Team Apol

INTROOS S23 - Nieva, Perez, Pol

After power initialized on the system, execution starts at a fixed memory location, registers in the CPU have well-defined values including the instruction pointer (EIP) which holds the memory address for the instruction being executed by the CPU. A hidden base address (an offset) is applied to EIP so that the first instruction executed is at address 0xFFFFFFF0 of the BIOS(Basic Input Output System). This address(called as reset vector) is standard for modern CPUs.

The RAM modules at this point have random values and are all mapped to the flash memory containing the BIOS. The motherboard ensures that the instruction at the reset vector is to be a jump to the memory location mapped to the BIOS entry point. This jump clears the offset present at power up. Now all of these memory locations should contain their right contents because of the memory map kept by the chipset. After these, the CPU starts executing BIOS code, which initializes the hardware, and runs the POST(Power-On Self Test) to check and test the components in the computer. If it fails then it displays error.

After the POST, the BIOS do the Master Boot Record(MBR) which reads the first 512-byte sector of the hard disk and contains two essential components: an OS-specific bootstrapping program at the start of the MBR and followed by a partition table for the disk. The BIOS loads the contents of the MBR into memory location 0x7c00 and jumps to that location to start executing the code inside the MBR. This code in the MBR could be a Windows MBR loader or a code from Linux loaders such as LILO or GRUB. In Microsoft, MBR code takes a look at the partition table, finds the only partition marked as active, loads the boot sector for that partition, and runs that code. Due to the small size of the MBR, the initial stage of any bootloader just contains basic information and leads to another sector, that contains the further code needed by the boot procedure. This is stage 2 of bootloader, in Linux is GRUB2 stage and in Windows is NTLDR or winload.exe. Then this leads to reading of the configuration file, for Linux is grub.conf and for Windows is boot.ini that contains the settings and arguments that are passed to the kernel. The MBR contains the first stage of the boot loader, this code load another sector from the disk which contains additional bootstrap code. The second stage code reads the configuration file, it gives boot options to the user or go with the single-boot system. The BIOS chip tells it to look in a fixed place, usually on the lowest-numbered hard disk for a special program called a boot loader. A boot loader is a small piece of code which locates the kernel, loads it into memory and starts the operating system. This process is called booting, which means starting a computer by loading a kernel.

For windows, after the configuration file, boot.ini, NTDETECT.COM which checks the hardware of the system and passes the related information to NTLDR. Then NTLDR creates a hardware key by using the passed information and that is the reason that after installation of a new device, system asks for restart because change in the hardware causes change in the hardware key and it needs to be updated. After all the task related to the bootloader is done, the CPU jumps to the kernel of the system. The ntoskrnal.exe is the kernel file in a Windows machine that is located in system32 folder. This file initializes kernel data structure, non-boot CPUs, object manager, I/O manager. Kernel is the layer which allows the user space applications and request to interact with the systems hardware. Next is the HAL layer (Hardware Abstraction Layer) that is the hal.dll which give the programs direct access to the hardware components of the system. The last thing to be loaded is the Device Driver, this allows the interactions with the user and the devices. This is responsible for operations performed by devices and are loaded from system32\config\system.

For Linux, the bootloader loads kernel as well as the temporary filesystem before the actual file system is loaded. The image of the kernel is has two parts on the basis of its loading that is real mode and protected mode. The process always occurs in the real mode. The first entry point in the initialization of kernel is head.S (arch/i386/boot/head.S), which is assembly language written file whose task is basic hardware setup. Then there is a jump to head.S (this is present in boot/compressed directory) whose task is set the basic environment and calls decompress\_kernel (present in boot/compresses/misc.c) and then to startup\_32 (present in kernel/head.S). Then it moves to main.c present in arch/x86/boot directory. After all these /sbin/init, /etc/init, /bin/init, and /bin/sh are called in the order; the first user process is init and they read etc/events.d or etc/inittab. Hence the system starts.

References:

<http://www.engineersgarage.com/tutorials/how-computer-pc-boots-up><http://www.tldp.org/HOWTO/Unix-and-Internet-Fundamentals-HOWTO/bootup.html>