Boot & Bootloader

Based on LPIC book

Chapter 1: Boot

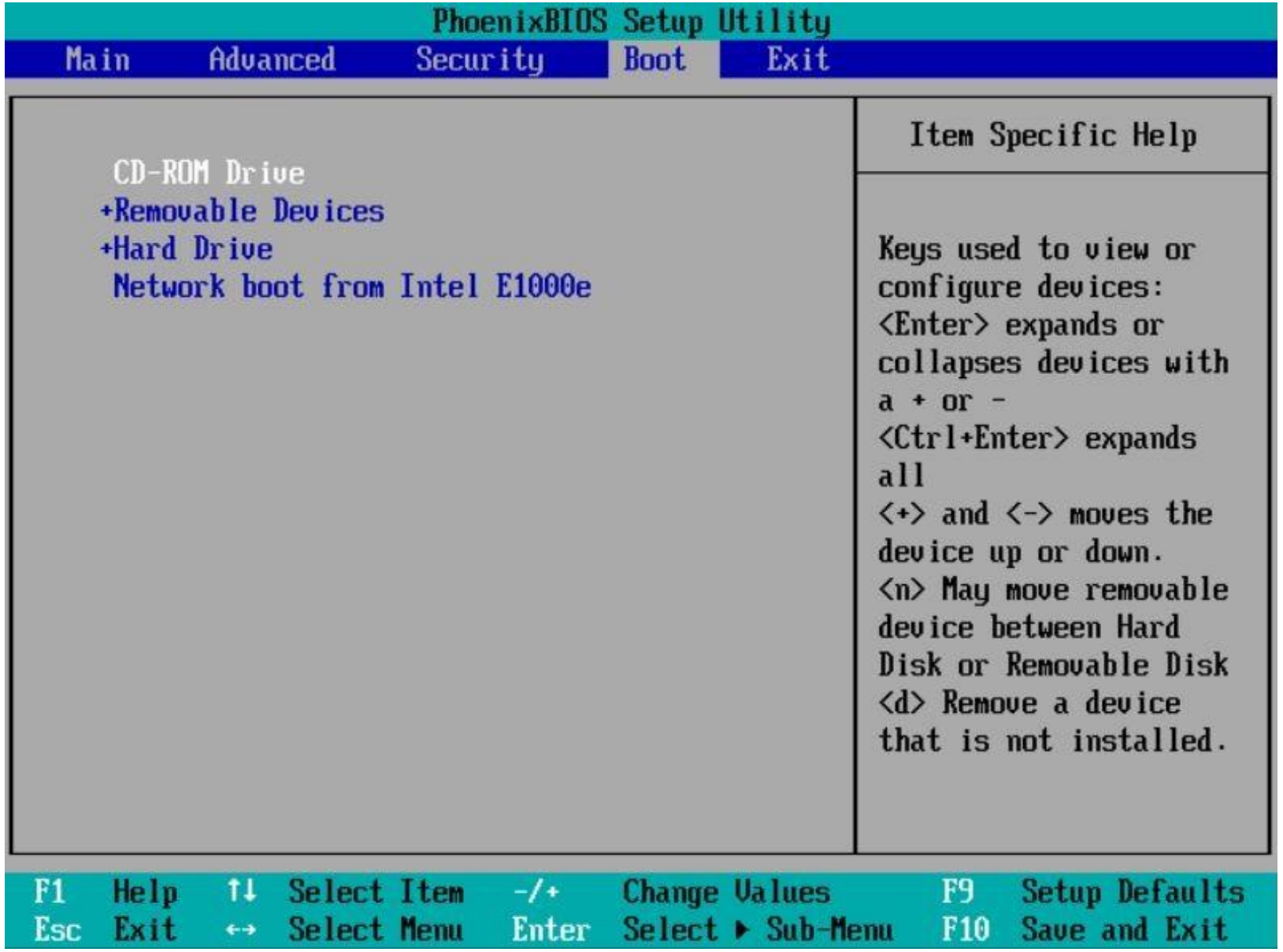
- The Boot Process
- BIOS
- UEFI
- Bootloader
- Kernel
  - dmesg
- init

The Boot Process

Boot process steps:

1. Motherboard Firmware does a PowerOnSelfTest
2. Motherboard loads the bootloader
3. Bootloader loads the Linux Kernel-based on its configs/commands
4. The Kernel loads and prepares the system (root filesystem) and runs the initialization program
5. Init program start the service, other programs, … (web server, graphical interface, networking, etc.)

