Linux Boot Sequence

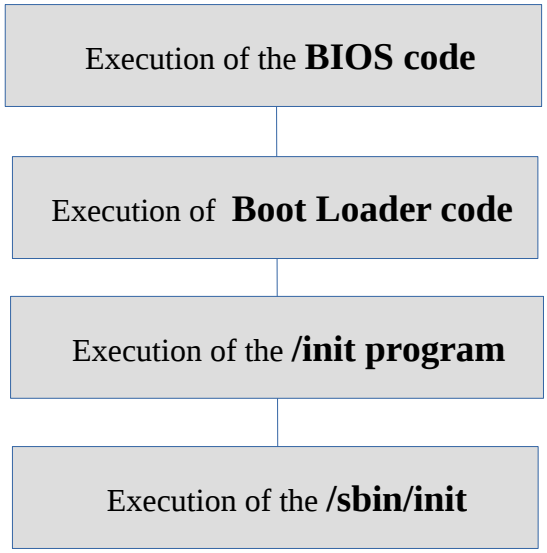
Ravi Shankar.G

Booting in Linux

- A **Linux boot** process is the initialization of the **Linux** open source operating system on a computer.
- Also known as the **Linux startup** process, a **Linux boot** process covers a number of steps from the initial bootstrap to the launch of the initial user-space application.

Debian and general Linux boot process

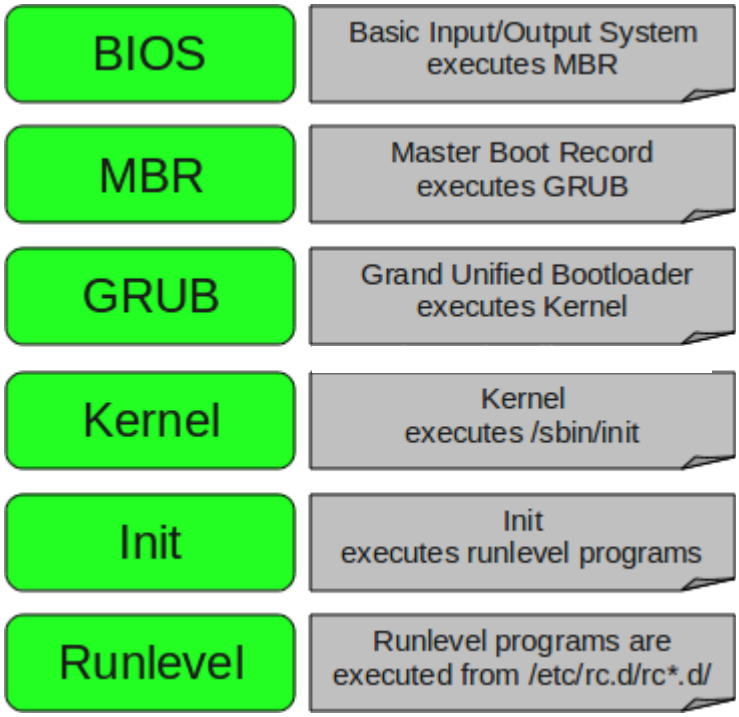
- **System Initialization**
  - The computer system undergoes several phases of initialization from the power-on event until it offers the fully functional operating system (OS) to the user.
  - Here is a rough sequence of events for the default installation of Debian with the Linux kernel on the typical PC platform.



- ✔ **/init program** : under the Linux kernel with the expanded **initramfs image** in memory as the temporary root file system.
- ✔ **/sbin/init** : under the Linux kernel while **switching the root file system** to the hard disk.

