Init Systems and Linux Boot Process

This tutorial is designed for **absolute beginners** as well as **working professionals** who want a clear, structured, and practical understanding of how Linux starts, how services are managed, and how init systems evolved.

---

1. What Happens When You Power ON a Linux System?

When you press the **power button**, Linux does **not** start immediately. A sequence of well-defined steps occurs, called the **Linux Boot Process**.

High-level stages: 1. BIOS / UEFI 2. Bootloader (GRUB) 3. Kernel Loading 4. Init System (PID 1) 5. User Space & Services

We will go through each stage step-by-step.

---

2. BIOS / UEFI Stage

What is BIOS?

- **BIOS (Basic Input Output System)** is firmware stored on the motherboard.
- It performs **POST (Power-On Self Test)**:
    - CPU check
    - RAM check
    - Keyboard and disk detection

UEFI (Modern Replacement)

- Faster
- Supports large disks (>2TB)
- Secure Boot support

Key Responsibility

➞ Find a **bootable device** (HDD, SSD, USB)

---

2. Bootloader Stage (GRUB)

What is a Bootloader?

A **bootloader** loads the Linux kernel into memory.

Common Bootloader

- **GRUB (GRand Unified Bootloader)**

What GRUB Does

- Displays OS menu
- Allows kernel selection
- Loads:
  - Kernel (`vmlinuz`)
  - Initramfs (`initrd` / `initramfs`)

Important Files

```
/boot/grub2/grub.cfg
/boot/vmlinuz-*
/boot/initramfs-*
```

---

4. Linux Kernel Stage

What is the Kernel?

- Core of the OS
- Manages:
  - CPU
  - Memory
  - Devices
  - Filesystems

Kernel Responsibilities During Boot

1. Decompress itself
2. Initialize hardware drivers
3. Mount root filesystem (temporary)
4. Execute the **first user-space process**

➞ This first process is **INIT (PID = 1)**

---

5. Init System (PID 1) – The Heart of Userspace

What is an Init System?

An **init system** is the **first process started by the kernel**.

- Process ID: **1**

- Parent of all processes
- Never exits

Responsibilities

- Start system services
- Mount filesystems
- Handle shutdown and reboot
- Manage system states (runlevels / targets)

---

6. Types of Init Systems in Linux

Evolution

1. SysVinit (Traditional)
2. Upstart (Intermediate – Ubuntu older versions)
3. systemd (Modern – Most distributions today)

We will focus on **SysVinit** and **systemd**.

PART 1: SysVinit

7. What is SysVinit?

- Oldest init system
- Derived from UNIX System V
- Sequential startup
- Script-based

Used in: - RHEL 5 - CentOS 5 - Very old Linux systems

---

8. SysVinit Architecture

Key Files and Directories

```
/sbin/init
/etc/inittab
/etc/init.d/
/etc/rc.d/
```

Process Flow

Kernel → `/sbin/init` → `/etc/inittab`

---

9. Runlevels in SysVinit

A **runlevel** defines the system state.

Runlevel	Meaning
0	Halt (Shutdown)
1	Single-user mode
2	Multi-user (no network)
3	Multi-user with network (CLI)
4	Unused / Custom
5	GUI mode
6	Reboot

---

10. /etc/inittab (Heart of SysVinit)

Example:

```
id:3:initdefault:
```

Meaning: - Default runlevel = 3

---

11. Service Startup in SysVinit

Services are scripts located in:

```
/etc/init.d/
```

Runlevel directories:

```
/etc/rc3.d/
/etc/rc5.d/
```

Symbolic links: - `S10network` → Start - `K10httpd` → Kill

Sequence

- Lower number → starts earlier

---

12. Commands in SysVinit

```
service httpd start
chkconfig httpd on
runlevel
init 3
```

---

13. Limitations of SysVinit

- Slow boot (sequential)
- No dependency management
- Hard to maintain scripts
- No monitoring

➞ These limitations led to **systemd**

PART 2: systemd

14. What is systemd?

- Modern init system
- Parallel startup
- Dependency-based
- Event-driven

Used in: - RHEL 7+ - CentOS 7+ - Ubuntu 16.04+ - Fedora

---

15. systemd Concepts

Units

Everything is a **unit**

Unit Type	Purpose
service	Daemons
target	System state
mount	Mount points
socket	Socket activation

Location:

```
/lib/systemd/system/
/etc/systemd/system/
```

---

16. Targets (Replacement for Runlevels)

SysV Runlevel	systemd Target
0	poweroff.target
1	rescue.target
3	multi-user.target
5	graphical.target
6	reboot.target

---

17. systemd Boot Flow

Kernel → systemd (PID 1) → default.target → services

Check default target:

```
systemctl get-default
```

Set default target:

```
systemctl set-default graphical.target
```

---

18. Managing Services in systemd

Start service:

```
systemctl start httpd
```

Enable at boot:

```
systemctl enable httpd
```

Check status:

```
systemctl status httpd
```

---

19. systemd Unit File Example

```
[Unit]
Description=Apache Web Server
After=network.target
```

```
[Service]
ExecStart=/usr/sbin/httpd
Restart=always

[Install]
WantedBy=multi-user.target
```

---

20. Advantages of systemd

- Faster boot (parallel)
- Service dependency handling
- Auto restart
- Centralized logging (journald)
- Better resource control

---

21. Logging in systemd

View logs:

```
journalctl
journalctl -u httpd
journalctl -b
```

---

22. SysVinit vs systemd (Comparison)

Feature	SysVinit	systemd
Startup	Sequential	Parallel
Scripts	Shell scripts	Unit files
Dependency	No	Yes
Speed	Slow	Fast
Logging	syslog	journalctl

---

23. Interview-Oriented Summary

- Kernel starts PID 1
- PID 1 is init system
- SysVinit uses runlevels
- systemd uses targets

- systemd is default in modern Linux

---

24. Hands-on Practice Suggestions

1. Check PID 1

```
ps -p 1 -o comm=
```

2. List services

```
systemctl list-units --type=service
```

3. Change default target

```
systemctl set-default multi-user.target
```
