ISLAMIC UNIVERSITY OF TECHNOLOGY

Board Bazar, Gazipur, Dhaka.

Assignment : Booting an Operating System
Operating Systems : CSE 4501

Name: Namisa Najah Raisa
Student ID: 210042112

Fifth Semester
Department of Computer Science and Engineering
BSc. in Software Engineering

October 5, 2024

Contents

1 Introduction

1.1 Booting

Booting is the process by which a computer system initializes and starts up its operating system (OS) after being powered on. It is a crucial phase because without it, the hardware would not be able to interact with the software, and the system would remain unusable. The boot process ensures that all hardware components are operational and that the OS is loaded into memory to manage system resources and execute user applications.

The boot process involves several stages, each responsible for a specific task, starting from powering on the system to handing over control to the operating system. Some of the key components in this process include the boot loader, BIOS (Basic Input/Output System), and the kernel.

1.2 Boot Loader

A small program responsible for loading the operating system into memory. It typically resides in the Master Boot Record (MBR) or the Unified Extensible Firmware Interface (UEFI) partition. The boot loader's main task is to locate the OS kernel and transfer control to it.

1.3 BIOS

The BIOS is a firmware embedded on the motherboard of a computer. It is the first code that runs when the computer is powered on, performing essential hardware checks, initializing hardware components, and locating the bootable device to pass control to the boot loader.

1.4 Kernel

The kernel is the core component of the operating system. It manages communication between hardware and software, handling tasks such as memory management, process scheduling, and hardware control. After being loaded by the boot loader, the kernel takes control of the system to initialize the OS.

Understanding these components and their roles helps in comprehending how a system transitions from a powered-off state to a fully functional OS, which is vital for both troubleshooting and optimizing system performance.

2 Stages of Booting

The booting process is divided into multiple stages, each playing a crucial role in preparing the system to load the operating system and run applications. Below are the major stages involved:

2.1 Power-On Self-Test (POST)

The Power-On Self-Test (POST) is the initial diagnostic test performed by the BIOS or UEFI firmware when a computer is powered on. The main purpose of POST is to ensure that all the essential hardware components—such as the CPU, memory, and storage devices—are functioning correctly.

2.1.1 What happens during POST:

- **Hardware Check:** POST checks critical hardware components, such as the CPU, RAM, and input/output devices. If a hardware issue is detected, the system may emit a series of beeps (POST codes) to indicate the failure type.
- **Peripheral Initialization:** It initializes peripheral devices such as the keyboard, mouse, and video card. System Memory Test: RAM is tested to ensure that memory cells are functioning properly.

- **Drive Detection:** POST detects storage devices such as hard drives or SSDs to identify bootable partitions.

2.1.2 Significance of POST

- **System Integrity:** POST ensures that all essential components are in good working condition before proceeding to boot the operating system. A failure during POST halts the boot process and alerts the user about faulty hardware, protecting the system from malfunction.
- **Error Indication:** If an error is detected, the system typically emits beep codes or displays error messages to assist with troubleshooting. POST failure prevents the boot process from continuing until the issue is resolved.

2.2 Boot Loder

Once the POST is successfully completed, the BIOS or UEFI searches for a bootable device, such as a hard drive, SSD, or USB drive. The bootable device contains the boot loader, a small program responsible for loading the operating system. The boot loader is typically located in the Master Boot Record (MBR) or UEFI partition of the bootable device.

2.2.1 Function of the Boot Loader:

- **Loading the OS Kernel:** The boot loader's primary function is to locate the operating system kernel, load it into memory, and transfer control to it. It reads configuration files to determine where the kernel is located on the disk and loads the necessary components.
- **Multi-booting:** Some boot loaders, such as GRUB (GRand Unified Bootloader), allow users to choose between multiple operating systems installed on the same machine. GRUB can present a menu during boot-up, allowing the user to select the desired OS.
- **Passing Parameters:** Boot loaders can pass parameters to the kernel, such as boot-time configurations or kernel options that define how the OS will behave during startup.

