Exploring the working principles, types, and multi-boot capabilities of the modern bootloader system

Zarif Rashid
ID: 23201317
Section: 18
CSE321 Assignment
Department of Computer Science and Engineering
BRAC University
Email: zarif.rashid@g.bracu.ac.bd

Abstract This article explains the start-up process of computers, comparing with the legacy BIOS and modern UEFI starting from power on, hardware detection, self-test, security check, and loading the operating system into the computer's memory. It also compares the Disk size and their partitioning in both of the boot systems. The next multi-boot system is explained to give the user a choice of which operating system they want to load. Finally, it shed light on common challenges in bootloaders and solutions to operating system detection and chain loading.

1. Introduction

A boot loader, also known as a bootstrap loader, is a small program that is responsible for loading the operating system's data into the computer memory during device startup. Boot loaders are responsible for making sure the hardware devices and the I/O devices are all correctly initialized, and also perform security checks. Boot loaders have evolved from the old-age BIOS system to modern UEFI systems, offering enhanced capabilities, better hardware support, and advanced security measures. These systems are slightly different from each other in their workflows, with varying architectural designs and implementations. Also, modern technology has implemented multiboot systems for multiple operating systems, which we will discuss in the latter part of the assignment.

2. Boot Workflow

2.1 Complete boot workflow from power-on to OS handoff for both BIOS and UEFI systems.

When we turn on our computer, it doesn't instantly start the operating system. There's a whole sequence of steps that happen first to get the hardware ready and then load the OS. This is called the boot process or boot workflow. BIOS holds the instructions on how to load basic computer hardware. The complete boot process of a BIOS system is given below,

Step 1 - When power is turned on, the power supply system sends electrical power to the hardware components of the computer, including the motherboard.

Step 2 - Upon receiving the power, the CPU begins execution at a predefined memory address where the jump instruction to the actual BIOS startup code is placed.

Step 3 - The BIOS performs the Power-On Self Test (POST), where it performs the basic hardware checks like CPU, RAM, keyboard, mouse, graphics card, and storage unit. It also checks for user input; if the F2 key is pressed for Windows, then it interrupts the normal boot process and enters the BIOS setup utility.

Step 4 - BIOS looks for the Video Card's built-in BIOS program and runs it, which then displays basic text and graphics on the monitor. Usually, most modern video cards will display information on the screen about the video card.

Step 5 - BIOS performs a system inventory of sorts, where it detects connected hardware like drivers, ports, and if supported by the Plug & Play standards, it will configure and plug-and-play devices at this time and display a message on the screen for each one it finds. This may flash quickly on the monitor's screen.

Step 6 - BIOS looks for the boot drive to start booting from, e.g., Floppy → HDD → CD → USB. Some BIOSs even let the PC boot from a CD-ROM drive.

Step 7 - Upon finding the boot drive, it looks for the Master Boot Record (MBR) and executes the bootloader code from the MBR sector.

Step 8 - Bootloader loads the operating system from the kernel, and the control passes from BIOS to the operating system. Here, the BIOS role ends and the OS begins its own initialization.

UEFI stands for Unified Extensible Firmware Interface. It does the same job as BIOS, but rather than storing the data in the firmware, it stores it at initialization and startup in an .efi file. This EFI system lies on the EFI System Partition on the Hard Disk, which also contains the bootloader. Let's look at the workflow of this UEFI boot system.

Step 1 - Same as BIOS. Powering on sends electric signals to the computer hardware, CPU resets, and the Chipset root-of-trust starts execution.

Step 2 - Security Phase (SEC). CPU starts at a predefined location, and it runs in 32-bit mode, sometimes switching to 64-bit. Sets up cache as RAM for temporary storage. Performs minimal platform initialization and security checks.

Step 3 - Pre EFI Initialization- Here. The PEIMs (modules) are responsible for base hardware initialization, such as primary memory configuration. It builds HOBs (hand-off data) for the next phase and prepares a minimal platform environment.

Step 4 - Driver Execution Environment (DXE) - Here, the UEFI drivers load, provide UEFI boot services, and runtime services.

Step 5 - BDS (Boot Device Selection) phase. Reads NVRAM boot entries, finds the EFI System Partition (ESP), and selects the EFI bootloader according to priority.

Step 6 - As soon as the BDS phase is completed, the TSL (Transient System Load) phase starts, where the system loads .efi bootloader and, after performing necessary verification (Secure Boot), the bootloader loads the OS kernel and necessary files. In the real Windows world, this is where the Windows Boot Manager will now load Windows.

Step 7 - Bootloader calls ExitBootService() and control is transferred to the OS kernel. Now OS initializes drivers and the runtime environment.

2.2 Roles of Firmware, Boot Records (MBR) & Boot Manager (Bootloader) in each process

Firmware, which is stored in ROM on the motherboard, is the first software that loads when the system powers on. It initializes hardware such as CPU, RAM, etc. (SEC & PEI phase) in UEFI. Performs security checks like Secure Boot. Provides UEFI runtime services to the OS. It also selects the boot device based on NVRAM boot entries. Mainly, firmware is responsible for preparing the environment and passing control to the next stage.

