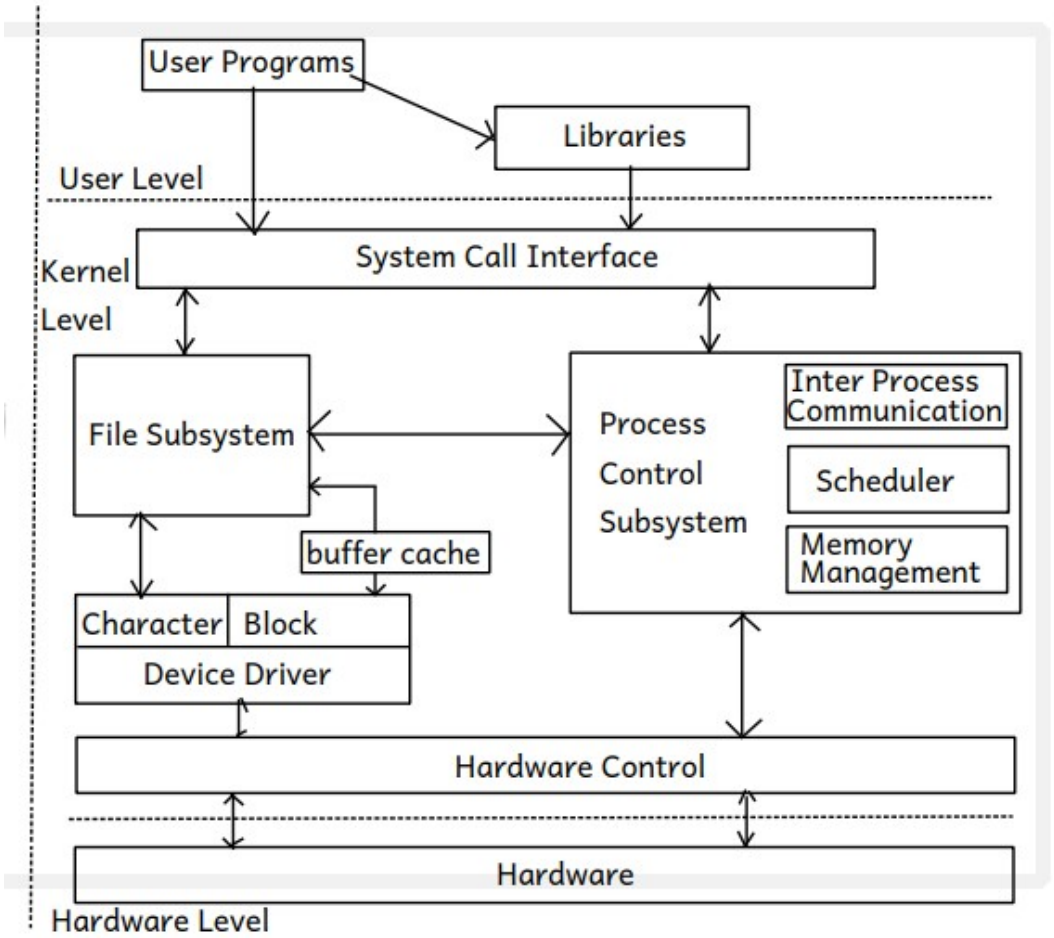
+ "UNIX" System Architecture Design:

- UNIX is an OS.
- UNICS: Uniplexed Information & Computing Services
- UNIX OS was developed at AT&T Bell Labs in the decade of 1970's, by Ken Thompson(B.E. Electricals), Denies Ritchie (M.Sc. Physics & Maths) & Team, in 1969 UNIX first time run on the machine "DEC-PDP-7".
- UNIX OS basic design purpose was for developement
- UNIX OS has beed designed for developers by the developer's.
- UNIX OS is reffered as mother of all modern OS's like Linux, Windows, MAC OSX Android, BSD UNIX etc...., as system arch design of UNIX is followed in all modern operating systems.



