How does booting work on FreeBSD,MAC and Gentoo Linux

Wouter Borremans

september 14, 2004

Contents

This report describes the findings of the bootprocess in three diffrent operating systems;

- FreeBSD
- Gentoo Linux
- Macintosh

1 Preface

When a user turns his computer on, the system does not know by definition what it has to do until the operating system is loaded. A mechanism had to be designed to actually build a bridge from the system itself to the operating system. This is called *bootstrapping* .

On Intel (x86) machines, another mechanism takes care of the booting. The BIOS (Basic Input Output System) has enough knowledge to load and run the MBR (Master Boot Record) and assumes that the MBR has enough data to carry out the rest of the tasks that involved in loading an operating system. If you only have one operating system installed on your disks then the standard MBR will suffice. This MBR searches for the first bootable slice on the disk, and then runs the code on that slice to load the remainder of the operating system. If you have installed multiple operating systems on your disks then you can install a different MBR, one that can display a list of different operating systems, and allows you to choose the one to boot from.

There are many bootloaders on the market at the moment, depending on your architecture you can choose several intellligent bootmanagers that are able to boot multiple partitions. A few examples of bootmanagers are:

- LiLo (Linux Loader)
- Yaboot (Mac)
- BootMagic (x86 architectures)

2 FreeBSD

The FreeBSD boot process is devided into three diffrent stages. The first stage is handled by the MBR, the MBR is able to get the system into the second stage. This stage will load the kernel into the memory and lets it execute. The kernel is started and begins to probe for devices and tries to initialize them for use. Once the kernel boot process is finished, the init(8) is executed. This init will mount the filesystem and starts up the networking etc. Since the practical assignment asks us to describe what happens after the kernel loads, i will now continue in "Stage 3" .

Stage 3 loads the bootloader which is usually located in the */boot/loader*. The loader is intended as a user-friendly method for configuration, using an easy-to-use built-in command set, backed up by a more powerful interpreter, with a more complex command set.

Loader Program Flow During the initialization, the loader will probe for devices and consoles. It will also figure out from which disk it is booting. It will set variables accordingly, and an interpreter is started where user commands can be passed from a script or interactively. The loader will then read the */boot/loader.rc*,this will load the kerbel selected modules (depending on */boot/defaults/loader.conf* and */boot/loader.conf*) Finally, by default, the loader issues a 10 second wait for key presses, and boots the kernel if it is not interrupted. If interrupted, the user is presented with a prompt which understands the easy-to-use command set, where the user may adjust variables, unload all modules, load modules, and then finally boot or reboot. The user can choose for a runlevel that he/she wants to load.

After this, things will be executed in user level (super user)

- sys/kern/init main.c will be run
- /sbin/init will take over, and /etc/default/rc.conf script will be run
- After this, the inlog screen appears

3 Gentoo

Once the system boots up, it tries to search for bootdevices (as mentioned earlier in this document). After this, the bootloader will be loaded. In our practise case the bootloader that we use is *Yaboot*, this is the most used bootloader with Mac. Now, the kernel will be unpacked and the CPU can start executing the kernel. After the kernel is loaded, it will initialize kernel specific structures and tasks and will eventually the init process.

init The init process is now started and starts to mount the filesystems which are defined in */etc/fstab*. After this, the */etc/init.d* will be started

and will execute several scripts. In specific order, the following steps are executed:

- The filesystems are mounted from /etc/inittab. This is done with the command: si::sysinit:/sbin/rc sysinit
- All scripts that have symlinks to /etc/runlevels/boot are executed.
- This is done with the command: rc::bootwait:/sbin/rcboot
- Init will how check which runlevel should be executed, it reads this from /etc/inittab. This file contains a line like: id:3:initdefault indicating a runlevel of 3.
- Init will now check what it should do with runlevel 3. Runlevel 3 is defined as:. l3:3:wait:/sbin/rc default
- Init will now starts the services with the default arguments
- Finally, when all scripts are executed, init activates the terminals
- The terminals are activiated with lines like: c1:12345:respawn:/sbin/agetty

4 Macintosh

The boot sequence of MacOS X consists of a few steps which are executed after each other. This section describes the steps in order they are executed.

