ChatGPT

Computer Boot Process: From Power-On to OS Login

Power-On Self-Test (POST)

When a computer is powered on, the CPU is reset and immediately begins executing firmware code at the reset vector in ROM. The firmware (BIOS or UEFI) performs the **Power-On Self-Test (POST)** – a set of diagnostics to verify core hardware functionality [1]. During POST, the system initializes and checks components such as:
- **CPU and chipset:** the processor and system controller logic are brought online.
- **System memory (RAM):** memory modules are counted and subjected to basic tests.
- **Video subsystem:** the graphics adapter is initialized so that status messages can be displayed.
- **Storage controllers and I/O devices:** disk controllers, keyboard, etc., are probed for presence.

If the POST encounters a critical error, the boot process is halted and an error code (often via beep codes or on-screen messages) is signaled [1]. Only when POST completes successfully does the firmware transfer control to the next stage – loading the bootloader.

Firmware (BIOS vs UEFI)

Modern PCs use one of two firmware interfaces to continue booting: legacy BIOS or the newer UEFI. They differ significantly in architecture, capabilities, and features:

- **Legacy BIOS:** The original PC firmware interface, written for 16-bit real-mode execution. It uses a Master Boot Record (MBR) on disk for booting. MBR-based partitioning is limited to ~2 TiB disk sizes and a maximum of four primary partitions [2]. BIOS initialization is largely sequential (checking one device at a time), and its setup UI is text-only. It has no built-in secure-boot mechanism.

- **UEFI (Unified Extensible Firmware Interface):** A modern 32-/64-bit firmware environment that replaces BIOS. UEFI can boot from GUID Partition Table (GPT) disks, supporting drives far larger than 2 TiB and many partitions [2]. It initializes hardware in parallel using driver modules (speeding up boot times) and often provides a mouse-driven graphical setup interface [3]. Crucially, UEFI can enforce **Secure Boot**, where each driver or bootloader must be digitally signed. The UEFI specification "secures the boot process by preventing the loading of ... drivers or OS boot loaders that are not signed with an acceptable digital signature" [4].

Aspect	Legacy BIOS	UEFI
CPU Mode	16-bit real mode (limited pre-boot environment)	32-/64-bit (protected/long mode, full driver support)

Aspect	**Legacy BIOS**	**UEFI**
Partitioning	MBR (≤2 TiB disks, ≤4 primaries)	GPT (>2 TiB disks, practically unlimited partitions) [2]
Bootloader Location	First sector of disk (MBR)	EFI executable (.EFI) in a FAT-formatted EFI System Partition [5]
Boot Speed	Slower (sequential device checks)	Faster (parallel hardware init) [3]
Secure Boot	Not supported	Supported: only signed bootloaders/drivers load [4]
Setup Interface	Text-based BIOS menu	Modern GUI with mouse support [3]

Figure: Legacy BIOS boot process. In legacy BIOS mode, after POST the firmware loads the Master Boot Record (MBR) from the boot disk. The small bootloader in the MBR then loads additional code (e.g. GRUB stages or Windows Boot Manager) from disk, which in turn loads the OS kernel.

Figure: UEFI boot steps. In UEFI mode, the firmware reads the boot entry from NVRAM and loads an EFI application (such as a `.efi` bootloader) from the EFI System Partition on disk. This EFI bootloader (GRUB, Windows Boot Manager, etc.) then proceeds to load the OS kernel.

Bootloader Stage (GRUB and Windows Boot Manager)

Once firmware chooses the boot device, it launches the bootloader. The exact mechanism differs for BIOS vs UEFI and Linux vs Windows:

- **Linux (GRUB):** On BIOS systems, the firmware jumps to the first sector (MBR) where a tiny GRUB Stage 1 is located. This Stage 1 code points to and loads GRUB Stage 1.5 (if present) or directly Stage 2 from the filesystem [6]. Stage 2 (the full GRUB engine) then displays a boot menu and loads the selected Linux kernel (the `vmlinuz` image) and initial RAM disk (`initrd`) into memory [7]. On UEFI systems, the firmware instead loads the GRUB EFI binary (e.g. `\EFI\grub\grubx64.efi`) from the EFI System Partition. The rest of the process is similar: GRUB shows a menu and loads the kernel/initrd.

- **Windows (Boot Manager):** On BIOS systems, the firmware loads the MBR and executes Windows' boot code, which runs **Windows Boot Manager** (`bootmgr`). On UEFI systems, the firmware directly loads the `bootmgfw.efi` file from the EFI System Partition [8]. The Windows Boot Manager reads the Boot Configuration Data (BCD) to locate the OS loader. It "finds and starts the Windows loader (Winload.exe)" on the boot partition [9]. Winload then loads the Windows kernel (`ntoskrnl.exe`), the Hardware Abstraction Layer (HAL), and core drivers into memory before transferring control to the kernel.

GRUB Boot Stages (Linux): GRUB's boot process consists of multiple stages [6]. Stage 1 (in the MBR) simply chains into Stage 2. An optional Stage 1.5 can be used to interpret filesystems. Stage 2 (loaded from `/boot/grub`) is the largest stage: it initializes the menu, loads the kernel and initrd image, and finally hands off to the Linux kernel [7]. GRUB is very flexible – it supports many filesystems and can present multiple OS entries.

