task manager to check CPU virtualization using CTRL + Shift + Esc. If processor support virtualization, it will be mentioned where other details are shown.
- Under performance tab, enabled mentioned or else disabled. And if processor doesn't support virtualization technology, Virtualization & Hyper-V mentioned anywhere at all

10.

Native or Bare metal hypervisor

They are softwares that run directly on the host's hardware to control the hardware & to monitor the Guest OS. The guest OS runs on a separate level above the hypervisor. All of them have virtual machine manager

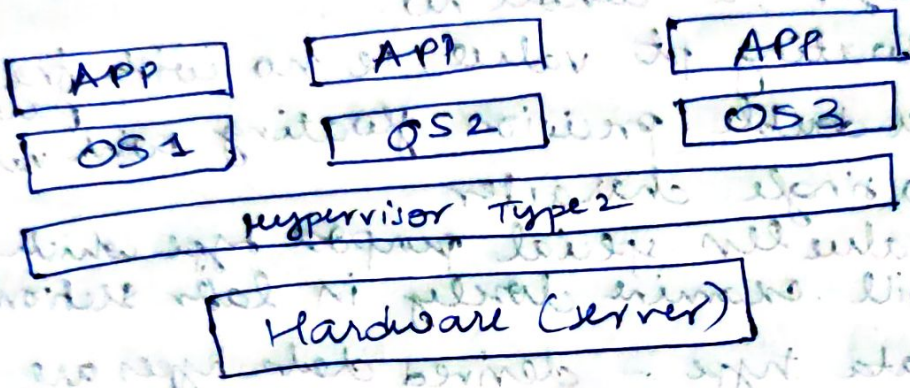
Eg of virtual machine architecture are Oracle VM, Microsoft Hyper-V, VMWare ESX & Xen



Hosted supervisor

Hosted supervisor are designed to run within a traditional OS. In other words, a hosted supervisor adds a distinct software layer on top of the host OS. While, the guest OS becomes a third software level above hardware.

Eg:- Vm virtual Box, vmware server & workstation, KVM, QEMU, parallels



11. Types of ~~virtualization~~ virtualization

1. server virtualization - It is visualizing your server infracture where you do not have to use any more physical servers for different purposes.

2. Client & desktop virtualization - similar to server virtualization, but this time is on the user's site where you virtualize their desktops & by utilizing the datacentre resources we change their desktops with thin clients.

3. Services & Applications Virtualization

The virtualization technology isolates applications from the underlying OS and from other applications, in order to increase compatibility & managability. Eg- docker can be used for that purpose

4. Network virtualization

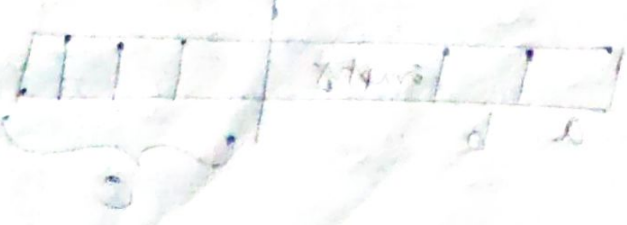
It is part of virtualization infracture, which is used especially if you are going to visualize your servers. It helps you in creating multiple switching, Vlans, NAT-ing etc

5. Storage visualization
This is widely used in data centres where you have a big storage & it helps you to create, delete, allocated storage to different hardware. This allocation is done through n/w connection. The leader on storage is SAN.

12. Data types in C
1. Primary data type
- int - integer, a whole no.
- float - floating pt value, ie no with fractional part.
- double - a double precision floating point value.
- char - a single character
- void - valueless special purpose type which we will examine closely in later sections.