The Firmware on the motherboard can be BIOS or UEFI.


PhoenixBIOS Setup Utility
Main
Advanced
Security
Boot
Exit
CD-ROM Drive
+Removable Devices
+Hard Drive
Network boot from Intel E1000e
Item Specific Help
Keys used to view or configure devices:
<Enter> expands or collapses devices with a + or -
<Ctrl+Enter> expands all
<+> and <-> moves the device up or down.
<n> May move removable device between Hard Disk or Removable Disk
<d> Remove a device that is not installed.
F1 Help ↑↓ Select Item -/+ Change Values F9 Setup Defaults
Esc Exit ↔ Select Menu Enter Select ▸ Sub-Menu F10 Save and Exit

BIOS

Basic **I**nput **O**utput **S**ystem

- Older
- Limited to one sector of the disk and needs a multi-stage bootloader
- Can start the bootloader from internal/external HDD, CD/DVD, USB Flash drive, Network server
- If booting from the HDD, the Master Boot Record will be used (1 sector)

UEFI

Unified **E**xtensible **F**irmware **I**nterface.

- Modern and fancy
- Specifies a special disk partition for the bootloader. Called EFI System Partition (ESP)
- ESP is FAT and mounted on `/boot/efi` and bootloader files has .efi extensions

You can check `/sys/firmware/efi` to see if you are using a UEFI system or not

Bootloader

Bootloader initializes the minimum hardware needed to boot the system and then finds and runs the OS.

Technically you can point your UEFI to run anything you want but typically under GNU/Linux systems, we use GRUB. Even the GRUB can be used to run any specific program you need but generally it runs the OS.

Kernel

The Kernel is the core of your operating system, the LINUX itself. Your bootloader loads the kernel in the memory and runs it. But kernel needs some initial info to start; Things like drivers are necessary to work with the hardware. Those are stored in `initrd` or `initramfs` alongside the kernel and used during the boot.

You can also send parameters to the kernel during the boot using the Grub configs. For example, sending a 1 or S will result the system booting in single-user mode (recovery). Or you can force your graphics to work in 1024×768x24 mode by passing `vga=792` to the Kernel during the boot.

dmesg

Linux will show you the boot process logs during the boot. Some desktop systems hide this behind a fancy boot splash which you can hide using the Esc key or press Ctrl+Alt+F1.

Fun Fact: During the bootup, only The Kernel is running so it should record and keep its logs!

dmesg command will show the full data from **kernel ring buffer** up to now.

But `cat /var/log/dmesg` will show only the data during the boot.

init

When the Kernel finished its initialization, its time to start other programs. To do so, the Kernel runs the Initialization Daemon process, and it takes care of starting other daemons, services, subsystems and programs. Using the init system one can say "I need service A and then service B. Then I need C and D and E but do not start D unless the A and B are running". The system admin can use the init system to stop and start the services later.

init

There are different init systems:

- **SysVinit** is based on Unix System V. Not being used much but people loved it and you may see it on older machines or even on recently installed ones
- **systemd** is the new replacement. Some people hate it but it is being used by all the major distros. Can start services in parallel and do lots of fancy stuff!
- **upstart** was an event-based replacement for the traditional init daemon. The project was started in 2014 by Canonical (the company behind Ubuntu) to replace the SysV but did not continue after 2015 and Ubuntu is now using the systemd as its init system.

init

The init process had the ID of 1 and you can find it by running the

```
# which init
/sbin/init
# readlink -f /sbin/init
/usr/lib/systemd/systemd
# ps -p 1
PID TTY TIME     CMD
1   ?   00:00:06 systemd
```

You can check the hierarchy of processes using the pstree command.

```
pstree
```

Chapter 2: Bootloader

- Boot overview
- GRUB
- GRUB Legacy
- GRUB2
  - GRUB2 commands
  - Interacting with GRUB2
- Kernel boot parameters

Boot overview

Most systems use BIOS or UEFI. When on BIOS, the system will do a self test called POST (Power-On Self-Test). Then it will hand over the boot process to the first sector of Master Boot Record (MBR) which is track (Cylinder) 0, side (Head) 0 and Sector 1 of the first disk.

MBR is only 512 bytes so we need a *smart bootloader* to handle larger boot managers and even multiple systems. Some of these boot loaders are LILO, GRUB and GRUB2.

