1. Draw or describe the Linux architecture layers (hardware → kernel → shell → user space). Where do system calls fit?

-		ers	tono = It it
-1	Shell & Com	mand Compiler & Internation	oreter {Shell
	system	Call Interface to ke	mel bo
	Signal Terminal Proce Terminal Or	Time Virt	uling Thermel
le nec 15.9	Terminal Controller Terminal	Interface to Hard Device Controller Controller Disks 19Pe Phy	16 35 478 15 16 17 18 19 19 19 19 19 19 19 19 19 19 19 19 19
WH CH	ard ware h		olication gram
# P	el: (ore of ocess Men ility to mili	nory / Fite System / Colize and Control all	evice Management resources on a comp
* 6 -Ax	Olication	rpretation, program	
# Var	scument edi	fing tools, network	cols related tools, etc

• System calls are the *bridge between user space and the kernel*, allowing user programs to request services like file I/O, networking, and process creation.

2. Explain the purpose of these directories: /, /bin, /sbin, /usr, /etc, /var.

- / (Root directory)
 - i. The top of the filesystem hierarchy.
- /bin (Essential user binaries)
 - i. Binary executables needed for basic system operation.
 - Contains programs required for all users, even in singleuser mode or maintenance.
- /sbin (System binaries)
 - Like /bin, but contains system administration tools (mostly for root).
 - ii. Used for system repair, configuration, and management.
- /usr (User programs & data)
 - i. Contains the bulk of user-space software.
- /etc (Configuration files)
 - i. System-wide configuration files (text files).
 - ii. No binaries here.
- /var (Variable data)
 - i. Contains files that change frequently during system operation.

3. Why does Linux treat everything as a file? Explain the difference between a program and a process.

- A. Uniform interface: Programs don't need to care whether they're reading from a text file, a hard disk, or a keyboard → they just use the same system calls: open(), read(), write(), close().
- B. Simplicity: One consistent API makes the OS easier to design and applications easier to write.
- C. Flexibility: You can redirect output of one program to another because everything is file-like.

• Program

- i. A file stored on disk (usually in /bin, /usr/bin, etc.).
- ii. Contains instructions + static data.

Process

- i. A program in execution.
- ii. When you run a program, the OS:
 - 1. Loads it into memory
 - 2. Creates a process control block (PCB)
 - 3. Assigns it a PID (Process ID)
 - 4. Manages its execution (scheduling, memory, I/O)