An Introduction to UEFI

Post Date: 2020-12-10

Post Reply

Author	Message	Topic Search	Topic Options

ErikW
Newbie
VERIFIED CUSTOMER
Joined: 26 Aug 2020
Online Status: Offline
Posts: 22

I'm a programmer, but I've just recently started using a computer with UEFI firmware. I read all of the UEFI technical information and found that I was clueless about how it has been implemented by computer manufacturers. One reason for that was no good definition of what UEFI is, what it does, and how to use it.

Before talking about UEFI it helps to look at the BIOS firmware that personal computers have been using since the 1980's. The BIOS was originally intended to start up a computer and provide some simple functions to access hardware in a standard way, while hiding most of the hardware implementation details. It was sort of a hardware abstraction layer (HAL). Here are some of the things a BIOS does.

- Startup diagnostics
- Provides information and access to hard disk sectors
- Load and run from the first sector of a hard disk (booting)
- Floppy disk drive sector access
- CD-ROM boot sector access and disk emulation
- Simple text input and output
- A periodic timer interrupt
- Memory information / allocation
- Access / configure PCI / PCIe hardware
- Switch CPU modes and access memory above 1 Mbyte
- Video card firmware startup and access
- Access to firmware on other disk controllers or peripherals

There are a few important things that I will point out about BIOS firmware. A BIOS doesn't understand or know about partitions, file-systems, files or file data types. All the BIOS knows is disk drives and disk sectors. The BIOS is designed to only load and execute a single disk sector. That "boot" sector has to contain all the software and information to load the next part of the boot software. While a BIOS does some coordination between hardware devices, it is not an operating system. There are rules about which devices can be used when, and the BIOS does very little to prevent software from doing two incompatible things at the same time. As hardware has gotten more complicated, the BIOS has been less and less useful for hiding the hardware details from software and maintaining a standard software interface. The BIOS starts up the CPU in Real mode, a limited 16-bit execution mode that used to be the only mode of an Intel x86 CPU. Every bootloader or OS has to switch the CPU into a more useful mode, usually 64-bit protected virtual mode. The BIOS may provide a few helpers for that, but every operating system has to perform a very exact set of steps to get the CPU into a usable mode.

Along with the BIOS, there was eventually a desire for hard disks to support more than just one operating system (DOS) and one filesystem (FAT16). The Master Boot Record (MBR) and Partition Table were added to hard disks to provide that. However, the BIOS knows nothing about partitions or filesystems. The first sector of the hard disk contains a boot sector for the whole disk, and the four-entry partition table. The contents of that sector have to be sufficient to locate, load and execute the first sector of a partition (Partition Boot Sector). It is actually the Master Boot Record, not the BIOS that knows about partitions. Each partition then has to have enough software and information inside its first sector to load the next part of the software (usually a bootloader). The first bootloader was simple (IO.SYS). Things got more complicated until we ended up with things like GRUB 2, or Windows bootmgr. To fit the necessary information into the boot sectors, software has to save information about where certain files are located, and that is complicated, error prone and breaks if files, or pieces of files are moved. Software got around some of that problem using hacks like having software understand part of the filesystem, or hiding extra software between the Master Boot Record and the start of the first disk partition.

It became very obvious that the BIOS needed to be replaced with something better. Intel decided to do that for some of its computers and that was named the Extensible Firmware Interface (EFI). Apple also sort of supported it, but it wasn't a well established standard. When an official standard was created, it was renamed the Universal Extensible Firmware Interface (UEFI).

UEFI is a very different thing than the BIOS.

- Startup diagnostics
- It's an Operating System
- It understands partitions, volumes, and at least one filesystem
- It understands at least one disk partitioning method (GPT)
- It defines a standard for booting from optical discs
- No uniquely created boot sectors or files on each different disk
- It provides a standard for adding programs and drivers
- It provides a standard for bootloader programs
- It really does hide the hardware details from the software
- It provides a standard "shell" with some standard commands
- There is a device and volume naming standard
- It defines the different stages of booting
- It defines a way to create and maintain a boot menu
- It defines a way to store some system dependent information

It may be surprising to think of the UEFI boot firmware as an operating system, but that's what it really is. The file-system that was chosen for UEFI to understand is FAT though there is some confusion about whether that includes FAT-12 for floppies or FAT-16 on some small disks or possibly CD-ROM boot images. The UEFI standard requires support for FAT32, FAT16 and FAT12,

BIOS booting of CD-ROMs had a number of different "image" formats, floppy emulation, hard disk emulation, and no emulation. The BIOS support for emulating floppy disks or hard disks on a CD-ROM was never great, and pretty much no-emulation is the only boot image used by most operating systems. UEFI created its own new boot image identifier and format rather than further extending the older BIOS standard. The UEFI boot image on a CD is simply a FAT filesystem image, just like it would appear in an actual hard disk partition. That means every kind of disk that UEFI boots from uses exactly the same filesystem for the boot files. That FAT filesystem is called the EFI System Partition (ESP) or sometimes system partition.

UEFI also separates the idea of disks and partitions from filesystems and volumes. What's the difference? The disk and partition identifies a hardware device or location of something. The location might change if that "something" is plugged in somewhere else, or maybe moved between computers. The "something" is a volume, a specific set of files and a specific filesystem with some kind of unique identifier. Software can refer to a particular volume and access it no matter where it happens to be located or relocated. It could even move from a local hard disk to a network drive.