2.2.2 Types of Boot Loaders

1. **GRUB (GRand Unified Bootloader):** GRUB is a popular boot loader for Linux systems, known for its flexibility and ability to support multiple operating systems. GRUB can load operating systems with complex configurations and provide recovery options.
2. **NTLDR (NT Loader):** NTLDR was used in older versions of Windows (Windows NT, 2000, XP). It is responsible for loading the Windows operating system by reading the boot.ini file and locating the Windows kernel.
3. **LILO (Linux Loader):** LILO is another older boot loader used in Linux systems. It is less flexible than GRUB, as it cannot easily handle changes to partitions or kernels without manual updates.

2.2.3 How the Boot Loader Works

1. **Executed by BIOS/UEFI:** After POST, the BIOS/UEFI looks for a bootable device. If it finds one, it executes the boot loader stored in the MBR or UEFI partition.
2. **Locates Kernel:** The boot loader locates the operating system kernel, typically stored on the disk, and loads it into memory.
3. **Transfers Control:** Once the kernel is loaded, the boot loader transfers control to the kernel, which then takes over system operations.

2.3 Loading the Kernel

Once the boot loader has successfully loaded the operating system kernel into memory, the system transitions into the next critical stage: initializing the kernel and preparing the hardware and software environment for user applications.

2.3.1 Process of Loading the Kernel

- **Decompressing the Kernel:** On most systems, the kernel is compressed to save space. The first task after loading the kernel into memory is to decompress it.
- **Hardware Initialization:** The kernel takes control of the system and begins initializing the hardware components, such as the CPU, memory, and I/O devices. The kernel uses drivers to ensure that all connected hardware is recognized and can communicate with the operating system.
- **Mounting the Root Filesystem:** After hardware initialization, the kernel mounts the root filesystem (e.g., / in Unix-like systems) from the storage device, which contains the essential files needed to operate the system.
- **Starting Init Process:** The kernel then starts the init process (or systemd on modern Linux systems), which is the first process in the operating system and has the process ID of 1 (PID 1). This process is responsible for starting and managing other system processes, including background services (daemons) and the user login prompt.

2.3.2 Kernel's Role

- **Managing Hardware:** The kernel ensures smooth communication between hardware and software through device drivers and the management of system resources.
- **Memory Management:** The kernel oversees how memory is allocated to different processes, ensuring that each application gets the necessary resources.
- **Process Management:** The kernel schedules and manages multiple processes, ensuring efficient multitasking and CPU usage.
- **Security:** The kernel enforces security by managing access controls and user permissions, ensuring that processes do not interfere with one another and that sensitive data is protected.

By the end of the kernel loading process, the system is fully operational, with the operating system ready to launch user applications. The kernel's initialization lays the foundation for everything the OS does, from running applications to interacting with hardware.