2. Derived data type - derived data types are nothing but 1° datatypes but a little twisted or grouped together like array, structure, union & pointer
- Array - an array is a collection of data items, all of the same type, accessed using a common name.
- Structure - a structure in C is a collection of items of diff types
- Union - A union is a special data type available in C that allows storing different data types in same memory location. You can define a union with many members, but only one member can contain a value at any given time.
- Pointer - a pointer is a variable whose value is address of another variable ie direct address of memory location. Like any variable or constant, you must declare a pointer before using it to store any variable address

```
#include <stdio.h>
#include <conio.h>
main()
{ clrscr();
  printf("short int is %2d bytes \n", sizeof(short int));
  printf("int is %2d bytes \n", sizeof(int));
  printf("int* is %2d bytes \n", sizeof(int*));
  printf("long int is %2d bytes \n", sizeof(long int));
  printf("long int* is %2d bytes \n", sizeof(long int*));
  printf("signed int is %2d bytes \n", sizeof(signed int));
  printf("unsigned int is %2d bytes \n", sizeof(unsigned
                                                  int));
  printf("float is %2d bytes \n", sizeof(float));
  printf("float* is %2d bytes \n", sizeof(float*));
  printf("double is %2d bytes \n", sizeof(double));
  printf("double* is %2d bytes \n", sizeof(double*));
  printf("long double is %2d bytes \n", sizeof(longdouble));
  printf("signed char is %2d bytes \n", sizeof(signed
                                                  char));
  printf("char is %2d bytes \n", sizeof(char));
  printf("char* is %2d bytes \n", sizeof(char*));
  printf("unsigned char is %2d bytes \n",
                                  sizeof(unsigned char));
  getch();
}
```

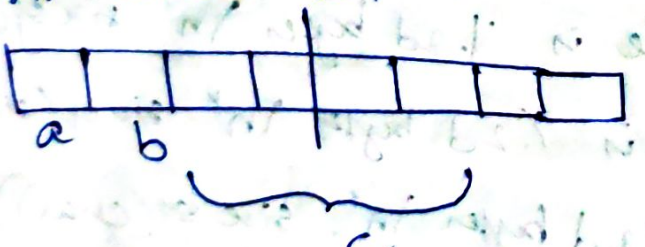


13. In order to align the data in memory, one or more empty bytes (addresses) are inserted (or left empty) b/w memory addresses which are allocated for other structure members while memory allocation This is padding.

Architecture of a comp. processor is such a way that it can read 1 word (4 byte in 32 bit processor) from memory at a time. To make use of this adv of processor, data are always aligned as 4 bytes package which leads to insert empty addresses b/w other member's address

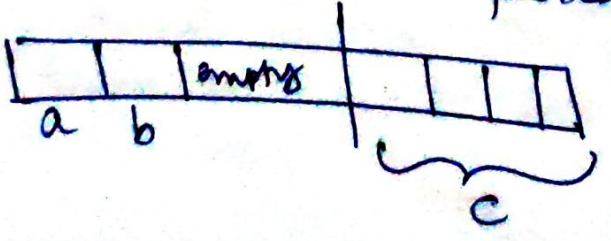
```
Struct student
{
char a; //1 byte
char b; //1 byte
int c; //4 bytes
}
```

If we have a 32 bit processor (4 byte at a time) then the pictorial repsn of memory for above str would be:



Structure occupies the contigous block of memory. Both char a and char b variables can be accessed in one CPU cycle; but we will face the problem when we access the int c variable as 2 CPU cycles are required to access the value of c variable. So in first cycle two bytes and in second cycle, other two bytes are accessed.

To overcome this problem, padding is used



In order to achieve padding, an empty row is created on the left as shown and the two bytes which are occupied by C variable are on the left are shifted to right. So, all the 4 bytes of C variable are on the right. Now the C variable can be accessed in a single CPU cycle. After padding, the total memory occupied by structure is 8 bytes (1 byte + 1 byte + 2 bytes + 4 bytes) which is greater than previous. Although memory is wasted, variable can be accessed within a single cycle.

15. Basis of comparison	Windows	MAC	Linux
1. Basic difference & history	Windows was first released in 1985. It was supposed to be a graphical user interface on top of MS-DOS. All features of MS-DOS were later integrated with Windows 95 release.	This OS is older than windows. It was released in 1984. It began as a graphical user interface right after its inception	It was released in 1991 and designed for GNU developers. GNU developers later integrated it into Linux
2. File Structure	Windows follow a directory structure to store the different kinds of files of user. It has logical drives & cabinet drawers. It also has folders. Some common folders like documents, pictures, music, videos & downloads. All these files can be stored in these folders and also new folders can be created. It also has files which can can be spreadsheet or an application	The file str of MAC is commonly known as MAC OS X. If you go to dig into your MAC's hard disk through finder you will see many directories. The root directory of MAC may encounter when they visit their own MAC book. You can explore file system & directory structure by going to directories like /Application, /Developer, /bin, /temp etc	Linux has a completely different file structure from windows & MAC. It was developed with different code base. It stores data in the form of a tree. There is a single file tree & all your drives are mounted over this tree

