Operating System: Boot and UEFI

August 1, 2023

1 Linux

The start-up of a Linux operating system follows a step-by-step process. This process starts with the power on or by running a command in the terminal.

1.1 BIOS and UEFI

First, the BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface) program kicks in once the system powers up. Usually, the BIOS contains all the code to gain initial access to the main machine devices:

1. keyboard
2. display screen
3. disk drives
4. serial communications

However, most of these devices will have dedicated device drivers taking over once the system boots fully.
BIOS and UEFI are firmware interfaces that computers use to boot up the operating system (OS). Yet, the two programs differ in how they store metadata on and about the drive:

1. BIOS uses the Master Boot Record (MBR)
2. UEFI uses the GUID Partition Table (GPT)

Next, the BIOS or UEFI runs the power-on self-test (POST). The POST does a series of tasks:

1. verify the hardware components and peripherals
2. carry out tests to ensure that the computer is in proper working condition
3. Further, if this test finds any error, it commonly shows an error message on the screen. In case the test fails to detect the RAM, POST triggers a beeping sound.

Finally, if the system passes the POST, it signals the start-up process to the next stage.

1.2 Boot Loader

The BIOS or UEFI has run the POST to check the state of the machine. Moving on, the BIOS/UEFI selects a boot device depending on the system configuration. Usually, there's a default boot order:

1. Hard drives
2. USB drives
3. CD drives

Of course, we can configure the BIOS or UEFI to choose the boot device in any order. Whatever their order, the computer looks for the boot loader in these devices one by one.
A BIOS system has the boot loader located in the first sector of the boot device: the MBR. It takes up the first 512 bytes on the disk. On the other hand, a UEFI system stores all startup data in an `.efi` file. The file is on the EFI System Partition, which contains the boot loader.
The boot loader is a small program that loads the operating system. The main job of the boot loader is to perform three actions with the kernel: locate on the disk, insert into memory, and execute with the supplied options.
The following are some of the available boot loaders for a Linux system:

1. LILO
2. SYSILINUX
3. GRUB2

In general, once the computer finds the boot loader on a device, it will run it. Consequently, this loads a larger and more complex program which eventually starts our operating system.

1.3 GRUB2

Almost all (non-embedded) modern Linux distributions use GRUB (GRand Unified Boot Loader) because it's very feature-rich:

1. ability to boot multiple operating systems
2. boots both a graphical and a text-based interface
3. allows ease of use over a serial cable
4. strong command line interface for interactive configuration
5. network-based diskless booting

This is what GRUB2 does in the boot process:

1. takes over from BIOS or UEFI at boot time
2. loads itself
3. inserts the Linux kernel into memory
4. turns over execution to the kernel

Notably, the GRUB configuration file is located at `/boot/grub` by default:
GRUB2 inserts the kernel into memory and turns control of the system over to the kernel.

1.4 Kernel

After going through BIOS or UEFI, POST, and using a boot loader to initiate the kernel, the operating system now controls access to our computer resources.
Here, the Linux kernel follows a predefined procedure:

1. decompress itself in place
2. perform hardware checks
3. gain access to vital peripheral hardware
4. run the init process

Next, the init process continues the system startup by running init scripts for the parent process. Also, the init process inserts more kernel modules (like device drivers).

1.5 Systemd

To reiterate, the kernel initiates the init process, which starts the parent process. Here, the parent of all Linux processes is `Systemd`.Following the booting steps, Systemd performs a range of tasks: To reiterate, the kernel initiates the init process, which starts the parent process. Here, the parent of all Linux processes is Systemd.Following the booting steps, Systemd performs a range of tasks:

1. probe all remaining hardware
2. mount filesystems
3. initiate and terminate services
4. manage essential system processes like user login
5. run a desktop environment

Indeed, these and other tasks allow users to interact with the system. Lastly, Systemd uses the /etc/systemd/system/default.target file to decide the state or target the Linux system boots into.

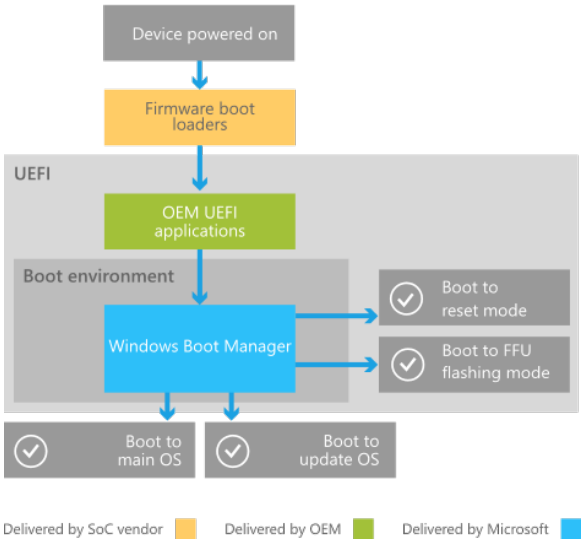
2 Windows

2.1 Overview of the boot process

When a Windows 10 device is turned on, it goes through the following high-level process:

1. The device is powered on and runs the specific firmware boot loaders, which initialize the hardware on the device and provide emergency flashing functionality.
2. The firmware boot loaders boot the UEFI environment and hands over control to UEFI applications written by the chip vendor, Microsoft, and OEMs. These applications can utilize UEFI drivers and services.

