LINUX BOOT PROCESS - EASY TECHNICAL EXPLANATION

FULL BOOT FLOW :

Power ON → BIOS/UEFI → MBR/GPT → GRUB2 → Kernel → initramfs → systemd → Login

---

1. Power ON (Electricity → Hardware Wakes Up)

What happens

- Power reaches the motherboard
- CPU resets and starts execution
- CPU jumps to a **fixed firmware address**

That address always contains **BIOS or UEFI code**

Who has control?

✓ CPU starts first
✓ BIOS/UEFI takes control automatically

If this fails →
✗ *You see beeps, black screen, or "No boot device found"*

Handover:
CPU starts executing firmware instructions → hands control to **BIOS/UEFI**

---

2. BIOS / UEFI - The Boot Manager

BIOS/UEFI decides **which disk to boot from**.

BIOS/UEFI is stored inside a small chip on the motherboard.

This chip is non-volatile memory (it doesn't lose data when power is off).

BIOS (Old systems)

- Works with **MBR partitions**
- Loads only the **first 446 bytes** of the disk

UEFI (Modern systems)

- Works with **GPT partitions**
- Boots from **EFI System Partition (ESP)** → FAT32 directory /boot/efi

BIOS/UEFI = **The system's boot decision-maker**

Handover:
BIOS/UEFI loads boot code from disk → hands control to **MBR or EFI file**

---

3. MBR or GPT - Where Bootloader Is Located

MBR (Master Boot Record)

MBR is the **first sector of the disk**, exactly **512 bytes**.

It contains:

Part	Size	Purpose
Bootloader Stage-1	446 bytes	Points to GRUB
Partition Table	64 bytes	Info about partitions
Magic Number	2 bytes	Integrity check

What MBR does

- Finds & runs **GRUB Stage-1.5 / Stage-2**
- Points to /boot/grub2

If MBR is corrupted →
✕ grub> prompt
✕ "Operating system not found"

Handover:
BIOS executes MBR code → MBR loads **GRUB**

GPT (GUID Partition Table)

Used in UEFI systems.

GPT includes:

- Many partitions
- Backup table
- Protective MBR

UEFI boots using **EFI System Partition** →
/boot/efi/EFI/redhat/grubx64.efi

Handover:

UEFI loads GRUB EFI binary → control moves to **GRUB2**

MBR/GPT = **The map telling the system where GRUB lives**

---

4. GRUB2 - The Bootloader

GRUB is responsible for loading:

✓ Kernel → /boot/vmlinuz-*

✓ initramfs → /boot/initramfs-*

GRUB Responsibilities:

- Shows boot menu
- Lets you select a kernel
- Passes boot parameters (rd.lvm.lv=..., root=UUID=...)
- Hands control to the kernel

If GRUB fails:

- grub rescue> appears
- Boot stops because kernel cannot load

GRUB = **The menu + loader that starts Linux**

Handover:
GRUB loads kernel into memory → hands control to **Kernel**

---

5. Kernel - The Heart of Linux

Once GRUB loads the kernel (vmlinuz), kernel starts:

Kernel does:

- Detects hardware (disks, NICs, CPU)
- Loads drivers
- Mounts a temporary root filesystem (initramfs)
- Starts memory management
- Starts process ID **PID 1 (systemd)**

If kernel fails →
✗ Kernel panic
✗ "Unable to mount root fs"

Kernel = **The main controller of hardware + OS**

Handover:
Kernel starts initramfs → later starts systemd

---

6. initramfs - Temporary Root Filesystem

Before the kernel can access your real / (root), it needs drivers.

initramfs includes:

- LVM drivers
- RAID drivers
- Multipath drivers
- Encryption drivers
- Filesystem drivers (ext4, xfs)

What initramfs does:

- Activates LVM
- Activates RAID
- Connects SAN disks
- Loads required modules
- Finds the real root filesystem
- Switches from **temporary root → real root**

If initramfs fails:
✕ You drop into **dracut emergency shell**

initramfs = **The helper that prepares storage before real root loads**

Handover:
initramfs mounts real / → returns control to **Kernel**

---

7. systemd - Service & Boot Manager

After switching to the real root, kernel starts:

/usr/lib/systemd/systemd

systemd is **PID 1**, the first process.

systemd does:

- Mounts filesystems
- Starts all services
- Applies targets (previously runlevels)
- Activates networking
- Activates SSH, firewalld, multipathd

Common issues:

❌ "A start job is running…"
❌ Boot drops to **emergency mode** (fstab problem)

systemd = **The OS startup manager**

Handover:
Kernel hands full userspace control to **systemd**

---

8. Login Prompt (CLI or GUI)

Finally, system reaches:

- **multi-user.target** → CLI login
- **graphical.target** → GUI login (for desktops)

System is now **fully ready**.