Boot Records, i.e., the MBR/GPT, are disk-resident structures that store information about disk partitions and initial boot code. It usually resides in the first sector, essentially the first 512 bytes of a Legacy BIOS System. It contains the partition table and bootstrap code to locate the bootloader. For the UEFI system, the EFI System Partition contains the GPT (GUID Partition Table), which has the bootloaders. Boot records point to or contain the first code to load the boot manager. They don’t handle hardware initialization or OS loading themselves.

Boot Manager, i.e., the Bootloader, is the software that loads the OS kernel. The UEFI boot manager (.efi) is located in the EFI System Partition, and it holds a boot menu, selects between multiple OSes, and enforces Secure Boot validation. Whereas the Legacy BIOS Bootloader resides in the MBR, it loads the OS kernel into main memory and passes control to the OS.

3. Architectural Comparison of BIOS versus UEFI

3.1 The technical architectural Differences

BIOS	UEFI
• A Legacy System that has been around since the early days of personal computing. • It is stored in the ROM on the motherboard. • Uses a 16-bit processor mode and is limited to the first 1MB of addressable memory. • Initializes hardware through POST and has a simple interface and a straightforward boot process that relies on MBR. • MBR supports up to 4 physical partitions, up to 2.2TB. • Compatible with Windows 7 and earlier versions and older Linux kernels. • Provides a user interface through a text-based interface, which is controlled by keyboard. • No native drivers for modern hardware. • It has basic security features,but no Secure Boot.	• Modern design of BIOS to overcome its limitations. • Stored in flash memory too, but designed as a modular C-based environment. • Operates in 32-bit or 64-bit mode with access to full system memory. • Has its own mini OS environment, which can be accessed through holding F2 as soon as you power on the computer. • Its OS can load drivers and applications. • Boots via EFI System Partition (ESP) and uses the GPT partitioning scheme. • Supports up to 128 physical partitions, up to 18 exabytes. • Compatible with Windows 7 and all newer operating systems, including modern Linux distributions and macOS. • Supports GUI (Graphical User Interface), which means mouse control making navigation easier. • Supports Secure Boot, which helps prevent unauthorized software from loading during the boot process.

Figure 1: Comparison of BIOS and UEFI

3.2 MBR vs GPT Partition Table

MBR is an older version of the partitioning scheme for the BIOS bootloader and has 2 types of partitions: primary and extended. Uses the first 512 bytes of the disk to store boot code and partition table. Supports only 4 primary partitions. Doesn’t support any error detection and is less secure.

GPT is a modern partitioning scheme for the UEFI bootloader, which has a special EFI System Partition (ESP) plus a partition table with globally unique IDs. It supports up to 128 partitions by default, which can be expanded and can only be limited by the OS. Supports CRC-32 error detection mechanism.

3.3 Disk Size Restrictions

MBR supports a sector of 512 bytes, and the maximum addressable space is about 2.2 TB. Whereas GPT supports disks up to 18 exabytes, and there is no 2TB barrier.

3.4 Hardware Compatibility

MBR is compatible with older OS like DOS, Windows XP, and early Linux. Still widely supported for legacy compatibility. Limited for modern high-capacity drives (fails beyond 2 TB). On the other hand, GPT is required for modern OS’s, Windows 10, Windows 11, MacOS, and Kali Linux. Needed for disks larger than 2 TB and SSDs with NVMe. Provides backward compatibility via CSM (Compatibility Support Module) to emulate BIOS if needed.

4. Multi-Boot Systems:

4.1 Process of Multiboot Loaders Managing Multiple Operating Systems

A multibootloader is software that allows a computer to boot into one of several installed operating systems, presenting a menu at startup to let the user choose which one to load. Multiboot loaders manage multiple operating systems by initializing a program that runs at startup and decides which OS to launch. There are a few types of multibootloaders like GRUB (GRand Unified Bootloader), mostly used in Linux, then rEFIND, which is used in UEFI systems, and Windows Boot Manager which is limited to different Windows operating systems. GRUB is commonly used because it can load both Windows and Linux. When Linux is installed, GRUB is installed. Let's see a step-by-step workflow on how it manages multiple OS.

Step 1 - When the computer is given power, the BIOS looks at the MBR, and the MBR tells the computer to run GRUB.

Step 2 - GRUB then shows a menu with options, for example, Ubuntu or Windows 10, or any other OS.

Step 3 - If Linux is chosen, GRUB loads the Linux kernel; or if Windows is chosen, GRUB uses a process called chainloading, where it passes control to its own boot program like Windows Boot Manager, thus starting Windows.