Windows Boot Steps: The Windows Boot Manager (in BIOS or UEFI mode) uses the BCD store to find the OS loader. Once Winload executes, it uncompresses the NT kernel and HAL, and begins loading essential kernel drivers. As one Microsoft document notes, "the PC's firmware … loads the Master Boot Record (MBR) into memory, and then starts Windows Boot Manager" in BIOS mode, while "in UEFI mode the firmware loads and starts the Windows Boot Manager EFI executable" [8]. The Boot Manager then starts Winload.exe (or `Winload.efi` under UEFI) [9], which in turn loads the OS.

Operating System Loading and Initialization

At this point, the OS kernel is loaded into RAM and execution continues inside the operating system:

- **Windows:** After Winload starts the kernel (`ntoskrnl.exe`), the Windows kernel initializes and loads the system registry and drivers flagged as BOOT_START [10]. It then starts the Session Manager process (Smss.exe) to set up user sessions. Smss initializes the Windows subsystem, launches `wininit.exe` and `winlogon.exe`, and eventually displays the login screen for the user.

- **Linux:** After GRUB loads `vmlinuz` and the `initrd` image, the kernel decompresses and mounts the initramfs. The kernel then executes the `/init` program on the initramfs (this is typically a shell or early-init script) [11]. `/init` is responsible for mounting the real root filesystem and pivoting over. Eventually the kernel runs `systemd` (or another init system) as PID 1. Systemd initializes all remaining services (networking, udev, etc.) in parallel for efficiency. When all services are up, systemd spawns the graphical or console login manager. In other words, once the kernel is in user-space, **systemd brings up userland**, and "[o]nce all necessary services have been started, the system is ready to use and the login manager is displayed" [12].

Windows vs Linux Boot Sequence (Step-by-Step)

1. **Power On & POST:** The CPU powers on and jumps to firmware. Both Windows and Linux systems run the firmware POST diagnostics (see above) [1]. Only if POST succeeds does the process continue.

2. **Firmware Boot Phase:**

3. *Windows:* The firmware locates the boot device. In BIOS mode it reads the Master Boot Record (MBR) of the primary disk, which contains Windows' first-stage boot code. In UEFI mode it reads the EFI boot entry and loads `\EFI\Microsoft\Boot\bootmgfw.efi` [8].

4. *Linux:* Similarly, in BIOS mode the firmware loads the MBR where GRUB stage1 resides (often also loading a hidden GRUB gap or stage1.5). In UEFI mode the firmware loads a specified EFI executable (for example, `\EFI\ubuntu\grubx64.efi` on many Linux distros).

5. **Bootloader Execution:**

6. *Windows:* The Windows Boot Manager starts (from the MBR or EFI executable). It reads the BCD to find and invoke Winload.exe [9]. Winload loads the NT kernel (`ntoskrnl.exe`), HAL, and core boot drivers into memory.
7. *Linux:* GRUB's Stage 2 takes over. It displays the boot menu (allowing the user to select a kernel or OS). Once selected (or default timed out), GRUB loads the Linux kernel image (`vmlinuz`) and the initial RAM disk (`initrd`) into memory, passing any kernel command-line parameters [7].

8. **Kernel Startup:**

9. *Windows:* The Windows kernel initializes. It loads the system registry and drivers marked for boot (BOOT_START) [10]. Then it transitions to start the user session.
10. *Linux:* The kernel decompresses and mounts the initramfs, running `/init` [11]. That script typically mounts the real root filesystem and then execs the real init (systemd or similar). Modules needed for disk, networking, etc. are loaded during this phase.

11. **User Space and Login:**

12. *Windows:* The kernel launches the Session Manager (Smss.exe) to set up the Windows subsystem. Smss starts `winlogon.exe`, which presents the user login screen. At this point, the user can log on and begin a Windows session.
13. *Linux:* The init system (systemd) continues initializing system services in parallel (network, GUI manager, getty terminals, etc.). Once initialization is complete, a login prompt or graphical login manager (e.g. GDM, LightDM) appears [12]. The user can then log in to their Linux session.

Each of the above steps involves many detailed sub-steps (e.g. loading device drivers, initializing services, mounting file systems), but this outline covers the major transitions from power-on to user logon for both Windows and Linux systems.

Sources: Authoritative technical documentation and articles were used, including firmware/boot specifications and operating system documentation [1] [13] [4] [14] [10] [15] [11] [16]. Diagrams from system boot references are used above to illustrate the legacy BIOS and UEFI boot flows.

---

[1] POST (Power-On Self-Test) test - BIOS and UEFI - CompTIA A+ questions | TrustEd Institute
https://trustedinstitute.com/concept/comptia-a-plus/bios-and-uefi/post/

[2] [13] Windows Setup: Installing using the MBR or GPT partition style | Microsoft Learn
https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/windows-setup-installing-using-the-mbr-or-gpt-partition-style?view=windows-11

[3] Comparing BIOS vs UEFI: Which is the better Boot in 2025?
https://cyberpanel.net/blog/bios-vs-uefi

[4] [5] UEFI - Wikipedia
https://en.wikipedia.org/wiki/UEFI

[6] [7] [15] What is GRUB Bootloader in Linux? | phoenixNAP KB
https://phoenixnap.com/kb/what-is-grub

[8] [9] [10] [14] Windows boot issues troubleshooting - Windows Client | Microsoft Learn
https://learn.microsoft.com/en-us/troubleshoot/windows-client/performance/windows-boot-issues-troubleshooting

[11] About initramfs
https://www.linuxfromscratch.org/blfs/view/12.3/postlfs/initramfs.html

[12] [16] Boot Process with Systemd in Linux | GeeksforGeeks
https://www.geeksforgeeks.org/boot-process-with-systemd-in-linux/