***Linux Interview Questions:***

Stages of Linux Boot Process:

Key Steps in the Linux Startup Proces:

1.The machine’s BIOS (Basic Input/Output System) or boot microcode hundreds or UEFI initializes hardware and runs a Power-On Self-Test (POST) or a boot loader.

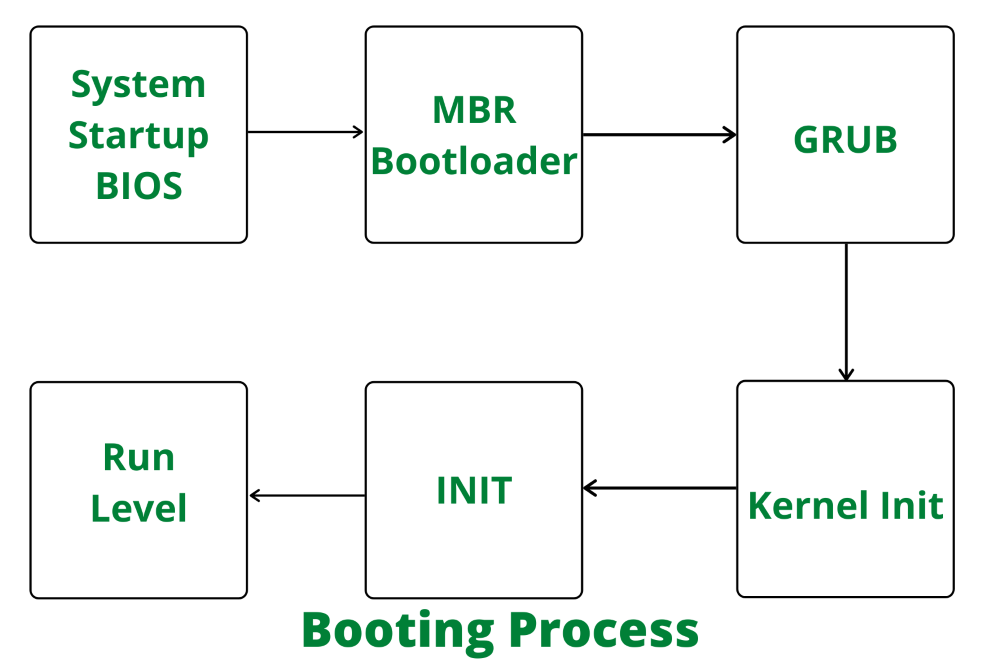
2.Boot loader finds the kernel image on the disk and loads it into memory, to start the system.

3.The kernel initializes the devices and their drivers.

4.The kernel mounts the basis filesystem.

5.The kernel starts a program referred to as init with a method ID zero

6.init sets the remainder of the system processes in motion.

7.For some purpose, init starts a method permitting you to log in, typically at the top or close to the top of the boot sequence.  


inux Boot Process: 6 Key Stages

1️⃣ BIOS / UEFI (Firmware Initialization)

What happens?

Runs POST (Power-On Self-Test)

Initializes hardware (CPU, RAM, storage, etc.)

Looks for a bootable device

Outcome: Loads the bootloader from the MBR or EFI partition.

2️⃣ Bootloader (GRUB/LILO/Systemd-boot)

What happens?

Loads and presents a boot menu

Lets you choose which kernel to boot (if multiple)

Loads the Linux kernel and initrd/initramfs into memory

Example bootloader: GRUB (most common)

3️⃣ Kernel Initialization

What happens?

Kernel decompresses and initializes

Detects hardware and mounts initramfs

Initializes drivers and sets up memory

Switches from real mode to protected mode

4️⃣ Initramfs / Initrd

What happens?

Temporary root file system in RAM

Helps mount the real root filesystem

Contains essential drivers and tools

Then: Hands control to the real root filesystem

5️⃣ Init System (systemd / SysVinit / Upstart)

What happens?

