Chapter 15. The FreeBSD Booting Process

Table of Contents

15.1. Synopsis

15.2. FreeBSD Boot Process

15.3. Device Hints

15.4. Shutdown Sequence

15.1. Synopsis

The process of starting a computer and loading the operating system is referred to as "the bootstrap process", or "booting". FreeBSD's boot process provides a great deal of flexibility in customizing what happens when the system starts, including the ability to select from different operating systems installed on the same computer, different versions of the same operating system, or a different installed kernel.

This chapter details the configuration options that can be set. It demonstrates how to customize the FreeBSD boot process, including everything that happens until the FreeBSD kernel has started, probed for devices, and started init(8). This occurs when the text color of the boot messages changes from bright white to grey.

After reading this chapter, you will recognize:

- The components of the FreeBSD bootstrap system and how they interact.
- The options that can be passed to the components in the FreeBSD bootstrap in order to control the boot process.
- The basics of setting device hints.

- How to boot into single- and multi-user mode and how to properly shut down a FreeBSD system.

> **Note**
>
> This chapter only describes the boot process for FreeBSD running on x86 and amd64 systems.

15.2. FreeBSD Boot Process

Turning on a computer and starting the operating system poses an interesting dilemma. By definition, the computer does not know how to do anything until the operating system is started. This includes running programs from the disk. If the computer can not run a program from the disk without the operating system, and the operating system programs are on the disk, how is the operating system started?

This problem parallels one in the book The Adventures of Baron Munchausen. A character had fallen part way down a manhole, and pulled himself out by grabbing his bootstraps and lifting. In the early days of computing, the term *bootstrap* was applied to the mechanism used to load the operating system. It has since become shortened to "booting".

On x86 hardware, the Basic Input/Output System (BIOS) is responsible for loading the operating system. The BIOS looks on the hard disk for the Master Boot Record (MBR), which must be located in a specific place on the disk. The BIOS has enough knowledge to load and run the MBR, and assumes that the MBR can then carry out the rest of the tasks involved in loading the operating system, possibly with the help of the BIOS.

> **Note**
>
> FreeBSD provides for booting from both the older MBR standard, and the newer GUID Partition Table (GPT). GPT partitioning is often found on computers with the Unified Extensible Firmware Interface (UEFI). However, FreeBSD can boot from GPT partitions even on machines with only a legacy BIOS with gptboot(8). Work is under way to provide direct UEFI booting.

The code within the MBR is typically referred to as a *boot manager*, especially when it interacts with the user. The boot manager usually has more code in the first track of the

disk or within the file system. Examples of boot managers include the standard FreeBSD boot manager boot0, also called Boot Easy, and GNU GRUB, which is used by many Linux® distributions.

> **Note**
> Users of GRUB should refer to GNU-provided documentation.

If only one operating system is installed, the MBR searches for the first bootable (active) slice on the disk, and then runs the code on that slice to load the remainder of the operating system. When multiple operating systems are present, a different boot manager can be installed to display a list of operating systems so the user can select one to boot.

The remainder of the FreeBSD bootstrap system is divided into three stages. The first stage knows just enough to get the computer into a specific state and run the second stage. The second stage can do a little bit more, before running the third stage. The third stage finishes the task of loading the operating system. The work is split into three stages because the MBR puts limits on the size of the programs that can be run at stages one and two. Chaining the tasks together allows FreeBSD to provide a more flexible loader.

The kernel is then started and begins to probe for devices and initialize them for use. Once the kernel boot process is finished, the kernel passes control to the user process init(8), which makes sure the disks are in a usable state, starts the user-level resource configuration which mounts file systems, sets up network cards to communicate on the network, and starts the processes which have been configured to run at startup.

This section describes these stages in more detail and demonstrates how to interact with the FreeBSD boot process.

15.2.1. The Boot Manager

The boot manager code in the MBR is sometimes referred to as *stage zero* of the boot process. By default, FreeBSD uses the boot0 boot manager.

The MBR installed by the FreeBSD installer is based on **/boot/boot0**. The size and capability of boot0 is restricted to 446 bytes due to the slice table and `0x55AA` identifier at the end of the MBR. If boot0 and multiple operating systems are installed, a