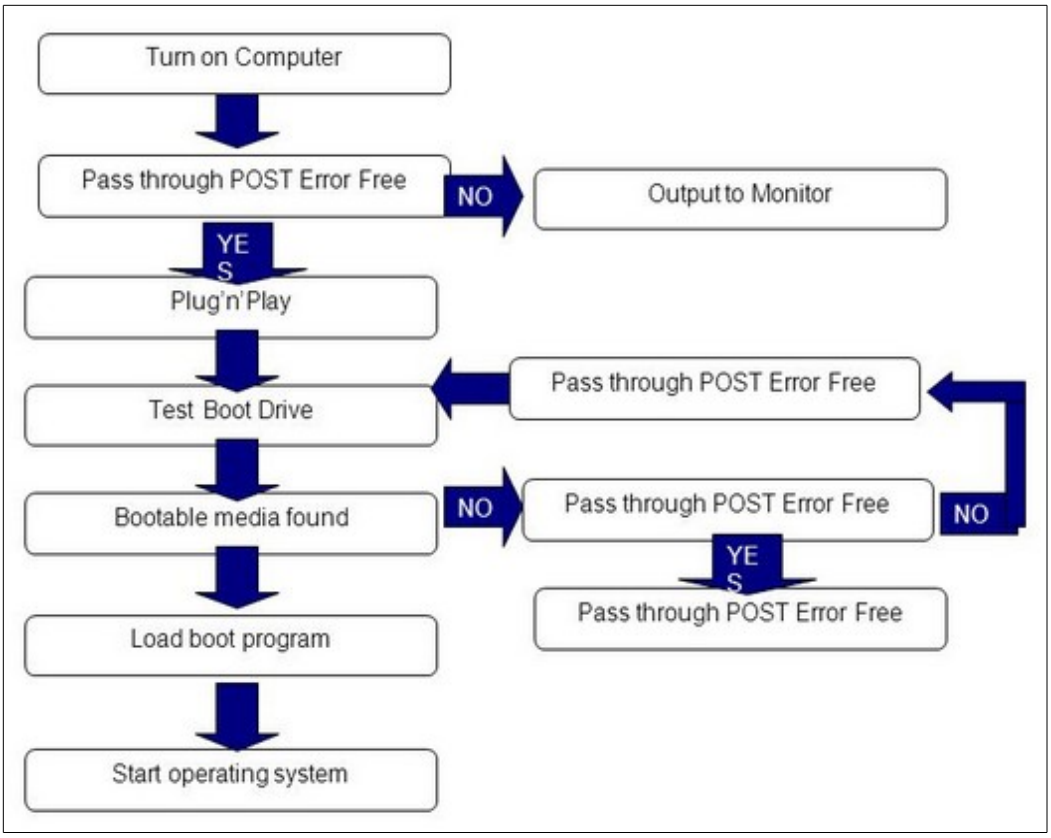
- Of course, these can be configured differently. For example, if we compiled our own kernel, we may be skipping the step with the ***initramfs image***.
- For simplicity, we'll limit discussion to the typical i386 PC platform

6 Stages of Linux Boot Process (Startup Sequence)



I. BIOS

- Program used by the processor to get the system started, once the system is turned on.
- The main function of the BIOS is to perform some **system integrity checks**.
- Searches, loads, and executes the boot loader program.
- It is a type of **firmware** used by the processor during the booting process.



- When the power-on event happens, the computer hardware initializes itself and executes BIOS code **residing on the ROM** from the particular memory location.
- Basically this BIOS code performs the basic initialization of the hardware (**POST : power on self test**) and hands the system control to the next step which we provide.
- Typically, the ***first few sectors of the first found selected device*** (hard disk, floppy disk, CD-ROM, ...) are loaded to the memory and this initial code is executed.
- This initial code can be:
  - ✔ the kernel code of the **target OS** if it fits in this small space
  - ✔ the kernel code of the **stepping stone OS** such as FreeDOS
  - ✔ the **boot loader code**
- Typically, the system is booted from the specified partition of the **primary hard disk partition.**
- The first **2** sectors of the hard disk contain the **Master Boot Record (MBR)**.
- The ***disk partition information*** including the boot selection is recorded at the end of this MBR.
- The first boot loader code executed from the BIOS for the hard disk occupies the rest of this MBR.

II. Boot Loader

- The default install of the Debian system **places first stage GRUB boot loader code into the MBR** for the PC platform.
- There are many boot loaders and configuration options available.
  1. GRUB
  2. Lilo
  3. Syslinux
  4. Isolinux
  5. Loadlin
  6. Neil Turton's MBR
- Here initrd is used as a generic term from the boot loader perspective indicating both **traditional initrd RAM disk image and new initramfs RAM disk image**.
- For GRUB, the menu configuration file is located at **/boot/grub/menu.lst.**

III. Master Boot Record

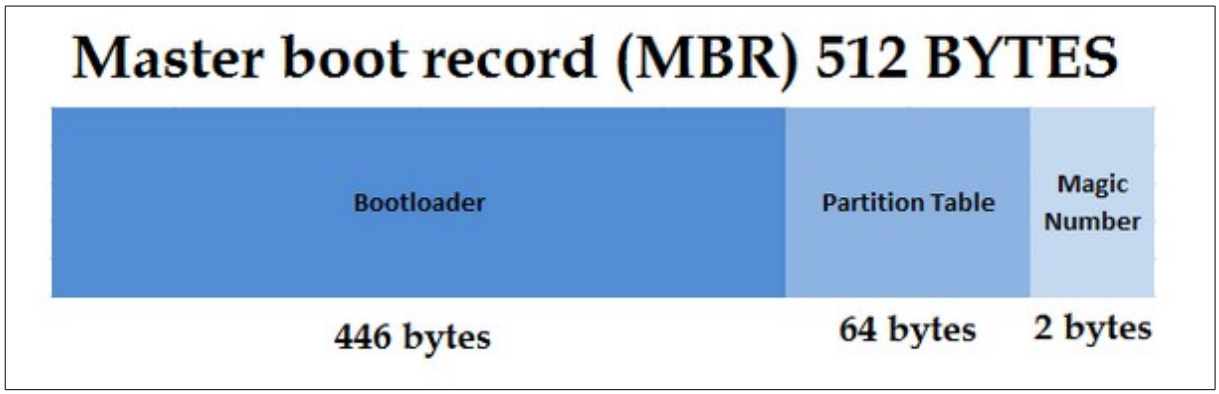
- MBR is a special type of ***boot sector at the very beginning of partitioned*** computer mass storage devices like fixed disks or removable drives.
- It is located in the **first 2 sector of the bootable disk**.
- Typically **/dev/hda, or /dev/sda**
- MBR is less than **512 bytes** in size.
- This has **four** components