3. The UEFI environment launches the Windows Boot Manager, which determines whether to boot to Full Flash Update (FFU) image flashing or device reset mode, to the update OS, or to the main OS.



2.2 boot loaders

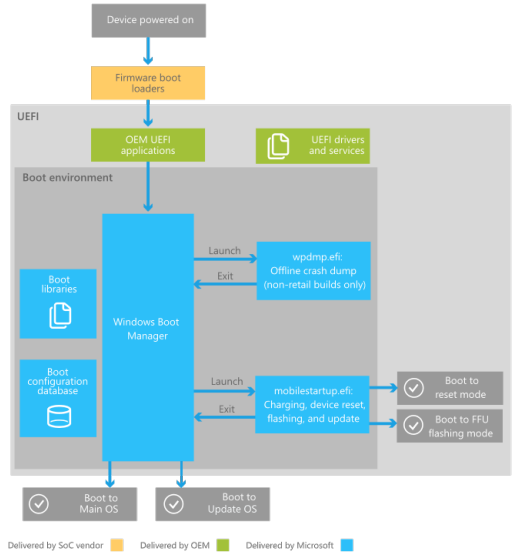
The firmware boot loaders initialize the minimal set of hardware required for the device to run. The chip firmware boot loaders are designed to finish as fast as possible, and nothing is drawn to the screen while they are running. After the chip firmware boot loaders finish, the device is booted into the UEFI environment.
The firmware boot loaders also contain an emergency flashing capability that allows devices to be flashed when the boot environment is not stable and Full Flash Update (FFU) image-based flashing using the Microsoft-provided flashing tool is not possible. Emergency flashing requires tools specific to the chip.

2.3 UEFI

Windows 10 utilizes the Unified Extensible Firmware Interface (UEFI) to support the hand-off of system control from the chip firmware boot loader to the OS. The UEFI environment is a minimal boot OS upon which devices are booted and the Windows 10 OS runs.

2.4 Understanding the Windows Boot Manager

The Windows Boot Manager is a Microsoft-provided UEFI application that sets up the boot environment. Inside the boot environment, individual boot applications started by the Boot Manager provide functionality for all customer-facing scenarios before the device boots.
The following diagram illustrates some of the key portions of the process that the Boot Manager follows after it is launched by the UEFI environment.



The following steps describe this process in more detail:

1. After the UEFI environment launches the Boot Manager, the Boot Manager initializes boot libraries, reads the boot configuration database to determine which boot applications to run and in which order to run them. The Boot Manager launches boot applications sequentially, and each application exits back to the Boot Manager after finishing.

   Boot libraries are libraries of functions that extend upon existing UEFI functionality, and are designed to be used within the boot environment. Only boot applications, which are launched by the Boot Manager, have access to the boot libraries.

2. The Boot Manager first captures any reserved hardware button combinations that are pressed by the user.

3. In non-retail OS images, the Boot Manager next runs an offline crash dump boot application which allows the device to capture a snapshot of physical memory from the previous OS session. When the device resets abnormally, the previous OS session's memory is preserved across the reset. When this happens, the offline crash dump application will save that memory and turn it into an offline crash dump file, which can be transferred off the device and analyzed. If the device did not reset abnormally in the previous OS session, the offline crash dump application exits immediately.

4. In all OS images, the Boot Manager next runs mobilestartup.efi. This application runs several boot libraries, some of which are only run on first boot (for example, to provision the secure boot policy) or only in non-retail images (for example, to enter USB mass storage mode). The following libraries are always run:

   First, mobilestartup.efi runs the library that implements UEFI battery charging. This library allows the user to charge their device while the device is in the boot environment (or is perceived as being turned off). This library is run first to ensure that the device has enough power to fully boot. For more information about scenarios involving the battery charging application, see Battery charging in the boot environment.

Next, mobilestartup.efi runs the libraries that implement flashing, device reset, and updates. These libraries determine whether the device should boot to flashing or device reset mode, or if the device should continue to the Update OS or Main OS.

5. If mobilestartup.efi does not boot to flashing or device reset mode, the Boot Manager boots into the Main OS or the Update OS.