**Basic Input/output System (BIOS)**

A basic input/output system (BIOS) is a preinstalled program used during startup on Windows-based computers. The CPU initially accesses the BIOS, after which the operating system is loaded.

A basic input/output system is also known as system BIOS or ROM BIOS.

The BIOS is built-in software that contains generic code required to control the keyboard, display screens, disk drives and other functions. The primary purpose of the BIOS is to set up hardware and further load and start an operating system. BIOS is placed in a nonvolatile ROM chip inside the computer, ensuring the availability of BIOS at all times and preventing accidental disk failure. The BIOS checks every hardware connection and locates the devices, after which the operating system is loaded into computer memory.

BIOS software is designed to work with the various devices that make up a complimentary system chipset. The BIOS library has certain functions used to operate and control system peripherals, which can be initiated by external software.

Users using the BIOS user interface can perform functions such as:

* Setting the system clock
* Enabling and disabling certain system components
* Hardware configuration
* Selecting boot drives
* Set password prompts for secured access to BIOS user interface function

Modern PCs have BIOS stored in rewritable memory, permitting contents to be rewritten or replaced. Such content rewriting is called flashing and is executed through a special program provided by system manufacturers.

**Purpose of BIOS**

BIOS enables computers to perform certain operations as soon as they are turned on. The principal job of a computer's BIOS is to govern the early stages of the startup process, ensuring that the operating system is correctly loaded into memory. BIOS is vital to the operation of most modern computers, and knowing some facts about it could help you troubleshoot issues with your machine.

**Unified Extensible Firmware (UEFI)**

It is an upgrade for the traditional computer BIOS that Intel started developing as EFI. UEFI is an update that includes improved boot times, better security, and better support for hardware found in today's computers.

While the UEFI system has major improvements over BIOS, it is still somewhat restricted due to the processor architecture. 64-bit processors have full support for UEFI systems, while x86 processors will have partial or no support and the OS must emulate a BIOS environment for them to work. When this happens, many of the extra features are lost. Microprocessor manufacturers and operating system developers have been working together to eliminate this problem, and to some extent, they have succeeded. Apple, Intel, AMD, Dell and others have given a lot of thought to UEFI implementation, and Microsoft also has added full support for UEFI in Windows 8.

**The differences of UEFI versus BIOS**

Here is a list of the main differences between UEFI and BIOS. It is about the characteristics that have been added in the first so that it is not limited to replace the second, but also to improve it significantly.

* The most notable difference for the average user between both firmwares is in appearance. The BIOS has a very MS-DOS design, and you can only move through it using the keyboard. The UEFI instead has a much more modern interface, allows you to include animations and sounds, and allows you to use the mouse to interact with it.
* Under the hood, the UEFI code runs in 32 or 64 bits, while the BIOS usually do it in 16 bits.
* Systems with BIOS only support up to four partitions and hard disks with a maximum capacity of 2.2 TB. That’s because they use the MBR partition scheme. UEFI for its part uses a more modern GPT, which puts the theoretical limit of hard drives capacities in 9.4 zettabytes, although at the moment no such large one is manufactured.
* The boot of the computer is faster with UEFI than it was with BIOS.
* UEFI also tries to improve security with its Secure Boot functionality. It is a safe boot that started using Windows 8 with considerable controversy, and prevents the start of operating systems that are not authenticated to protect you from bootkits, a malware that runs when Windows starts.
* And finally, the memory chip in which the UEFI is installed is not blocked on the motherboard as BIOS. This means that you can add third-party extensions, such as overclocking tools or diagnostic software.

**Booting process**

Windows

1. System is powered on
2. The CMOS loads the BIOS and then runs POST
3. Looks for the MBR on the bootable device
4. Through the MBR the boot sector is located and the BOOTMGR is loaded
5. BOOTMGR looks for active partition
6. BOOTMGR reads the BCD file from the \boot directory on the active partition
7. The BCD (boot configuration database) contains various configuration parameters( this information was previously stored in the boot.ini)
8. BOOTMGR transfer control to the Windows Loader (winload.exe) or winresume.exe in case the system was hibernated.
9. Winloader loads drivers that are set to start at boot and then transfers the control to the windows kernel.