    1) Primary boot loader info in 1st **440 bytes**

    2) Error Message section **6 bytes.**

    3) Partition table info in next **64 bytes**

    4) MBR validation check in last **2 bytes (Magic Number).**

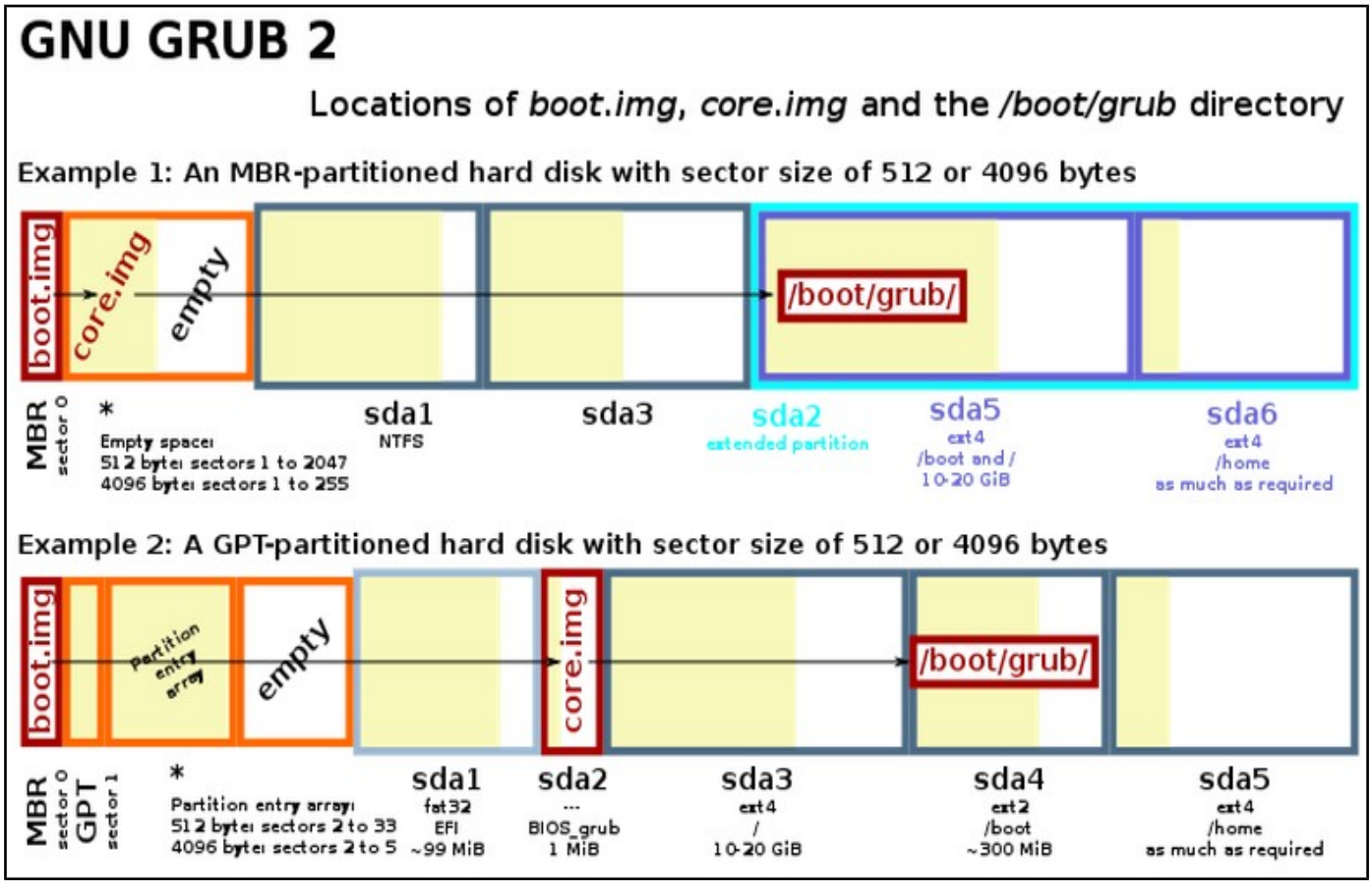


- It contains information about **GRUB** (or LILO in old systems).
- So, in simple terms MBR **loads and executes the GRUB boot loader**.