PID 1 process starts (usually systemd)

Initializes all user-space services and daemons

Mounts filesystems, sets hostname, configures network

6️⃣ Login / Display Manager

What happens?

Starts getty for command-line login or

Display Manager for GUI login (e.g., GDM, LightDM)

User logs in and starts a shell or desktop environment

🖼️ Visual Summary

BIOS/UEFI  
 ↓  
Bootloader (GRUB)  
 ↓  
Kernel + Initramfs  
 ↓  
Real Root Filesystem  
 ↓  
Init System (systemd)  
 ↓  
Login / Shell / Desktop

fstab in Linux?  
fstab stands for **File System Table**. It’s a configuration file in Linux located at:

/etc/fstab

It tells the system **what file systems to mount**, **where to mount them**, and **how to mount them** **at boot time**.

Purpose of fstab:

Automatically mount partitions at boot (e.g., /home, /var, external drives)

Define mount options (read-only, noexec, user access, etc.)

Improve consistency and automation in managing storage

📄 Sample /etc/fstab Entry:

UUID=abc123-def456 /data ext4 defaults 0 2

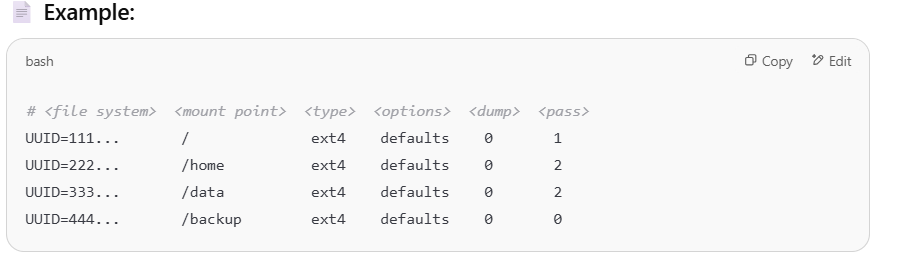
|  |  |
| --- | --- |
| **Field** | **Description** |
| **UUID/device** | Identifier for the partition (can use /dev/sda1, UUID=..., or LABEL=...) |
| **Mount Point** | Directory where the file system will be mounted (/, /home, /data, etc.) |
| **Filesystem Type** | Type of filesystem (ext4, xfs, ntfs, swap, etc.) |
| **Options** | Mount options (e.g., defaults, noexec, ro, nosuid, user) |
| **Dump** | Used by the dump utility; usually set to 0 |
| **Pass** | Used for fsck order; 1 = root, 2 = others, 0 = do not check |

In fstab entries we have UUid.... in the last we have numbers like 0 1, 0 2 , 0 0. What these numbers represent?  
In the /etc/fstab file, the **last field** (Pass) controls the order in which filesystems are **checked by fsck** (filesystem consistency check) **at boot time**.  
**Pass Field – Used by fsck**

|  |  |
| --- | --- |
| **Value** | **Meaning** |
| 0 | **Do not check** this filesystem at boot |
| 1 | Check **first**. Used only for the **root (/) filesystem** |
| 2 | Check **after root**. Used for **all other file systems** (e.g. /home, /var) |

**How It Works During Boot:**

* When the system boots, fsck runs **in order of pass number**:
  + Filesystems with 1 (usually just root /) are checked **first**.
  + Then, filesystems with 2 are checked **in parallel** (if on different disks).
  + Filesystems with 0 are **skipped**.