- There are two major subsystem's of UNIX OS:
1. File Subsystem
2. Process Control Subsystem

- UNIX point of view "file" & "process" are two very very important concepts
- What is file ? What is Process?
- Whatever that can stored is considered as a "file", whereas whicever is active is considered as a "process".
- "File has Space & Process has Life".
- UNIX OS treats everything as file, i.e. for UNIX keyboard is a file, monitor is a file, hard disk drive is also a file, etc... UNIX os treats all devices also aas

a file.
- UNIX OS categorised devices into two catagories and hence files:
1. "character devices": devices from which data gets transferred char by char reffered as character devices. e.g. KBD, Monitor, Printer, Serial Port, Parellel Port etc...

2. "block devices": devices from which data gets transferred block by block i.e. sector by sector reffered as block devices, e.g. all storage devices

- 7 types of files are there in UNIX/UNIX based OS like Linux:
1. Regular File (-): text file, image file, audio file, vedio file etc...
2. Directory File (d):
3. Block special device file (b): all stoarage devices
4. Character special device file (c)
5. Linkable File (l)
6. Pipe File (p)
7. Socket File (s)

- when we execute any program an OS bydefault opens 3 files for that process
1. "stdin": buffer in a main memory which is associated with standard i/p device KBD, bydefault any program reads input from stdin file

2. "stdout": buffer in a main memory which is associated with standard o/p device i.e. Monitor, bydefault any program writes ootput into stdout file.

3. "stderr": buffer in main memory which is associated with standard o/p device in which bydefault list of errors (compiler error or run time error) gets written into the stderr file

```
int main(void)
{
	int n1, n2, res;

	printf("enter n1 & n2: ");//stdout - is a buffer exists in main memory
	which is associated with standard o/p device

	scanf("%d %d", &n1, &n2);

	res = n1 + n2;

	printf("result : %d\n", res);

	return 0;
}
```

+ "system calls": are the functions defined in C, C++ & assembly language, which provides interface of services made available by the kernel for user (programmer).
Or in other words,

if programmer want to use services which are made available by the kernel in his/her program, it can be used either directly giving call to the system calls (i.e. system call API's) or indirectly by giving call to the system calls through set of library functions (provided in that programming language).

- In Original UNIX OS, total 64 system calls are there
- In Linux, total more than 300 system calls are there
- In Widows, more than 3000 sytem calls are there
- System calls are also reffered as "software interrupts": as due to the system call the CPU stops an execution of user process and starts an execution of a system process.

- there are 6 catagories of system calls:

1. process control system calls:
    - fork() -> to create a new/child process
    - _exit() -> exit() lib function in C internally makes call to _exit()
    - wait() -> to suspend a process for given time interval
    - etc...

2. file manipulation system calls:
    - open() -> to open a file/ to create a new file -> fopen()
    - close() -> to close a file -> fclose()
    - read() -> fgetc(), fgets(), fscanf(), fread()
    - write() -> fputc(), fputs(), fprintf(), fwrite()
    - lseek() -> fseek() - to set file cursor (to set byte position in a file)
    - etc...

3. device control system calls:
    - open()
    - read()
    - write()
    - ioctl()

4. inter process communication system calls:
    - pipe()
    - shmget()
    - shmat()
    - shmdt()
    - signal()
    - kill()
    - etc....

5. accouting information system calls:
    - getpid()
    - getppid()
    - stat() - use to display info about file & filesystem

6. protection & security system calls:
    - chmod() - to change mode bits and hence access perms

chown()
etc....

- when the CPU executes user defined code instructions system runs in "user mode"
whereas when the CPU executes system code instructions system runs in "system mode",
and this is reffered as "dual mode operation".

- system node is reffered as "kernel mode"/"super visor mode"/"priviledged mode"/"monitor mode"
- user mode is also reffered as "non-priviledged mode".