Basmala Ehab Mohamed Yousry

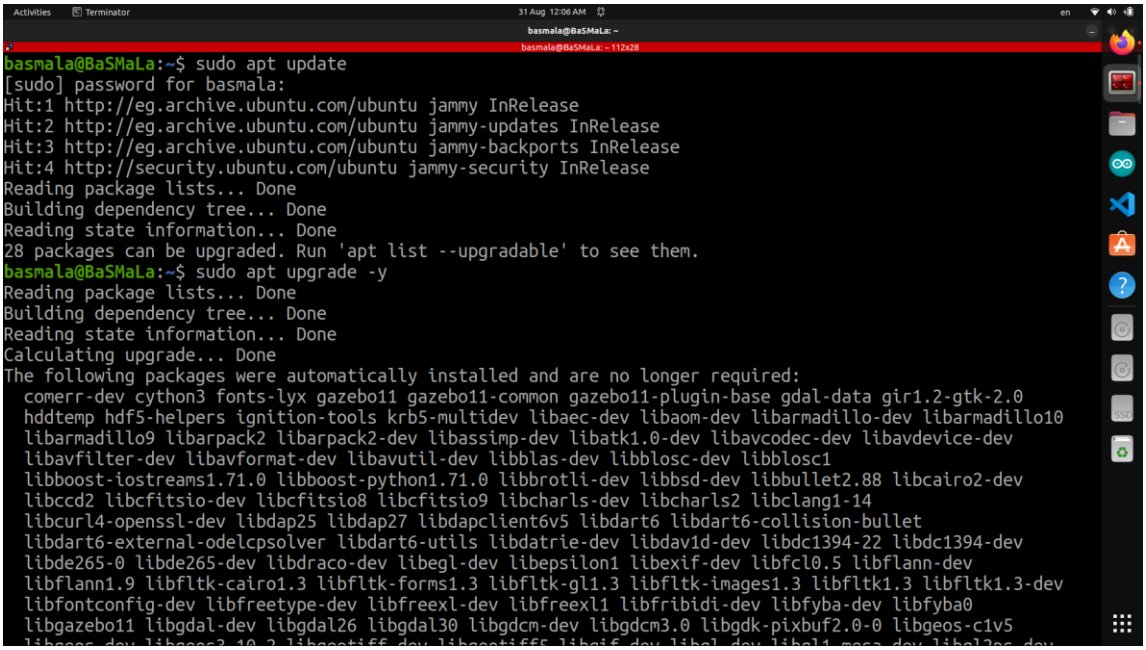
Group: IoT 702

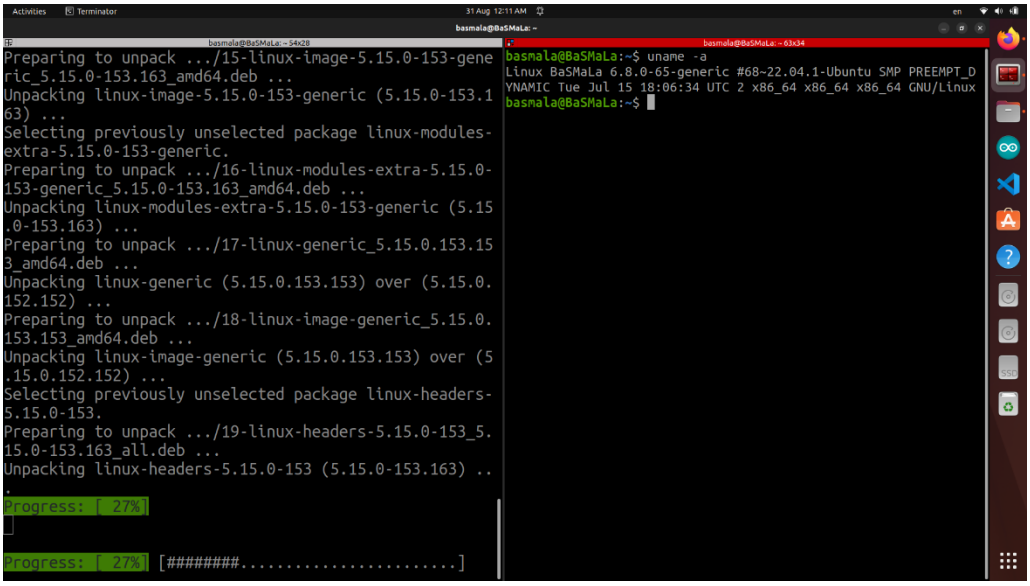
Capstone Task – IoT Sensor Data Logger on Raspberry pi