	In addition to this windows also provides recycle bin where all deleted files can be stored. Recycle can be configured to ↑ its size		
Registry	Windows registry is a master database which is used to store all settings on your computer. It is responsible to store all user info with its passwords & device relate info. The registry also has an editor which allows you to view all keys & values or even drivers if necessary	MAC stores all application settings in a series of .plist files which have the various preferences folder in MAC. This .plist file contains all properties in either plain text or binary format. These are stored at /Library/Preferences folder	Linux also does have a specific registry of its own. Application setting is stored on per user basis under the different users in same hierarchy format of the files being stored. There is no centralized database for storing these details & periodic cleaning is also required.
Interchangeable Interfaces	Windows interface was not interchangeable until Windows 8. Windows XP had some improvements but not par. Start menu, task bar, system tray & windows explorer	MAC has a facility to bridge virtual n/w interfaces. This can be done by going to system preferences & managing the interfaces.	Linux is easy to switch interfaces. You can switch the env to without having to carry all installations. They are utilities like GNOME & KDE which help in catering to these needs. They help in focussing on different aspects.
Command terminal	A terminal or command prompt is a black box ideally used to execute commands. It is also called the Window Command Processor. It is used to execute commands & different batch files. It can also be used for administrative jobs & trouble shoot & solve all window issues	MAC provides a console as a terminal application. It has a console, command line, prompt & terminal. Command line is used to type your commands. Prompt will provide you with some info & also enable you to run commands. A terminal is an actual interface which will provide the modern graphical user interface as well. You can terminals at Applications -> Utilities	Linux also provides a terminal. You can find terminal at: Applications -> System or Application -> Utilities. In addition to this, there is also a shell prompt. The most common shell used in bash. It defines how the terminal will behave & look when it is run.

19. fgets() reads in at most one less than size characters from stream & stores them into the buffer pointed to by S. Reading stops from an EOF or a new line... A terminating null byte is stored after the last character in the buffer. So it adds '\n' after your 4 letters, returning string-length + 1