The executable file format that UEFI chose is a Microsoft standard called Portable Executable (PE), not to be confused with Windows Preinstallation Environment (Windows PE). UEFI uses a different identifier for the executable files, so they aren't regular Windows executable files. UEFI executable files have the extension .EFI. Windows "bootmgr" can load (chain to) Windows executable files, but not UEFI executable files (other bootloaders). The only real difference is the ID byte saying which OS it runs on. In theory one could create a "Windows" executable bootloader that uses UEFI system functions and chain to it from "bootmgr". It would have to be signed with a digital signature to pass secure boot.

UEFI has a standard set of operating system functions, and it has a format for drivers, so that manufacturers can provide a single solution for accessing their hardware during booting. That helps for things like fake RAID controllers and special storage devices. It also helps for graphics cards.

Even the UEFI shell (text console) commands have a standard, though not every motherboard manufacturer supplies the UEFI shell. Because there is a standard, you can download a UEFI shell that will work on any UEFI computer having a compatible version of the UEFI spec. I found that very helpful for adding firmware boot entries to my MSI motherboard that did not provide the UEFI shell, or an easy way to add firmware boot entries manually.

If you are familiar with DOS or the Windows command line, you will find that the UEFI shell is very similar. It uses the FAT filesystem, so file names and directories are separated with backslashes just like on DOS. Rather than using letters like "C:" for filesystems, UEFI uses numbers prefixed by "FS". For example, "FS0:" or "FS4:.

There are ways to identify other things and map them to the simpler names. Names for block devices (accessed by disk block not filesystem) are "BLK0", "BLK1", etc. UEFI also relies heavily on the ACPI (Advanced Configuration and Power Management Interface) standard for naming things. That is the same standard used for plug-and-play hardware, PCI and PCIe. The information already has to be maintained for operating systems to read, so it made sense to use it in UEFI.

UEFI is guaranteed to understand Global Partition Table (GPT) disks and a FAT-32 EFI System Partition on every device that can be booted. The standard actually permits more than one EFI System Partition on a device, but I wouldn't depend on manufacturers supporting that in their UEFI firmware. Some UEFI firmware implementations support MBR partitioned disks or ISO filesystems on DVDs/CDs. UEFI is supposed to support FAT32, FAT16 and FAT12, but it is probably safest to format the EFI System Partition with FAT32.

UEFI defines a standard set of graphics functions, named GOP (Graphics Output Protocol). The older Intel EFI standard defined a UGA (Universal Graphic Adapter) but that is not supported by UEFI. You can write graphical bootloaders and other programs to run on UEFI before booting another operating system.

So far I haven't mentioned booting on UEFI. It's amazingly simple. Every device that can be booted, has an EFI System Partition (ESP) formatted with the FAT32 filesystem. That is a specific partition type on hard disks and a specific "El-Torito" boot record type on optical discs. The boot record on an optical disk is essentially treated like a block device (partition) with a FAT filesystem in it. The UEFI firmware is supposed to, at a minimum look for the file "\EFI\BOOT\BOOTX64.EFI" in every ESP and add that to the boot menu. The UEFI firmware can also maintain other boot menu entries that refer to files in the EFI System Partitions, such as the familiar "\EFI\MICROSOFT\BOOT\BOOTMGFW.EFI". On a 32-bit UEFI system the file is called "BOOTIA32.EFI" and located in the same folder. Both files can be there. There really can be any number of files named for the supported "architectures".

The last thing that I will mention is backward compatibility. Versions of Windows earlier than Windows 10 (including Windows 7 64-bit) require a few BIOS video functions during booting (even if they are booted using UEFI!) . UEFI firmware may have partial support for BIOS functions, known as the CSM (Compatibility Support Module). That can support booting MBR partitioned disks using BIOS mode bootloaders, or it may only support a few BIOS functions. When there is no CSM support for the needed BIOS graphics functions, Windows 7 will not boot from UEFI. If you try to boot Windows 7 64-bit from a properly formatted GPT disk with an EFI System Partition, you will find that it only works on some UEFI systems. On others it hangs while displaying one of the first graphics screens. The cause is the Windows 7 use of a few BIOS graphics functions and the lack of a CSM that supports those. The CSM is getting rarer as operating systems are adding support for UEFI. Windows 10 can boot on UEFI systems with no CSM. So can Linux.

I hope this wasn't another TLDR time waster, and it shed some light on the mysterious UEFI.

Edited by ErikW - 10 Dec 2020 at 8:54pm

Cretae
DS Veteran

VERIFIED CUSTOMER
Joined: 22 Mar 2010
Online Status: Offline
Posts: 7195

Appreciate the time you put into this. Thanks for the enlightenment. I think many will find it interesting.

hoserator
DS Veteran
We don't need no stinking "Avatars"!
VERIFIED CUSTOMER
Joined: 08 Oct 2014
Online Status: Offline
Posts: 7779

When is the quiz?

Excellent!

Cretae
DS Veteran

VERIFIED CUSTOMER
Joined: 22 Mar 2010
Online Status: Offline
Posts: 7195

Post Reply

Forum Jump -- Select Forum --

Forum Permissions