IV. GRUB

- GRUB stands for **Grand Unified Bootloader**.
- If we have **multiple kernel images** installed on your system, we can choose which one to be executed.
- GRUB displays a **splash screen**, waits for few seconds, if we 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).

- So, in simple terms GRUB just loads and executes Kernel and initrd images.



V. Kernel

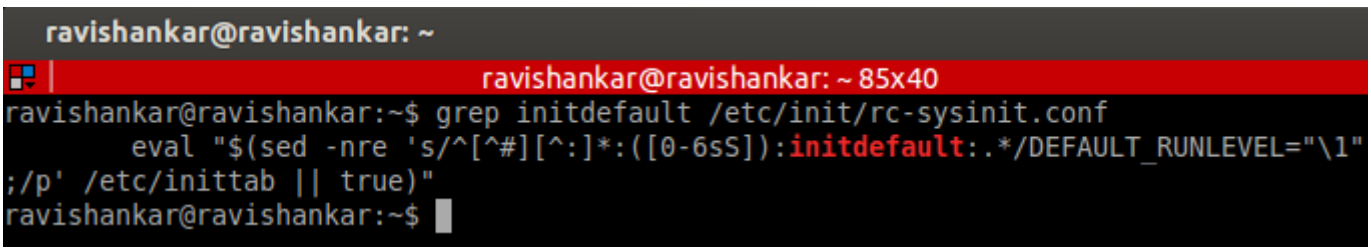
- Mounts the **root file system** as specified in the **"root="** in grub.conf
- Kernel executes the **/sbin/init** program
- Since **init** was the 1st program to be executed by Linux Kernel, it has the process id (PID) of **1**.
- Do a **'ps -ef | grep init'** and check the pid.

```
ravishankar@ravishankar: ~
ravishankar@ravishankar: ~ 80x24
ravishankar@ravishankar:~$ ps -ef | grep init
root         1     0  0 11:40 ?        00:00:01 /sbin/init
ravisha+  1531  1152  0 11:41 ?        00:00:00 init --user
ravisha+ 13467 13454  0 16:21 pts/0    00:00:00 grep --color=auto init
ravishankar@ravishankar:~$
```

- 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.

VI. /init on the initramfs RAM disk image

- The current default Debian system uses the 2.6 series Linux kernel with the initramfs RAM disk image which is a gzipped cpio archive when booted initially.
- The **/init** program is executed in this initramfs as **a shell script program** which initializes the kernel in the user space.
- And it hands control over to **/sbin/init** on the hard disk while switching to the root file system.
- This is a preparatory part of boot process.
- It's an optional process and offers flexibility to the boot process such as adding kernel modules before the main boot process or mounting the root file system as an encrypted one.
- The commands used in this environment is a stripped down Unix system provided by the busybox etc.
- We can interrupt this part of the boot process to gain root shell by providing break=init etc. to the kernel boot parameter.
- Following are the available run levels
    - N– System bootup (NONE).
    - 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 the system to identify the default run level



- Typically we would set the default run level to either 3 or 5.

VII. /sbin/init on the hard disk (Init Scripts)

- This is the main part of the boot process.
- The runlevel at the start of this process is **"N"** (none).
- The /sbin/init program initializes the system following the description in the /etc/inittab configuration file.
- Debian normally uses the traditional sysvinit scheme with the sysv-rcpackage.
- See man 8 init, man 5 inittab, and /usr/share/doc/sysv-rc/README.runlevels.gz for the exact explanation.
- Following is a simplified overview of this boot process.

  1. The Debian system goes into runlevel S to initialize the system under the single-user mode to complete hardware initialization etc.

  2. The Debian system switches itself to one of the specified multi-user runlevels (2 to 5) to start the system services.
- The initial runlevel used for multi-user mode is specified with the "init=" kernel boot parameter or in the "initdefault" line of /etc/inittab.
- The Debian system as installed starts at the runlevel 2.
- All scripts executed by the init system are located in the directory /etc/init.d/.
- There are four other directories which correspond to the multi-user runlevels:

  /etc/rc2.d

  /etc/rc3.d

  /etc/rc4.d

  /etc/rc5.d
- These directories contain relative links to the scripts in /etc/inint.d/.

VIII. Runlevel programs

- When the Linux system is booting up, we 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 the run level.
- Depending on our 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/

- Under /etc/rc.d/rc*.d/ directories, we can 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.