Day1 – phase1: System Update & Directory Setup

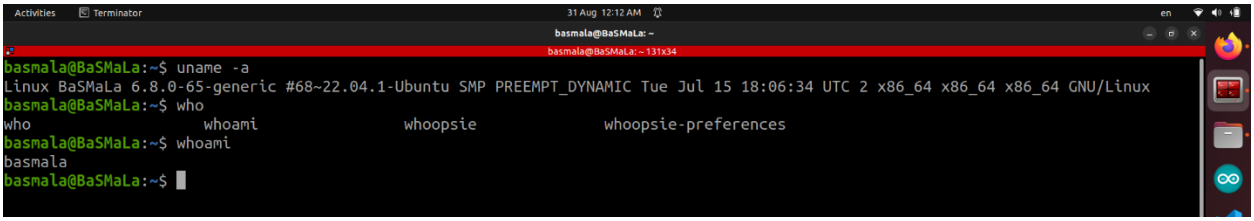
First, update and upgrade

sudo -y → y used to automatically answer “yes” to every question

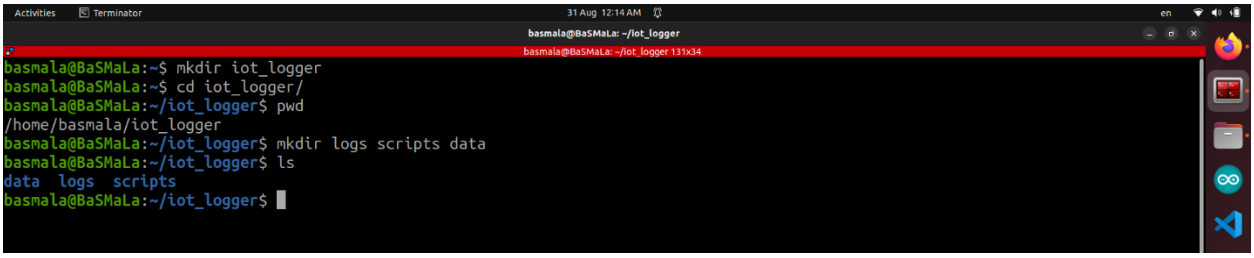




Verifying system details: kernel version, hostname, time and the user

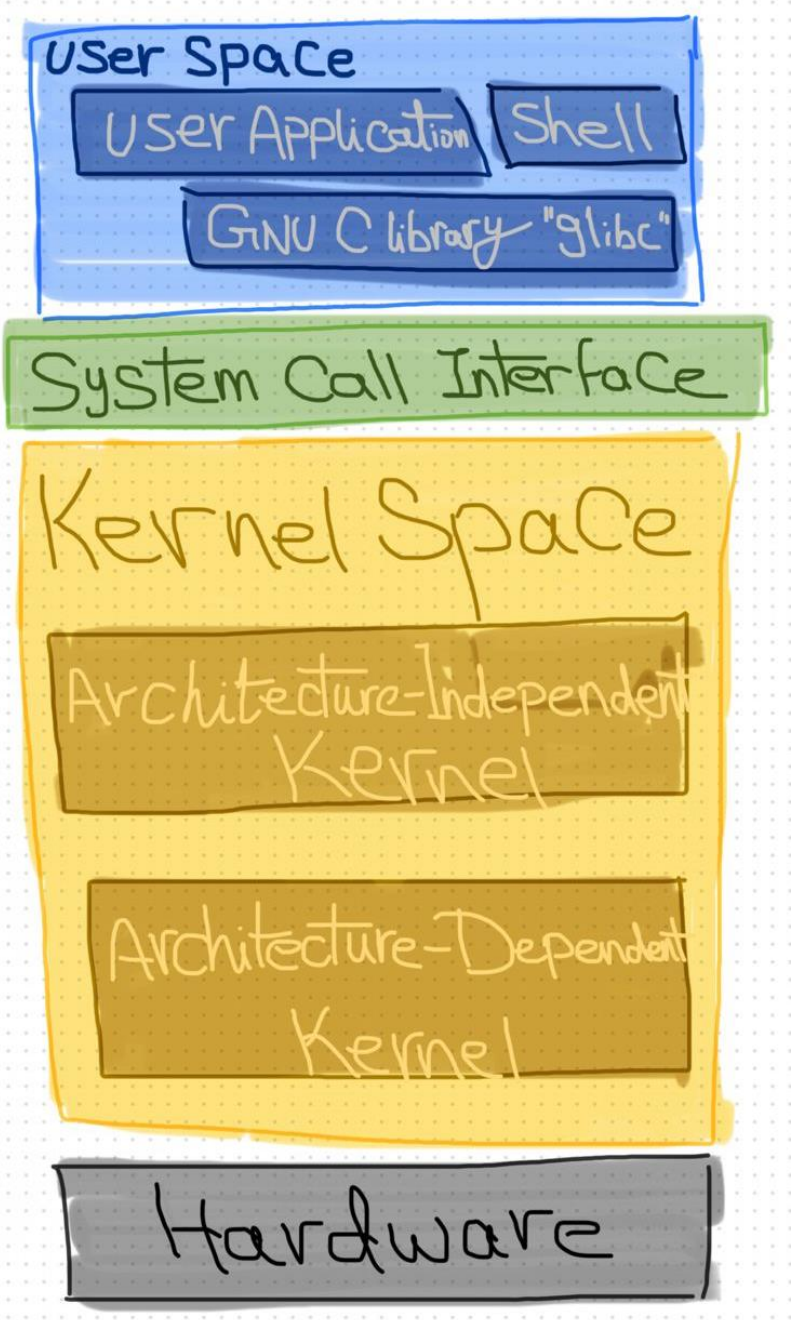


Creating the Directories



Open-Ended Questions:

Linux Architecture layers



User Space: the area where all the applications and processes run.

System Call Interface: functions that act as a bridge between the shell and the kernel, translating user commands into instructions the kernel can execute.

Kernel Space: The part of the system that communicates directly with the hardware, often through assembly language.

Architecture-Independent Kernel: Kernel components that perform general tasks as file management on a hard disk, without depending on the specific type or brand of hardware.

Architecture-Dependent Kernel: Kernel components written in assembly code that interact directly with hardware features, such as registers and the hardware.

Hardware: the CPU, Screen, Keyboard, etc.

Explain the purpose of these directories:

/ → root directory where all other files and directories branch out from it

/bin → Contains essential user commands as ls, cp, cat…

/sbin → Contains essential system administration binaries
/usr → includes the main executable library files and header files

/etc→ Stores system configuration files as passwd and network

/var → Contains files that change frequently during system operation

Why does Linux treat everything as a file?

Everything in Linux is treated as a file to keep the system simple and standardized. Instead of creating separate commands for each type of resource, the same system calls (open, read, write, close) can be used for all. This means a developer doesn't need to worry whether they are working with a disk file, a hardware device, or a network connection, all behave the same. Because of this design, tools can be easily combined using pipes (|) and redirection (>/<).

Program → file containing a set of instructions

Process→ program in execution