SIC-IOT702 / task 03 : capstone Task / phase 1/open ended questions

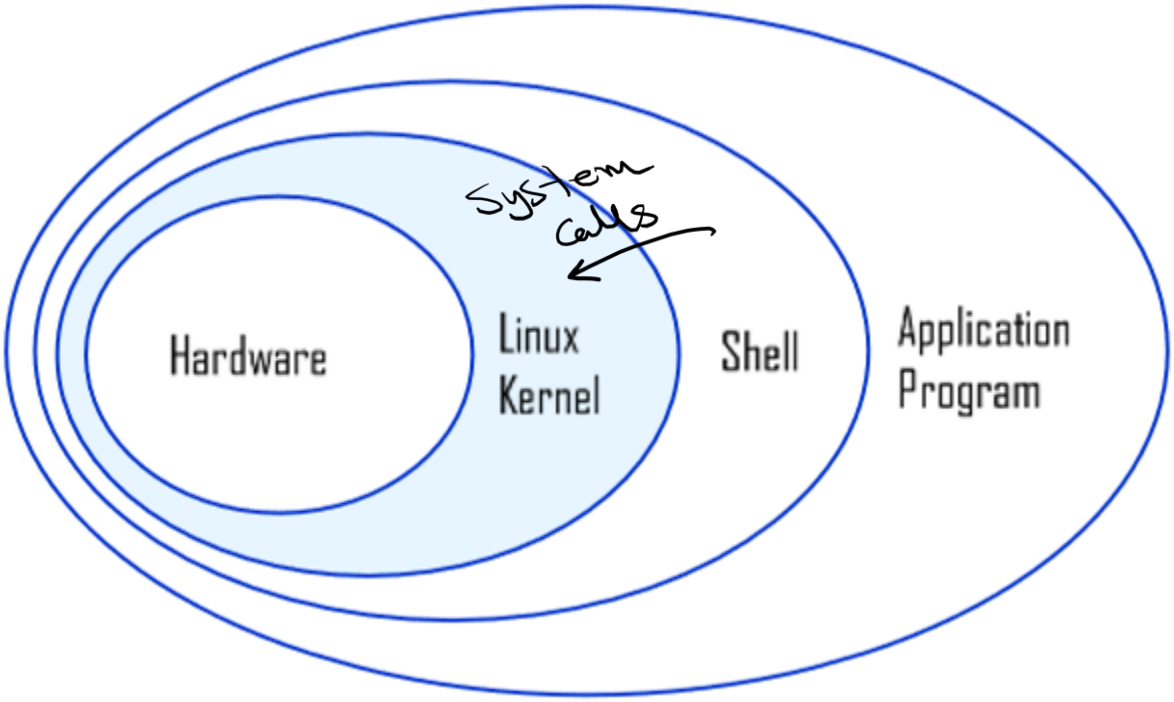
Instractor : eng/ Mohamed Ahmed

Student name: Ahmed Mohamed Elsayed

Linux Architecture:

- The Linux architecture is typically described in four layers:

hardware, **kernel**, **shell**, and **user space**.



- **Hardware** is the physical machine components, like the CPU, memory, and peripherals.
- The **Kernel** is the core of the operating system. It's a low-level program that directly interacts with the hardware and manages system resources, such as memory and processes.

- **Shell** is a command-line interpreter (like Bash) that acts as an interface between the user and the kernel. It takes commands from the user and translates them into system calls that the kernel can understand.
- **User Space** consists of all the applications and programs that users run, such as web browsers, text editors, and utility programs.
- **System calls** are the mechanism by which user-space programs request services from the kernel. They fit between the user space and the kernel, forming the bridge that allows applications to access hardware resources and perform privileged operations.

 Linux Directories:

- /: The **root directory**. This is the top of the file system hierarchy.
- /bin: Contains essential user **binaries** (executable programs) needed for basic system operations. Examples include

ls, cp, and mv.

- /sbin: Contains essential system **binaries** used by the system administrator for system maintenance. Examples include

reboot, fdisk, and fsck.

- /usr: Stands for "Unix System Resources." It is one of the largest directories and contains most user applications and utilities. Subdirectories like

/usr/bin and /usr/lib hold a vast collection of programs and libraries.

- /etc: Stores system-wide configuration files. It's where you'll find files that control how the system and applications behave, such as

/etc/passwd and /etc/ssh/sshd_config.

- /var: Stands for "variable." It contains files whose content is expected to change frequently, such as log files, mail queues, and temporary files. It's crucial for system logging and data that accumulates during system operation.

 "Everything as a File" & Program vs. Process:

- The principle "everything is a file" means that in Linux, all resources—including physical devices (like a hard drive or keyboard), directories, and sockets—are represented and managed by the kernel as files in the file system. This creates a

consistent and simple interface for interacting with various resources, allowing the same commands (

cat, cp, ls) to work across different resource types. For example, reading from a device like

/dev/null is done just like reading from a regular file.

- A **program** is a set of instructions stored on disk (e.g., an executable file like /bin/bash). It is a passive entity.
- A **process** is an active instance of a program being executed. When you run a program, the kernel creates a process to manage its execution. Each process has its own unique Process ID (PID), memory space, and other resources. A single program can have multiple processes running at the same time. For instance, if you open two separate terminal windows, you have two different

bash processes, even though they are running the same bash program.