3 Detailed Flowchart

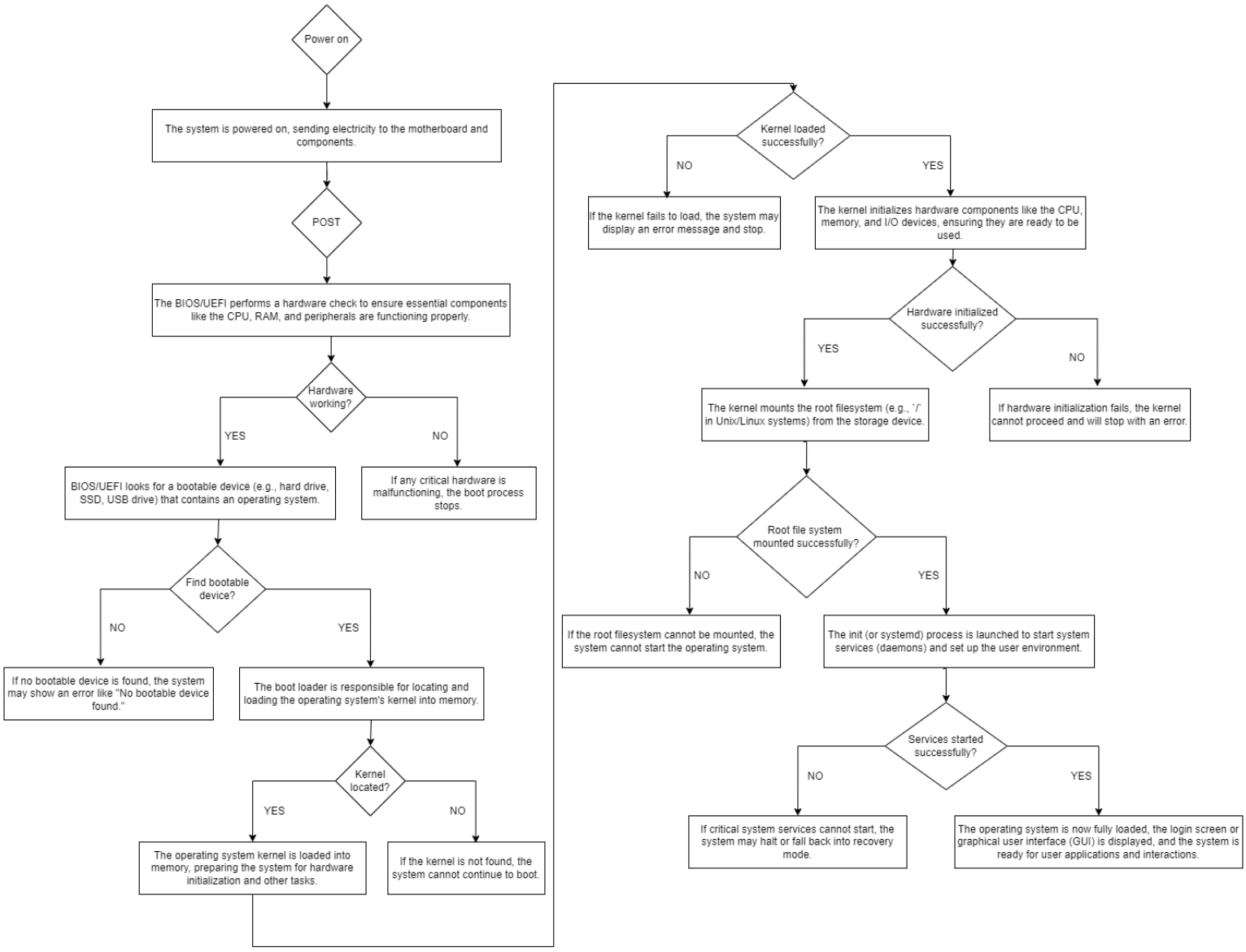

Figure 1: Detailed flowchart

4 Boot Process in Different Operating Systems

While the boot process follows a similar overarching structure across different operating systems, such as loading the operating system into memory and initializing the kernel, the specific steps, boot loaders, and system components can vary significantly. Here's a comparison of the booting process in Windows and Linux, highlighting both similarities and differences.

4.1 Windows Boot Process

The boot process in modern Windows operating systems, such as Windows 10 and Windows 11, involves several stages. The main components include the BIOS/UEFI, Boot Manager, Windows Boot Loader, and the Windows Kernel. **Steps in the Windows Boot Process**

1. **BIOS/UEFI Initialization:** When the system is powered on, the BIOS or UEFI firmware performs the Power-On Self-Test (POST) to check hardware integrity. The firmware then identifies the bootable device by checking the boot sequence.

2. **Windows Boot Manager (Bootmgr):** After POST, the firmware loads the Windows Boot Manager from the boot sector (in legacy BIOS) or the EFI partition (in UEFI-based systems). The Boot Manager is responsible for identifying and executing the correct boot loader.

3. **Windows Boot Loader (Winload.exe):** Once the Windows Boot Manager is loaded, it identifies the operating system to boot. It then loads the Windows Boot Loader, Winload.exe, which is responsible for loading the Windows kernel (ntoskrnl.exe) and essential drivers.

4. **Loading the Kernel:** The boot loader (Winload.exe) loads the Windows kernel (ntoskrnl.exe) and essential system files into memory. It also loads the HAL (Hardware Abstraction Layer), which ensures compatibility between hardware and the operating system. The kernel then initializes system resources, such as memory and hardware drivers.