Eg:-

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main(void)
{ char name[10] = {0};
  printf("Enter your name \n");
  fgets(name, 10, stdin);
  printf("your name is %s & it is %d letters
                    \n", name, strlen(name));
  name[strcpn(name, "\n")]=0;
  printf("New - your name is %s & it is %d
                    letters \n", name, strlen(name));
  return 0;
}
```

20. • strlen() fn:
used to find the length of a character string

Eg:- int n
char st[20] = "Bangalore";
n = strlen(st);

• strcpy() fn:
strcpy() fn copies contents of one string into another string is given below

Syntax: char * strcpy(char * destination, const char * source);

Eg:- strcpy(str1, str2) - It copies contents of str2 into str1

strcpy(str2, str1) - It copies contents of str1 into str2.

- strcat() fn:
strcat() fn in C language concatenants two given strings. It concatenates source string at the end of destination string. Syntax for strcat fn is given below
Syntax: char *strcat(char* destination, const char * source)

- Strncat() fn:
strncat() fn in C language concatenates (appends) portion of one string at the end of another string.
Syntax: char * strncat(char * destination const char * source, size_t num);
eg: strncat (str2, str1, 3); - first 3 characters of str1 is concatenated at the end of str2.

- Strcmp() fn:
This fn in C compares 2 given strings & returns 0 if they are same. If length of string1 < string2 it returns < 0 value. If length of string1 > string2 it returns > 0 value.
Syntax: int strcmp(const *char * str1, const char * str2);

- strlwr() fn.
converts a given string into lowercase
syntax char * strlwr(char *string);
strlwr() fn is non std fn which may not be available in std library in C

- strupr() fn:
strupr() fn converts a given string into uppercase
Syntax: char * strupr(char * string);
strupr fn is non std fn which may not be available in std library in C.

- strrev() fn:
  strrev() fn reverses a given string in C language
  Syntax: char *strrev(char *string);
  strrev() fn is non std fn which may not be available in std library in C.
  Eg:- char name[20] = "ftf";
  then, strrev(name) = ftf

- atoi() fn:
  It converts string value to numeric value & it converts a numeric-string value to equivalent integer value
  Syntax: int atoi(string);
  Eg: ("o/p = %d", atoi("123") + atoi("234"));
  → 357 will be printed

- atoll() fn:
  It converts a long int string value to equivalent long integer value.
  Syntax: long int atoll(string);
  Eg:- printf("o/p = %d", atoll("486394") - atol("112233"));
  → 374161 will be printed

- atof() fn:
  converts a floating pt text format value to double value
  Syntax: int atoi(string);
  Eg: printf("%f", atof("3.1412")*5*5);

- strchr() fn:
  It returns pointer to first occurance of character in a given string.
  Syntax Eg:- char *strchr(const char *str, int character);
  eg:- char city[20] = "Madras";
  char town[20] = "Mangalore";
  strcmp(city, town);

16. Distributed Operating system is one of the important type of OS.

Multiple central processors are used by distributed systems to serve multiple real-time applications & multiple users. Accordingly, data processing jobs are distributed among processors.

Processors communicate with each other through various comm. lines (like high speed buses or telephone lines). These are known as loosely coupled systems. Processors in this system may vary in size & function. They are refered as sites, nodes, computers & so on.

High speed bus: A mechanism that transfers data b/w components inside a computer.

A DOS involves of autonomous computer system, which are able to comm. with each other through LAN/WAN. This system will provide a virtual machine abstraction to its users & wide sharing of resources like computational capacity & I/O etc.

This system as mentioned, incorporates various autonomous interconnected computers that comm. with each other using a shared comm. n/w, further more they are independent system that possess their own memory unit & CPU.

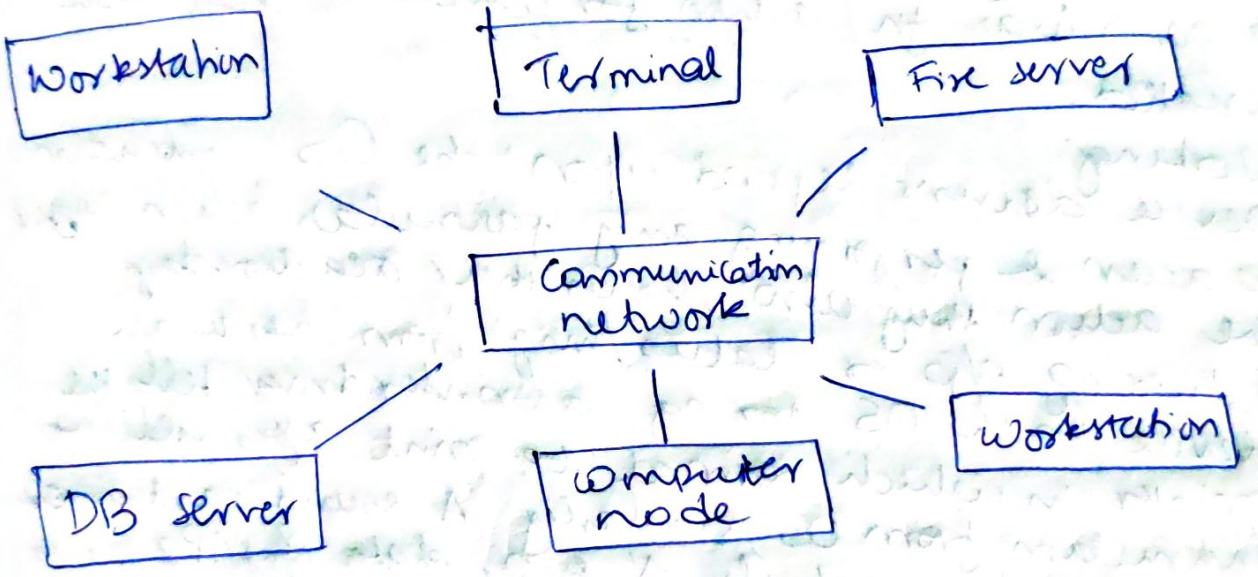
Another term which is used along side-distributed OS is a loosely coupled system. The processors in these systems may differ in size & function.