  
/ is checked **first**  
/home and /data are checked **after /**  
/backup is **never checked**  
**Why It Matters:**  
 Ensures the **root filesystem is clean** before mounting  
 Helps avoid **data corruption** on secondary drives  
 Skipping fsck (0) can speed up boot, but at risk if filesystem becomes inconsistent

Partitions in linux, their catogories?  
**Partition Usage Categories (by Function)**

Once partitions are created, they’re **mounted** to specific directories. Common functional categories include:

|  |  |  |
| --- | --- | --- |
| **Mount Point** | **Purpose** | **Type** |
| / | Root filesystem (required) | Primary or logical |
| /home | User data | Logical or primary |
| /var | Logs and variable files | Logical |
| /boot | Kernel and bootloader files | Usually primary |
| /tmp | Temporary files | Logical |
| /opt | Optional software | Logical |
| swap | Memory swap area | Primary or logical |

## **Commands to View Partitions**

lsblk # Lists block devices and mount points  
fdisk -l # Lists partition table and details  
parted -l # More advanced partitioning info

**Types of Partitions in Linux**

There are **three main categories** of disk partitions:

### **1. Primary Partitions**

* A physical disk can have **up to 4 primary partitions**
* Each primary partition is labeled as: /dev/sda1, /dev/sda2, etc.
* Can be **used to boot** the OS directly

✅ Example:

bash

CopyEdit

/dev/sda1 --> primary partition

### **2. Extended Partition**

* Only **one** extended partition is allowed per disk
* Acts as a **container** for **logical partitions**
* Used to **bypass the 4 primary partition limit**

✅ Example:

bash

CopyEdit

/dev/sda4 --> extended partition

### **3. Logical Partitions**

* Created **within** an extended partition
* Can have **more than 4** logical partitions on a single disk
* Numbering starts from /dev/sda5

✅ Example:

bash

CopyEdit

/dev/sda5, /dev/sda6, ... --> logical partitions

**How do you perform patching in a linux server ?**  
A) first back the important data. ---> snapshot, backup configuration files ec: /etc

B) Check current system info --> uname –r ,,,, cat /etc/os-release

c) Check available patches --> sudo dnf check-update  
d) Apply patches ---> sudo dnf upgrade –y   
e) once completed ---> reboot if necessary ---> sudo reboot

f) finally check the services are restarted correctly and are working fine ---> sudo systemctl status service-name

g) uname –r ---> to verify the version

Lets say you want to check the particular version of a package which is installed on linux  
rpm -q <package-name>  
yum list installed <package-name> # RHEL/CentOS 7

dnf list installed <package-name> # RHEL/CentOS 8+

Lets say I have deleted a particular processwhich was consuming most of the disk space but still after killing that process the space is not free what could the possible reason . How can you check that ?  
- The process might have an open file (ex: a log or temp file) that was deleted but the file descriptor is still open by the process before it was killed   
When you rm a file that's still **open by a running process**, the **file is deleted from the directory tree**, but **space is not freed** until:

* The **process closes the file**, or
* The **process is fully terminated**, **and the file descriptor is released**

**-check the files**   
**sudo lsof | grep deleted**   
output like:  
python3 2345 user 4r REG 8,1 52428800 123456 /var/log/app.log (deleted)  
  
**- identify the process & kill it (if still running**  
**sudo kill –9 pid**

Linux process vs services

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Process** | **Service (Daemon)** |
| 🔍 **Definition** | A running instance of a program | A **special kind of process** that runs in the background |
| 📦 **Started by** | A user or system directly | Usually started by **init system** (e.g., systemd) |
| 🛠️ **Managed by** | ps, kill, top, etc. | systemctl, service, chkconfig |
| ⏳ **Lifespan** | Runs as long as the parent/user allows | Often runs **indefinitely** or **auto-restarts** |
| 👤 **Runs as** | Any user (foreground or background) | Usually runs as **root** or system user |
| 📄 **Example** | firefox, vim, gcc, sleep 1000 | sshd, nginx, cron, mysql, docker |
| 🔄 **Persistence** | Ends when user logs out or process exits | Designed to **survive reboots**, auto-started |

I have deleted a 10 GB log file but the disk space did not free up   
Firstly find it using   
  
$ lsof | grep deleted  
  
The space is held by a process still writing to it   
The space is released only when the process closes the file or terminates.  
  
Kill the process or restart the service to remove the file.