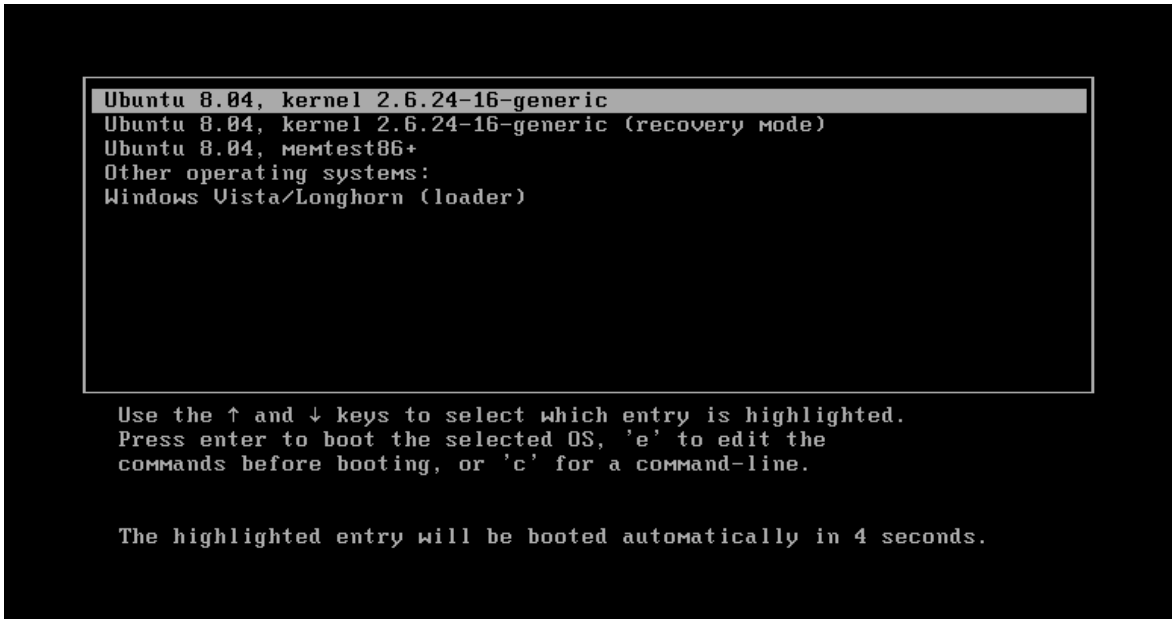

Figure 2: Startup Screen for Multiboot Loader for Multiple OS

4.2 Menu Implementation

GRUB uses configuration files like /boot/grub/grub.cfg at startup to list all the available OS's. It can automatically detect installed OSes through tools like os-prober. Its menu displays text-based options showing multiple OS options and recovery modes. It supports chainloading for other boot loaders. It is the most flexible and works on BIOS & UEFI. On the other hand, Windows Boot Manager uses configuration data or boot.ini to list OS entries. It displays a simple text-based menu but can only detect Windows OS's. Lastly, rEFInd automatically scans the EFI System Partition to detect all available bootable OSes. It displays ga raphical, user-friendly icon-based menu. Works only on UEFI systems and supports chainloading too.

4.3 Common Challenges

Firstly, if one OS is installed in BIOS/MBR mode and another in UEFI/GPT mode, the bootloader may not detect the other OS. Secondly, bootloaders may not automatically detect other OSes, like GRUB disables its auto-detection by default for security purposes. Also, rEFInd can miss OSes if 32/64 bit firmware doesn't match. Furthermore, Windows cannot be booted directly by Linux bootloaders; they need their own bootloader to start. Lastly, if Linux's bootloader, i.e., shim, isn't trusted, the Secure Boot program won't let it start.

4.4 Solution to OS detection & Chainloading

If using GRUB, enabling the os-prober will solve the OS detection issue. In Windows Boot Manager, copying the Windows EFI directory to the ESP that the system boot scans will solve the problem of it not detecting other Windows OSs that are on another disk partition or EFI system partition.

When chainloading, if a firmware mismatch occurs, then keeping all OSes in UEFI/GPT or all in BIOS/MBR will solve the problem. To stop Secure Boot from disrupting the chainloading, using signed bootloaders or updating the shim will solve the issue.

5. Conclusion

To summarize, modern bootloaders like BIOS and UEFI systems have the same purpose, which is to initialize hardware components and then load the operating system, but they differ significantly in architecture, memory access, partition support, and security. If we recall, their key difference is that BIOS relies on MBR, which serves for 16-bit operations, while UEFI uses GPT, which operates in 32/64-bit mode, making a huge difference in how modern computers are built and used. Then, the multiboot systems using bootloaders like GRUB, rEFInd, and Windows Boot Manager have made life easier for users to have multiple operating systems on the same device, and managing between them has become easier due to chainloading methods. Though they have some challenges, those are not unsolvable, but rather easily manageable. Therefore, understanding these modern bootloader systems is essential for system administrators, developers, and advanced users, especially for CSE students like us.