Linux

Six stages of Linux boot process

**1. BIOS**

* BIOS stands for Basic Input/Output System
* Performs some system integrity checks
* Searches, loads, and executes the boot loader program.
* It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 of F2, but it depends on your system) during the BIOS startup to change the boot sequence.
* Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
* So, in simple terms BIOS loads and executes the MBR boot loader.

**2. MBR**

* MBR stands for Master Boot Record.
* It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
* MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
* It contains information about GRUB (or LILO in old systems).
* So, in simple terms MBR loads and executes the GRUB boot loader.

**3. GRUB**

* GRUB stands for Grand Unified Bootloader.
* If you have multiple kernel images installed on your system, you can choose which one to be executed.
* GRUB displays a splash screen, waits for few seconds, if you don’t enter anything, it loads the default kernel image as specified in the grub configuration file.
* GRUB has the knowledge of the filesystem (the older Linux loader LILO didn’t understand filesystem).
* Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this). The following is sample grub.conf of CentOS.
* As you notice from the above info, it contains kernel and initrd image.
* So, in simple terms GRUB just loads and executes Kernel and initrd images.

**4. Kernel**

* Mounts the root file system as specified in the “root=” in grub.conf
* Kernel executes the /sbin/init program
* Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a ‘ps -ef | grep init’ and check the pid.
* initrd stands for Initial RAM Disk.
* initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

**5. Init**

* Looks at the /etc/inittab file to decide the Linux run level.
* Following are the available run levels
* 0 – halt
* 1 – Single user mode
* 2 – Multiuser, without NFS
* 3 – Full multiuser mode
* 4 – unused
* 5 – X11
* 6 – reboot
* Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.
* Execute ‘grep initdefault /etc/inittab’ on your system to identify the default run level
* If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
* Typically you would set the default run level to either 3 or 5.

**6. Runlevel programs**

* When the Linux system is booting up, you might see various services getting started. For example, it might say “starting sendmail …. OK”. Those are the runlevel programs, executed from the run level directory as defined by your run level.
* Depending on your default init level setting, the system will execute the programs from one of the following directories.
* Run level 0 – /etc/rc.d/rc0.d/
* Run level 1 – /etc/rc.d/rc1.d/
* Run level 2 – /etc/rc.d/rc2.d/
* Run level 3 – /etc/rc.d/rc3.d/
* Run level 4 – /etc/rc.d/rc4.d/
* Run level 5 – /etc/rc.d/rc5.d/
* Run level 6 – /etc/rc.d/rc6.d/
* Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.
* Under the /etc/rc.d/rc\*.d/ directories, you would see programs that start with S and K.
* Programs starts with S are used during startup. S for startup.
* Programs starts with K are used during shutdown. K for kill.
* There are numbers right next to S and K in the program names. Those are the sequence number in which the programs should be started or killed.
* For example, S12syslog is to start the syslog deamon, which has the sequence number of 12. S80sendmail is to start the sendmail daemon, which has the sequence number of 80. So, syslog program will be started before sendmail.

**Difference between RAID and LVM**

**RAID**

* RAID is used for redundancy.
* A RAID device is a physical grouping of disk devices in order to create a logical presentation of one device to an Operating System for redundancy or performance or a combination of the two.
* RAID is a way to create a redundant or striped block device with redundancy using other physical block devices.
* RAID is either a software or a hardware technique to create data storage redundancy across multiple block devices based on required RAID levels.
* RAID is NOT any kind of Data backup solution. Its a solution to prevent one of the SPOFs (Single Point of Failure) i.e. DISK failure. By configuring RAID you are just providing an emergency substitute for the Primary disk. It NEVER means that you have configured DATA backup.

**LVM**

* LVM is a way in which you partition the hard disk logically and it contains its own advantages.
* LVM is a logical layer that that can be anipulated in order to create and, or expand a logical presentation of a disk device to an Operating System.
* LVM usually sits on top of RAID blocks or even standard block devices to accomplish the same result as a partitioning, however it is much more flexible than partitions. You can create multiple volumes crossing multiple physical devices, remove physical devices without loosing data, resize the volumes, create snapshots, etc
* LVM is a software tool to manage large pool of storage devices making them appear as a single manageable pool of storage resource. LVM can be used to manage a large pool of what we call Just-a-bunch-of-Disk (JBOD) presenting them as a single logical volume and thereby create various partitions for software RAID.
* LVM is a disk management approach that allows us to create, extend, reduce, delete or resize the volume groups or logical volumes.