4.1 BootROM

When de MAC is turned on, the BootROM firmware is will be executed. In general, the BootROM itself has two main functions; initialize the present hardware and selecting an operating system to boot.

The BootROM has to components to help it carry out it's functions:

- *Performing a Power-On Self Test*
  This initializes some hardware interfaces an verifies that sufficient memory is available and in a good state.
- *Open Firmware*
  The Open Firmware initalizes the rest of the hardware and builds a tree with useful device information. The MacOS kernel will read that information on starting up and use that information to continue booting.

4.2 BootX

When the BootROM selects the MacOS X as the operating system, the BootX booter is started. BootX loads the kernel environment. BootX will eventually shows the boot image on the screen. The BootX program is located at */System/Library/CoreServices*.

While loading the kernel environment, BootX first attempts to load a previously cached set of device drivers (mkext cache) for hardware that is involved in the boot process. When the cache is missing or isn't appropriate, BootX will search for for drivers and other extensions which are important for the boot process. (The extensions are selected depending on the *OSBundleRequired* property.)

Once the kernel and all necessary drivers which are needed for booting are loaded, the BootX program will start the kernel initialization. The kernel will now try to find the root device. From this point Open Firmware is not accessible anymore.

The kernel initializes the Mach and BSD data structures and initializes the I/O Kit. The I/O Kit links the loaded drivers into the kernel. It uses the device tree to determine which drivers to link. Once the kernel finds the root device, it roots BSD off of it. Finally, the kernel starts the mach init process.

After the root filesystem is mounted, system initialization proceeds to run the system startup items and launch the present system daemons.

4.3 System initialization

The *init* process is now executed . The init process is the father of all processes and has process ID (PID) 1. It owns every other process on the system. In principle the init process has four tasks:

- Determining which user mode the user wants (Booting from CDROM or single-user mode)
- Running system-initialization shell scripts (*/etc/rc.boot* and */etc/rc*) which completes basic initialization tasks.
- *Init* launches the getty command. Init launches the login window application.
- As the parent of all processes, *init* handles all necessary cleanup of the system processes as they terminate.

As mentioned, the *init* process is the parent of all the processes. Though, there are two kinds of processes:

- *System processes* are processes that are started or initialized before the loginwindow application is executed.

- *User processes* are started afer the loginwindow application. A user process is always associated with a particular user session.

Note that user processes are killed when the loginwindow application logs a user out. System processes itself are only killed when the system shuts down or reboots.

4.4 rc.boot and rc scripts

The rc.boot and rc scripts in */etc* perform basic initialization tasks in a specific order.

First the rc.boot script performs a file system consistency check and synchronizes memory with the file system. After that, the rc script performs the following actions:

- It starts the device-driver loader (kextd).
- It loads the mach bootstrap-based services
- It runs the *update* background process, which flushes the file-system cache.
- It creates the swap file for the virtual memory system and starts the dynamic pager.
- It launches the *fix-prebinding* daemon, which fixes out of date prebinding information for applications on an as-needed basis.
- It Finally, the rc script starts the SystemStarter program to process the local and system startup items.

Since MacOS X is a fact, a new boot mechanism is used; the *register-mach-bootstrap-servers* to load the deamons for the current context. There are two diffrent kinds of contexts; the startup context and the user context. One benefit of using the *register-mach-bootstrap-servers* is that it registers each daemon as a bootstrap device. This provides a tremendous preformance advantage.

5 Conclusion

This document has mentioned several issues related to the startup procedures of FreeBSD, Gentoo and MacOS X. Although the base of the systems is the same, there are some slight diffrences in the startup process;

- Gentoo uses symlinks from /etc/rc?.d to /etc/init.d/ to start all services
- FreeBSD uses /etc/default/rc.conf to start all services
- Mac uses the /etc/rc script to start all services

6 References

References

[1] http://www.faqs.org/docs/linux scratch/chapter07/usage.html

[2] http://www.freebsd.org/doc/en US.ISO8859-1/books/handbook/boot.html

[3] http://infocom.cqu.edu.au/Units/aut99/85321/Resources/Print Resources/Textbook/chap12/

[4] http://www.kevinboone.com/boot.html

[5] http://openskills.info/seminars/distro/5.htm

[6] http://developer.apple.com/documentation/MacOSX/Conceptual/BPSystemStartup/Concepts