Boot overview

If the system is using UEFI, the hardware will follow the UEFI stages. They start with a security phase and will continue till the end phase where the UEFI looks for an EFI System Partition, which is just a FAT32 partition (Usually the first one, but that's implementation-defined) with PE executables and runs them.

In both cases, the binary starts the boot loader. It might be a complete bootloader on `/boot/efi/` of your computer or a small loader for the main grub on the MBR or a windows loader or even a chainloader.

Chain Loading is when a boot loader, loads another boot loader. This is done when a Linux bootloader needs to start a Windows system.


GNU GRUB version 2.04
*Ubuntu
Advanced options for Ubuntu
Memory test (memtest86+)
Memory test (memtest86+, serial console 115200)
Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, `e' to edit the commands
before booting or `c' for a command-line.
The highlighted entry will be executed automatically in 27s.

GRUB

GRUB (**GR**and **U**nified **B**ootloader) started to replace the older LILO. The first version (1) is called Grub Legacy and started in 1999. The 2nd version started in 2005 and is a complete rewrite of version 1.

It's a menu-based system where you can choose which Kernel or chainloader to boot. It is also possible to edit the menus on the fly or give direct commands from a command line.

GRUB Legacy

Usually the GRUB v1 (actually 0.9) is installed in `/boot/grub`. Its main configuration is in `/boot/grub/menu.lst` but nowadays some distros (including RedHat Based ones) link this to the `/boot/grub/grub.conf`.

A sample `menu.lst` / `grub.conf` file for GRUB legacy consists of two sections. The first section contains global configs and the 2nd part defines different kernel/initram or chainloader options.

GRUB Legacy

The global configs are:

Config	Description
#	Comment
color	Foreground and background colors for normal and active items
default	Which boot menu item is the default
fallback	Which boot menu should be used if the *default* fails
hiddenmenu	Hide the menu options
splashimage	Show this image in the background!
timeout	Wait this much and then start the default
password	Security is important! Will ask this password
savedefault	Remember the last booted item

GRUB Legacy

On the second part of the config, we have these:

Config	Description
title	Defines the section name
root	Disk and partition where `/boot` directory is. In the form of (hddrive, partition), say (hd0, 0) or (hd0, msdos0)
kernel	Kernel image file name in `/boot`
initrd	Initramfs file in `/boot`
rootnoverify	Defines a non-Linux root partition
chainloader	Another file will act as stage 1 loader. Used for booting Windows systems

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You do not have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /, eg.
#          root (hd0,5)
#          kernel /boot/vmlinuz-version ro root=/dev/sda6
#          initrd /boot/initrd-version.img
#boot=/dev/sda6
default=1
timeout=10
splashimage=(hd0,5)/boot/grub/splash.xpm.gz
#hiddenmenu
password --md5 $1$RW1VW/$4XGAklxB7/GJk0uO47Srx1
title Upgrade to Fedora 11 (Leonidas)
    kernel /boot/upgrade/vmlinuz preupgrade \
      repo=hd::/var/cache/yum/preupgrade stage2=\
      hd:UUID=8b4c62e7-2022-4288-8995-5eda92cd149b:/boot/upgrade/install.img \
      ks=hd:UUID=8b4c62e7-2022-4288-8995-5eda92cd149b:/boot/upgrade/ks.cfg
    initrd /boot/upgrade/initrd.img
title Fedora (2.6.26.8-57.fc8)
    root (hd0,5)
    kernel /boot/vmlinuz-2.6.26.8-57.fc8 ro root=LABEL=FEDORA8 rhgb quiet
    initrd /boot/initrd-2.6.26.8-57.fc8.img
title Fedora (2.6.26.6-49.fc8)
    root (hd0,5)
    kernel /boot/vmlinuz-2.6.26.6-49.fc8 ro root=LABEL=FEDORA8 rhgb quiet
    initrd /boot/initrd-2.6.26.6-49.fc8.img
title GRUB Menu
    rootnoverify (hd0,1)
    chainloader +1
title Windows
    rootnoverify (hd0,0)
    chainloader +1
```

A sample of GRUB-Legacy config

GRUB2

This is the most common boot loader these days. On BIOS systems it is installed on `/boot/grub/` or `/boot/grub2/` and under UEFI it goes in `/boot/efi/EFI/distro-name/` (say `/boot/efi/EFI/fedora/`). GRUB2's configuration file is called `grub.cfg`.

GRUB2

Here is a simplified `grub.cfg`:

```
set default="0"
menuentry "Fedora" {
  set root=(hd0,1)
  linux /boot/vmlinuz-5.10.0-9-arm64 ro quiet
  initrd /boot/initrd.img-5.10.0-9-arm64
}
menuentry "Windows" {
  chainloader (hd1,msdos2)+1
}
```

As you can see, GRUB uses Linux-style numbering for partitions, so the first partition on the first hard disk is `(hd0,1)` or `(hd0,msdos1)` for DOS partitions or `(hd0,gpt1)` for GPT drives.

GRUB2

Here you can see some of the options:

Option	Description
menuentry	Defines a new menuentry
set root	Defines the root where `/boot` located
linux, linux16	Defines the location of the Linux kernel on BIOS systems
linuxefi	Defines the Linux kernel on UEFI systems
initrd	Defines the initramfs image for BIOS systems
initrdefi	Defines the initramfs image for UEFI systems

GRUB2 commands

The installation is done with `grub-install /dev/sda` and after changing the config files, you need to issue `grub2-mkconfig` or `grub-mkconfig`. It reads the configuration files from `/etc/grub.d/` and `/etc/default/grub/` and create the `grub.cfg` file based on them. You run it like this:

```
grub2-mkconfig > /boot/grub2/grub.cfg
```

or

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

GRUB2 commands

There is also a command called `update-grub` as a frontend to `grub-mkconfig`

which runs `grub-mkconfig -o /boot/grub/grub.cfg`

Please note that on some modern distros, you have both `grub` and `grub2` commands available for compatibility reasons, and one links to the other.

Interacting with GRUB2

If you press `c` on the grub menu, you will go into the *GRUB Command Line* or *GRUB shell*. There you can type commands like `root` and `kernel` and `initrd` and boot the system with `boot` or press the `Esc` key to return back to the menu.

Kernel boot parameters

In the above configs, we sent some parameters to the kernel like this:

```
    linux   /boot/vmlinuz-5.10.0-9-arm64 root=/dev/sda1 ro  quiet
```

This tells the kernel to boot in *ReadOnly* mode and does not show lots of logs during the boot (*quiet*).

Kernel boot parameters

Option	Description
console=	Set the console
debug	Start in debug mode
init=	Run an specific program instead of the default init
initrd=	Use this initrd
ro	Mount the root filesystem read only
rw	Mount the root filesystem for read and write
root=	Use this as the root filesystem
selinux	Disable `selinux` on boot
single,S,1,Single	Boot in single user mode for troubleshooting (SysV)
systemd.unit=	Boot in this systemd target

Explore more…

- GRUB Manual:

  https://www.gnu.org/software/grub/manual/grub/grub.html

- GRUB Documentation:

  https://www.gnu.org/software/grub/grub-documentation.html

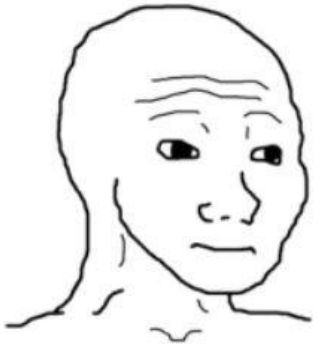
- GRUB Source Code:

  https://github.com/coreos/grub/

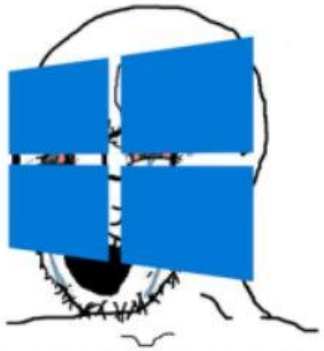
- Linux Source Code:

  https://github.com/torvalds/linux

The End

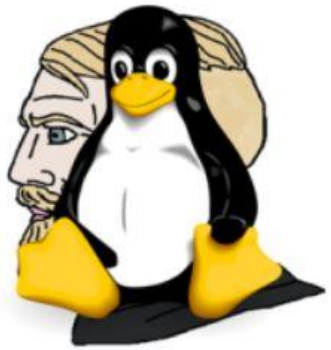


hey can i uninstall edge



NOOO!!! YOUR SYSTEM WILL BREAK

im going to uninstall the bootloader



go ahead lol