One of the big advantages of working with DOS is that it is always possible that one user can access the files or software which are not present on another system n/w.

Distributed systems can be considered to be more reliable than a central system because if the system has only one instance of a critical peripheral (component, like CPU, n/w interface, disk & so on) if that one instance fails, the system will go down completely. However when there are multiple instances in the system, like in a distributed OS, then if a component fails, the system may be able to continue to function despite the failure. Distributed systems also allow software failures to be dealt with, rather than stopping the whole system

Architecture of a distributed OS:
In a DOS following occurs
- all software & hardware compounds are located remotely, in order for them to comm. with each other, they pass messages.
- One of the most important aspects of a distributed system is resource sharing. Resources are managed by servers & clients use these resources.

A DOS runs on a no. of independent sites which are connected through a comm. n/w. However it is protrayed to the user that they run their own OS.



5. A multitasking OS allows more than one program to be running at the same time, from the point of view of human time scales. A single-tasking system has only one running program. Multi-tasking can be of 2 types: preemptive & co-operative. In preemptive multitasking, the OS slices the CPU time & dedicates one slot to each of programs. Unix-like OS such as Solaris & Linux support preemptive multitasking as does Amiga OS. Cooperative multitasking is achieved by relying on each process to give to the other processes bit versions of both Windows NT & used preemptive multi-tasking MAC OS prior to OS X used to support cooperative multitasking.

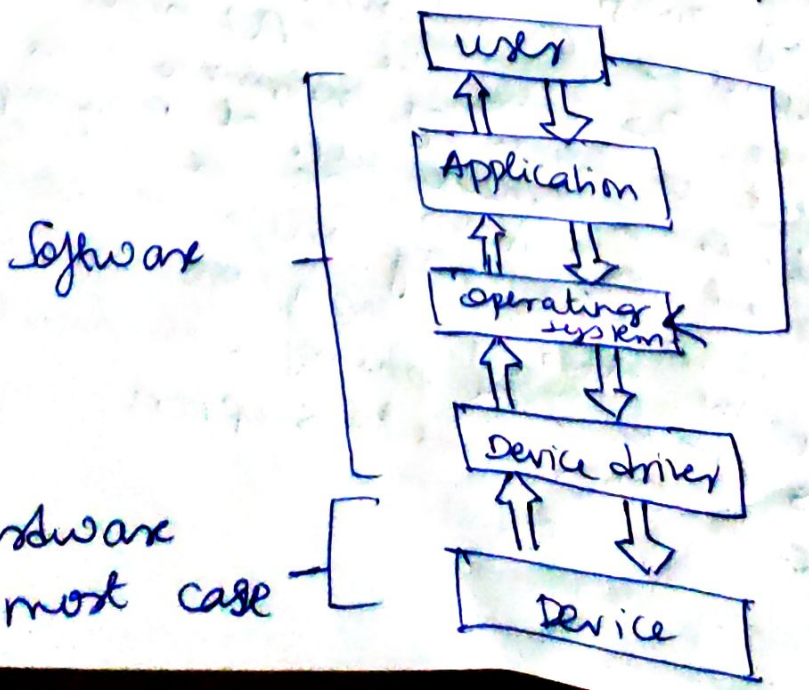
6. Device driver in computing refers to a special kind of software program or a specific type of software application which controls a specific hardware device that enables different hardware devices for comm. with computer's OS

A device driver comm. with computer's hardware by computer subsystems or computer bus connected to hardware.

Device drivers are very essential for a comp system to work properly because without device driver the particular hardware fails to work accordingly means it fails in doing in doing a particular fn / action for which it has been created.

Working:

Device drivers depend upon the OS's instruction to accen & performing any particular action. After the action they also shows their reactions by delivering o/p or status / msg from hardware device to the OS. for eg, a printer driver tells the printer in which format to print after getting instruction from OS. Similarly A sound card driver is there due to which 1's & 0's, data of MP3 file is converted to audio signals & you enjoy the music. Card reader, controller, modem, n/w card, sound card, printer, video card, USB devices, RAM, speakers etc need device drivers to operate.



16. Kernel is central component of an OS that manages operations of computer & hardware. It basically manages operations of memory and CPU time. It is core component of an OS. Kernel acts as a bridge b/w applications & data processing is loaded & remains into memory until OS is shut down again. It is responsible for various tasks such as disk management, task mgt, & memory mgt. It decides which process should be allocated to processor to execute & which process should be kept in main memory to execute. It basically acts as an interface b/w user applications & hardware. The major aim of kernel is to manage comm. b/w software i.e user-level applications & hardware i.e CPU & disk memory.

There are 5 types of kernel
- A micro kernel, which only contains basic functionality
- A monolithic kernel which contains many device drivers
- Hybrid kernel
- Exo kernel
- Nano kernel

17. scanf() works great while taking i/ps such as integer, character, float etc. It falls behind while taking string i/p containing white spaces

