AUGUST 10, 2020 / #COMPUTER SCIENCE

UEFI vs BIOS: What's the Difference?

By Vaibhav Kandwal

So you might have heard the acronyms BIOS and UEFI thrown around, especially when trying to switch Operating Systems or messing around with overclocking.

And you might know what these acronyms stand for (Unified Extensible Firmware Interface and Basic Input/Output System, respectively). But have you ever wondered how they're used in a computer system?

Let's demystify these terms and their meanings now.

Boot Procedure

First things first – I know we're deviating from the topic, but I promise this will help you with some concepts later on.

So, how does a computer boot? Let's go step by step:

1. You press the power button on your laptop/desktop.
2. The CPU starts up, but needs some instructions to work on (remember, the CPU always needs to do something). Since the main memory is empty at this stage, CPU defers to load instructions from the firmware chip on the motherboard and begins executing instructions.
3. The firmware code does a Power On Self Test (POST), initializes the remaining hardware, detects the connected peripherals (mouse, keyboard, pendrive etc.) and checks if all connected devices are healthy. You might remember it as a 'beep' that desktops used to make after POST is successful.
4. Finally, the firmware code cycles through all storage devices and looks for a boot-loader (usually located in first sector of a disk). If the boot-loader is found, then the firmware hands over control of the computer to it.

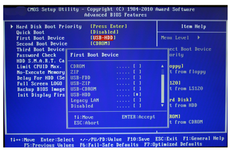
We don't need to know more about this topic for the purposes of this article. But if you're interested, then read on (otherwise, you can skip to next section).

5. So now that the boot-loader is loaded, its job is to load the rest of the operating system. GRUB is one such boot-loader that is capable of loading unix-like operating systems and is also able to chain-load GRUB Boot Loader loads Windows Boot Loader to continue Windows OS. Boot-loader is only available in the first sector of a disk, which is 512 bytes. Given the complexity of modern operating systems, some of these boot-loaders tend to do multi-stage loading, where the main boot-loader loads the second-stage-boot-loader in an environment which is not restricted to 512 bytes.
6. The boot-loader then loads the kernel into memory. Unix-like operating systems then run the `init` process (the master process, from which other processes are forked/executed) and finally initialize the run-levels.
7. In Windows, `wininit.exe` is loaded along with some other processes like `services.exe` for service control, `lsass.exe` for local security and authority (similar to run-levels) and `lsm.exe` for local session management.
8. After all this, and after some other drivers are initialized, the Graphical User Inferface (GUI) is loaded and you are presented with the login screen.

This was a very high-level overview of the boot process. If you're interested in Operating Systems, I would recommend that you read more on osdev.net.

Now let's get back to our original topic.

BIOS:



BIOS stands for Basic Input/Output System, the firmware we talked about in the above boot procedure.

It is stored on an EPROM (Erasable Programmable Read-Only Memory), allowing the manufacturer to push out updates easily.

It provides many helper functions that allow reading boot sectors of attached storage and printing things on screen. You can access BIOS during the initial phases of the boot procedure by pressing `del`, `F2` or `F10`.

UEFI:

ASUS UEFI

UEFI stands for Unified Extensible Firmware Interface. It does the same job as a BIOS, but with one basic difference: it stores all data about initialization and startup in an *.efi* file, instead of storing it on the firmware.

This *.efi* file is stored on a special partition called EFI System Partition (ESP) on the hard disk. This ESP partition also contains the bootloader.

UEFI was designed to overcome many limitations of the old BIOS, including:

1. UEFI supports drive sizes upto 9 zettabytes, whereas BIOS only supports 2.2 terabytes.
2. UEFI provides faster boot time.
3. UEFI has discrete driver support, while BIOS has drive support stored in its ROM, so updating BIOS firmware is a bit difficult.
4. UEFI offers security like "Secure Boot", which prevents the computer from booting from unauthorized/unsigned applications. This helps in preventing rootkits, but also hampers dual-booting, as it treats other OS as unsigned applications. Currently, only Windows and Ubuntu are signed OS (let me know if I am wrong).
5. UEFI runs in 32bit or 64bit mode, whereas BIOS runs in 16bit mode. So UEFI is able to provide a GUI (navigation with mouse) as opposed to BIOS which allows navigation only using the keyboard.

You might not need UEFI

Though all modern computers come equipped with UEFI by default, some reasons why you might choose BIOS over UEFI are:

1. If you're beginner and don't care about messing with any type of firmware, BIOS is for you.
2. If you have < 2 TB per hard disk or partition, you can go with BIOS.
3. BIOS allows running multiple operating systems without changing any settings This can be a security issue from a modern standpoint, but hey, no hassles for the user.
4. BIOS provides system information to the operating system. So if your OS runs in 16 bit mode, it does not require writing code for interacting with hardware. It can directly use methods provided by BIOS. Else if the OS switches over to 32bit or 64bit mode, then it needs to provide its own subroutines for interacting with hardware.
5. If you are someone who prefers a keyboard and text based UI over navigation with a mouse and GUI, then BIOS is for you.

UEFI takes these limitations into account and provides a Legacy mode. In it you can run everything as if you had a BIOS firmware. But keep in mind that Intel has announced that it won't support traditional BIOS from 2020.

Conclusion

This post gave you an overview of the differences between BIOS and UEFI. It also advises you when to choose either one of them and how they are different from each other.

If you have any questions, I will always be available on Twitter. Thank you for your time.

If you read this far, thank the author to show them you care. Say Thanks

Learn to code for free. freeCodeCamp's open source curriculum has helped more than 40,000 people get jobs as developers. Get started