UD 07 - Understanding the Boot Process

Ranvir Sing - Angel Berlanas

December 17, 2019

Contents

1 Abstract

The boot process is universe unto its own. A lot of steps are needed to happen before your operating system takes over and you get a running system. In some sense, there is a tiny embedded OS involved in this whole process. While the process differs from one hardware platform to another, and from one OS to another, let's look at some of the commonalities that will help us gain a practical understanding of the boot process.

Let's talk about the regular, non-UEFI, boot process first. What happens between that point in time where you press the power ON button to the point where your OS boots and presents you with a login prompt.

2 Legacy BIOS — Basic Input/Output System

Step1: The CPU is hardwired to run instructions from a physical component, called NVRAM or ROM, upon startup. These instructions constitute the system's firmware. And it is this firmware where the distinction between BIOS and UEFI is drawn. For now let's focus on BIOS.

It is the responsibility of the firmware, the BIOS, to probe various components connected to the system like disk controllers, network interfaces, audio and video cards, etc. It then tries to find and load the next set of bootstrapping code.

The firmware goes through storage devices (and network interfaces) in a predefined order, and tries to find a bootloader stored within them. This process is not something a user typically involves herself with. However, there's a rudimentary UI that you can use to tweak various parameters concerning the system firmware, including the boot order.

You enter this UI by typically holding F12, F2 or DEL key as the system boots. To look for specific key in your case, refer your motherboard's manual.

Step2: BIOS, then assumes that the boot device starts with an MBR (Master Boot Record) which containers a first-stage boot loader and a disk partition table. Since this first block, the boot-block, is small and the bootloader is very minimalist and can't do much else, for example, read a file system or load a kernel image.

So the second stage bootloader is called into being.

Step3: The second stage bootloader is responsible for locating and loading the proper Operating System kernel into the memory. The most common example, for Linux users, is the GRUB bootloader. In case you are dual-booting, it even provider you with a simple UI to select the appropriate OS to start.

Even when you have a single OS installed, GRUB menu lets you boot into advanced mode, or rescue a corrupt system by logging into single user mode. Other operating systems have different boot loaders. FreeBSD comes with one of its own so do other Unices.

Step4: Once the appropriate kernel is loaded, there's still a whole list of userland processes are waiting to be initialized. This includes your SSH server, your GUI, etc if you are running in multiuser mode, or a set of utilities to troubleshoot your system if you are running in single user mode.

Either way an init system is required to handle the initial process creation and continued management of critical processes. Here, again we have a list of different options from traditional init shell scripts that primitive Unices used, to immensely complex systemd implementation which has taken over the Linux world and has its own controversial status in the community. BSDs have their own variant of init which differs from the two mentioned above.

This is a brief overview of the boot process. A lot of complexities have been omitted, in order to make the description friendly for the uninitiated.

3 UEFI specifics

The part where UEFI vs BIOS difference shows up is in the very first part. If the firmware is of a more modern variant, called UEFI, or Unified Extensible Firmware Interface, it offers a lot more features and customizations. It is supposed to be much more standardized so motherboard manufacturers don't have to worry about every specific OS that might run on top of them and vice versa.

One key difference between UEFI and BIOS is that UEFI supports a more modern GPT partitioning scheme and UEFI firmware has the capability to read files from a small FAT system.

Often, this means that your UEFI configuration and binaries sit on a GPT partition on your hard disk. This is often known as ESP (EFI System Partition) mounted at /efi, typically.

Having a mountable file system means that your running OS can read the same file system (and dangerously enough, edit it as well!). Many malware exploit this capability to infect the very firmware of your system, which persists even after an OS reinstall.

UEFI being more flexible, eliminates the necessity of having a second stage boot loader like GRUB. Often times, if you are installing a single (well-supported) operating system like Ubuntu desktop or Windows with UEFI enabled, you can get away with not using GRUB or any other intermediate bootloader.

However, most UEFI systems still support a legacy BIOS option, you can fall back to this if something goes wrong. Similarly, if the system is installed with both BIOS and UEFI support in mind, it will have an MBR compatible block in the first few sectors of the hard disk. Similarly, if you need to dual boot your computer, or just use second stage bootloader for other reasons, you are free to use GRUB or any other bootloader that suits your use case.

4 Conclusion

UEFI was meant to unify the modern hardware platform so operating system vendors can freely develop on top of them. However, it has slowly turned into a bit of a controversial piece of technology especially if you are trying to run open source OS on top of it. That said, it does have its merit and it is better to not ignore its existence.

On the flip-side, legacy BIOS is also going to stick around for at least a few more years in the future. Its understanding is equally important in case

you need to fall back to BIOS mode to troubleshoot a system. Hope this article informed you well enough about both these technologies so that the next time you encounter a new system in the wild you can follow along the instructions of obscure manuals and feel right at home.