```
#include <stdio.h>
int main()
{
  char string[10];
  Printf("Enter string");
  scanf("%s", string);
  return 0;
}
```

scanf() stops scanning as soon as it ~~to~~ encounters white space or newline. This, in fact, makes taking string i/p using scanf() a bit troublesome. This can be easily avoided by using some other i/p fn like gets() & fgets()

- gets() is a pre-defined fn in C which is used to read a string or a text line. And store the i/p in a well-defined string variable. The fn terminates its reading session as soon as it encounters a newline character

Eg:-
```
#include <stdio.h>
int main()
{ char string[10];
  printf("Enter string ");
  gets(string);
  printf("\n%s", string);
  return 0;
}
```

- fgets()
The std C library also provide us with yet another function, the fgets() fn. The fn reads a text line or a string ~~or~~ ~~from~~ the specified file or console. And then stores it to ~~to~~ the respective string variable.
Similar to the gets() fn. fgets also terminates reading whenever it encounters a newline character. But unlike gets() the fn also stops when EOF is reached or even if the string length exceeds the specified limit, n-1

Syntax, fgets(char *str, int n, FILE *stream)
str → variable in which string is going to be stored
n → It is the max length of string that should be read
stream → It is the file handle, from where the string is to be read

Eg:-
```
#include <stdio.h>
int main()
{
  char string[20];
  FILE *fp;
  fp = fopen("file.txt", "r");
  fgets(string, 20, fp);
```

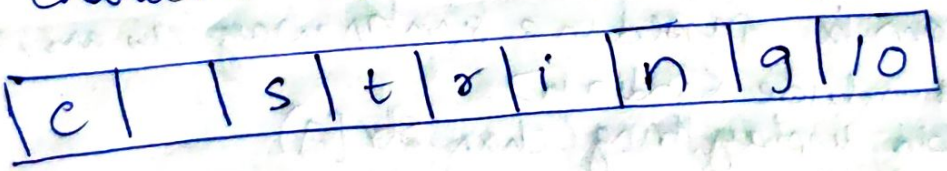
```
    printf("The string is %s", string);
    fclose(fp);
    return 0;
}
```

Even though both the functions gets() & fgets() can be used for reading string i/p The biggest difference b/w the two is the fact that the latter allows the user to specify the buffer size. Hence it is slightly recommended over the gets() fxn.
The gets() fxn doesn't have provision for the case if i/p is larger than buffer. As a result, memory clogging may occur. This is the part where the fgets() fxn comes & provides an ultimate solution

18. In C programming, a string is a sequence of characters terminated with a null character \0.

For eg, char c[] = "c string";

when the compiler encounters a sequence of characters enclosed in double quotation marks, it appends a null character \0 at the end by default



Declaration of string

```
char s[5];
```

Here, we have declared a string of 5 characters

Initialization of string

```
char c[] = "abcd";
char c[50] = "abcd";
char c[] = {'a', 'b', 'c', 'd', '\0'};
char c[5] = {'a', 'b', 'c', 'd', '\0'};
```

Read string from user

Scanf() can be used read a string. scanf() reads sequence of characters until it encounters white space.

We can use fgets() fn to read a line of string. puts() to display string

Eg:-
```
#include <stdio.h>
int main()
{ char name[30];
  printf("Enter name");
  fgets(name, sizeof(name), stdin);
  printf("Name");
  puts(name);
  return 0;
}
```

fgets() used in the example to read a string from user.

Passing strings to fn:

Strings can be passed in a similar way as arrays.

Eg:-
```
#include <stdio.h>
void displaystring(char str[]);
int main()
{ char str[50];
  printf("enter string:");
  fgets(str, sizeof(str), stdin);
  displaystring(str);
  return 0;
}
void displaystring(char str[])
{ printf("String O/p");
  puts(str);
}
```