5. **Windows Session Initialization:** The kernel starts the Session Manager Subsystem (smss.exe), which loads system environment settings, starts services, and initializes the login screen (through winlogon.exe). The user can then log in, and the system is ready for use.

4.2 Linux Boot Process

The Linux boot process also involves multiple stages, but the specifics differ, especially in the boot loader and kernel initialization. The main components include the BIOS/UEFI, Boot Loader (GRUB/LILO), Linux Kernel, and Init Process (systemd). **Steps in the Linux Boot Process:**

1. **BIOS/UEFI Initialization:** Like Windows, Linux systems start with the BIOS/UEFI performing POST and checking the system hardware. The BIOS/UEFI firmware then identifies the bootable device, typically containing a Linux boot loader.

2. **Boot Loader (GRUB):** After POST, the BIOS/UEFI loads the boot loader, usually GRUB (GRand Unified Bootloader), from the Master Boot Record (MBR) or UEFI partition. GRUB provides a menu allowing users to select between different installed kernels or operating systems (multi-boot).

3. **Loading the Kernel:** After selecting the OS, GRUB loads the Linux kernel into memory. The kernel is often compressed, so the first step is decompressing it. The boot loader also loads an initial RAM disk (initramfs or initrd) into memory, which contains essential drivers and modules needed to initialize the system.

4. **Kernel Initialization:** The kernel initializes hardware components, mounts the root filesystem, and loads essential drivers. The Linux kernel also starts process scheduling and memory management.

5. **Init Process (systemd):** The kernel hands control to systemd, the modern init system used in most Linux distributions. Systemd is responsible for managing system services and running scripts to prepare the system for users. It launches essential background services (daemons) and prepares the user environment, including starting a graphical login screen.

4.3 Similarities

1. **BIOS/UEFI Initialization:** Both Windows and Linux start the boot process with BIOS/UEFI, which performs POST and identifies the bootable device.

2. **Boot Loader Role:** Both operating systems use a boot loader to locate and load the kernel. While Windows uses the Boot Manager and Winload.exe, Linux primarily uses GRUB.

3. **Kernel Loading:** Both OSs load their respective kernels into memory and begin system initialization.

4.4 Differences

1. **Boot Loader Flexibility:** Linux's GRUB is more flexible and allows for easy multi-booting between different OSs and kernels. Windows Boot Manager is more restricted but allows recovery options like Safe Mode.

2. **Init system:** Windows uses the Session Manager and other system components to start the system, whereas Linux uses systemd, which is more modular and configurable.

3. **Initial RAM disk:** Linux uses an initial RAM disk (initramfs/initrd) to load necessary drivers before the root filesystem is mounted, a process not seen in Windows booting.

5 Conclusion

The booting process is a critical phase in a computer's operation, transitioning the system from a powered-off state to a fully functional operating system ready for user applications. This process involves several key components, including the POST for hardware integrity checks, the boot loader for locating and loading the operating system kernel, and the kernel initialization to prepare the system's hardware and software environments.

In both Windows and Linux, the boot process starts with BIOS/UEFI firmware performing a Power-On Self-Test (POST) to verify hardware functionality. The boot loader then plays a central role by loading the operating system's kernel into memory, whether it's Windows Boot Manager or GRUB for Linux. Each OS uses its unique kernel and initialization process to handle hardware, system resources, and load essential services.

While Windows booting focuses on ease of use and system recovery options, Linux provides more flexibility, allowing users to select different kernels and operating systems. Despite the differences, both operating systems prioritize efficient hardware-software interaction, essential for proper system functioning.

The booting process is crucial for system stability and security. By ensuring that all hardware components are functioning properly before the operating system loads, the system can avoid potential failures and ensure smooth operation. The kernel's role in initializing system components and managing resources is essential for performance and multitasking capabilities. Additionally, secure boot mechanisms help prevent unauthorized modifications to the boot loader or kernel, thus safeguarding the system from security threats.

In conclusion, the boot process is a foundational element of operating system functionality. It not only initializes the system but also ensures that the underlying hardware and software can work in harmony to provide a stable, secure computing environment for users.