CMPSCI 577 Operating Systems Design and Implementation **Spring 2020**

Lecture 5: Feb 4th, 2020

Lecturer: ***Prashant Shenoy*** *Scribe:* ***Serena Chan***

5.1 OS Boot Process

While seemingly mundane and not directly relevant to the modifications we will be making in this course, understanding the OS boot process is important for modifying the kernel. There are six steps that take the machine to a running state from when it is first turned on.

5.1.1 Step 1: BIOS (Basic Input/Output System)

The BIOS is a small piece of code that lives in the firmware and is the first software that runs upon boot. When the computer is turned on, the BIOS runs the Power-on Self Test, where the RAM is initialized and hardware such as disks and peripherals are identified and checked. Once the disk is found, the BIOS will start the boot process from the disk ('bootstrapping').

The BIOS is often stored on EEPROM/ROM (read-only memory); because of its hardware-specific nature, it is not designed to be easily user modifiable. In addition, since the BIOS is the lowest level of software in the PC, it also acts as an interface for the OS to perform I/O and communicate with hardware.

5.1.2 Step 2: MBR (Master Boot Record)

The MDR is the first 512 bytes of memory and consists of three components. In particular, the first 440 bytes contain the bootstrap code that is used to continue the boot process, or the 1st stage boot loader; this is executed by the BIOS. The functionality of the code is to simply search through the partition table and find the root partition, where is where the OS resides. Note that, with multiple partitions, some partitions may not have an OS; however, there must be at least one partition with the OS.

5.1.3 Step 3: Partition Boot Sector & Boot Loader

5.1.3.1 Partition Boot Sector

The Partition Boot Sector is the equivalent of the MBR, but for partitions. Like the MBR, it resides on the first 512 bytes of the partition and contains bootstrap code (2nd stage boot loader) the that the MBR jumps to. Since it is for a specific partition, it also contains OS-specific or file system-specific information.

The bootstrap of the PBS will look up and point execution to the boot loader (or in other cases, the kernel).

5.1.3.2 Boot Loader

The boot loader is commonly referred to as a 2nd-stage bootloader, even though it is technically a 3rd-stage boot loader (after the MBR and PBS). The boot loader loads OS-specific configuration files and provides boot options, which are particularly important if there are multiple kernels available. For instance, the Minix boot loader allows the user to select which kernel and which options they are booting with.

Specifically, the boot loader first needs to understand the structure of the native file system of the OS in order to read its files. Therefore, the the boot loader will first load the minimal drivers required to do so. The boot loader then loads the configuration files, which enable boot options; some boot loaders have additional modules that are also loaded. The system menu is then displayed. After the user selects the preferred option from the system menu (or the default option is invoked), the boot loader will start booting the OS kernel.

Aan example of a boot loader is the Grand Unified Bootloader (GRUB), often used in Linux machines. We will study how the boot process continues in view of GRUB.

5.1.4 Step 4: Kernel

When the kernel boots up, it does not necessarily have access to the disk, so it creates and mounts a small RAM file system (separate from the root file system) that contains files needed for booting. The temporary file system is called `initramfs`, and is a small EXT file system. After the kernel boots and gains access to the disk, the actual file system on disk is able to take over.

While it is possible to use the RAM-based file system (`ramfs`) for other purposes as well, the volatile nature of `ramfs` may make it unsuitable in some situations.

5.1.5 Step 5 and 6: INIT and runlevel

After booting the kernel, the next step is to start all the services that are needed for the machine - this is done through a process called INIT. INIT (sometimes known as `systemd`) is a Linux script that start all sorts of services, such as `dhcp` or `sshd`.

The `inittab` file configures what services to start - one important option is the run level, which dictates different modes of operation. In Linux, different levels have different run scripts, and they are stored in `/etc/rc.d/rc_.d/`, where `_` is replaced with the desired run level.

INIT is the first user-level process - it typically has the smallest process ID, and all user-level processes are children of INIT.

5.2 Booting in MINIX

The boot image contains the microkernel as well as some of the services. The MINIX boot process is very similar to those outlined above, except for the fact that it does not have an `rc.d` directory - everything is in one script.

The `/etc/system.conf` defines the permitted actions for each system services.