GRUB

GRUB is the first thing that runs when you start up your computer, especially on Linux systems. It loads itself into memory and shows you a menu where you can choose which operating system or kernel you want to boot. After you make a choice, GRUB loads the kernel into memory and hands over control so the system can fully start. It also has the ability to load other bootloaders or operating systems if needed, making it really flexible.

UEFI and BIOS

UEFI and BIOS are both firmware systems that help start your computer. BIOS is the older of the two, with a simpler design, and it initializes hardware and loads the operating system. UEFI, the modern replacement, offers faster boot times, better security features, and supports larger hard drives. UEFI also has a more user-friendly interface and can handle more complex tasks than BIOS, making it a more flexible choice for newer systems.

Load and execute kernel in OS

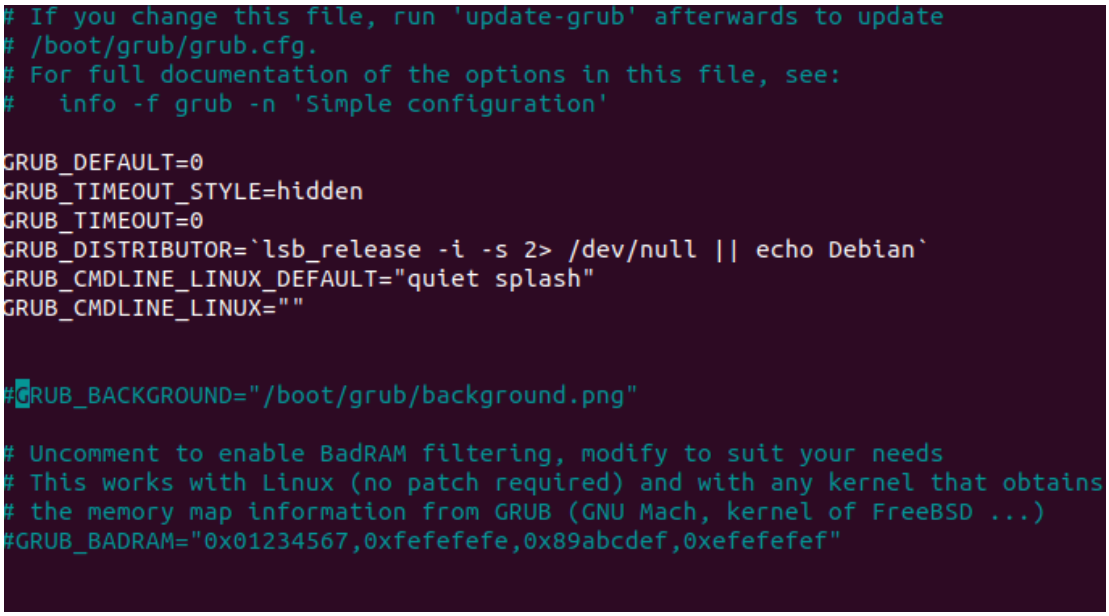
To load and execute a kernel in an operating system, the bootloader first loads the kernel into memory from the boot device. The bootloader then hands over control to the kernel, which begins initializing hardware and setting up system resources. Once the kernel has everything ready, it starts running system processes and the rest of the operating system takes over, allowing the computer to become fully functional. Essentially, the kernel is the heart of the OS, and the bootloader is the bridge to get it running.

MBR

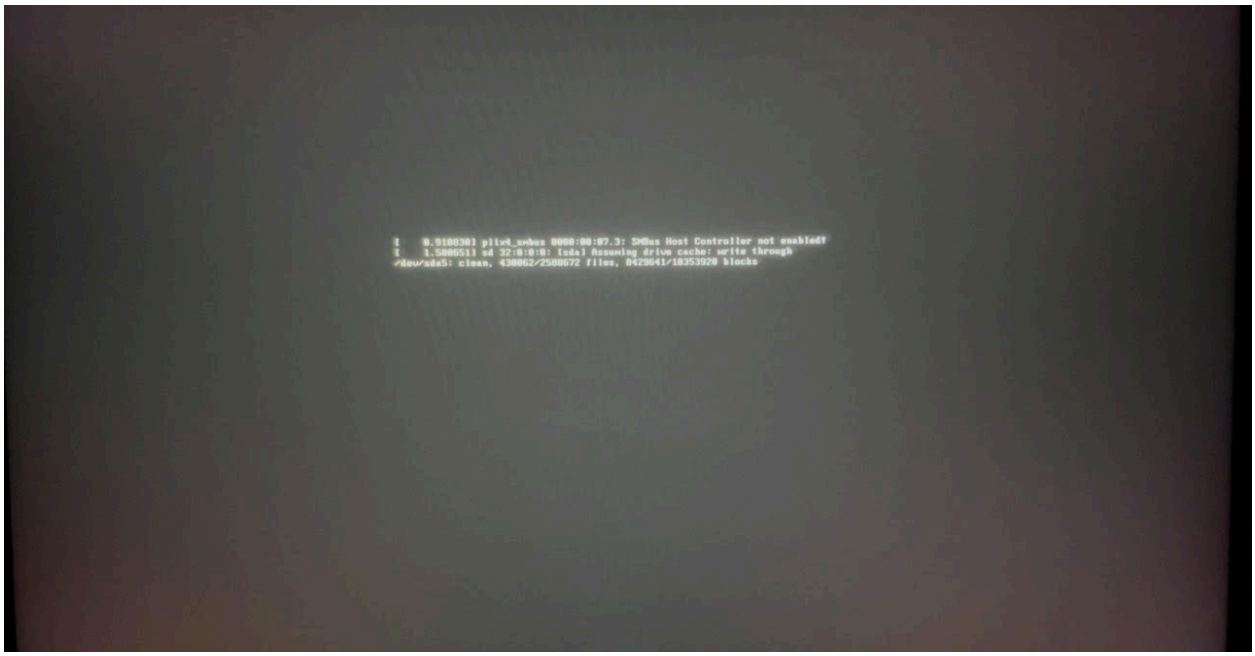
The **partition table** in the MBR is like a map that tells the computer how the disk is divided into up to four primary partitions. The **boot signature (0x55AA)** is a tiny "stamp" at the end of the MBR that confirms it's valid and bootable—without it, the system won't start. **WinHex** is a handy tool that lets experts peek into these raw disk structures, edit them, or recover lost data, acting like a digital microscope for hard drives. Together, they ensure your system knows where to find everything when it boots up.

GRUB configuration

Initially, the */etc/default/grub* file has the following configuration:



It shows no menu and the OS starts immediately:



Then, using the following command, I edited the */etc/default/grub* file:

```
erfan@erfan-virtual-machine:~/Desktop$ sudo nano /etc/default/grub
```

```
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=menu
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash init=/usr/local/bin/boot-script.sh"
GRUB_CMDLINE_LINUX=""


#GRUB_BACKGROUND="/boot/grub/background.png"

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
```

GRUB_TIMEOUT_STYLE=menu displays the operating system selection menu before booting.
The *GRUB_TIMEOUT=5* parameter sets a 5-second delay before automatic booting.
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash init=/usr/local/bin/boot-script.sh" replaces the normal startup process with my custom script to show on the screen before the boot process is complete:

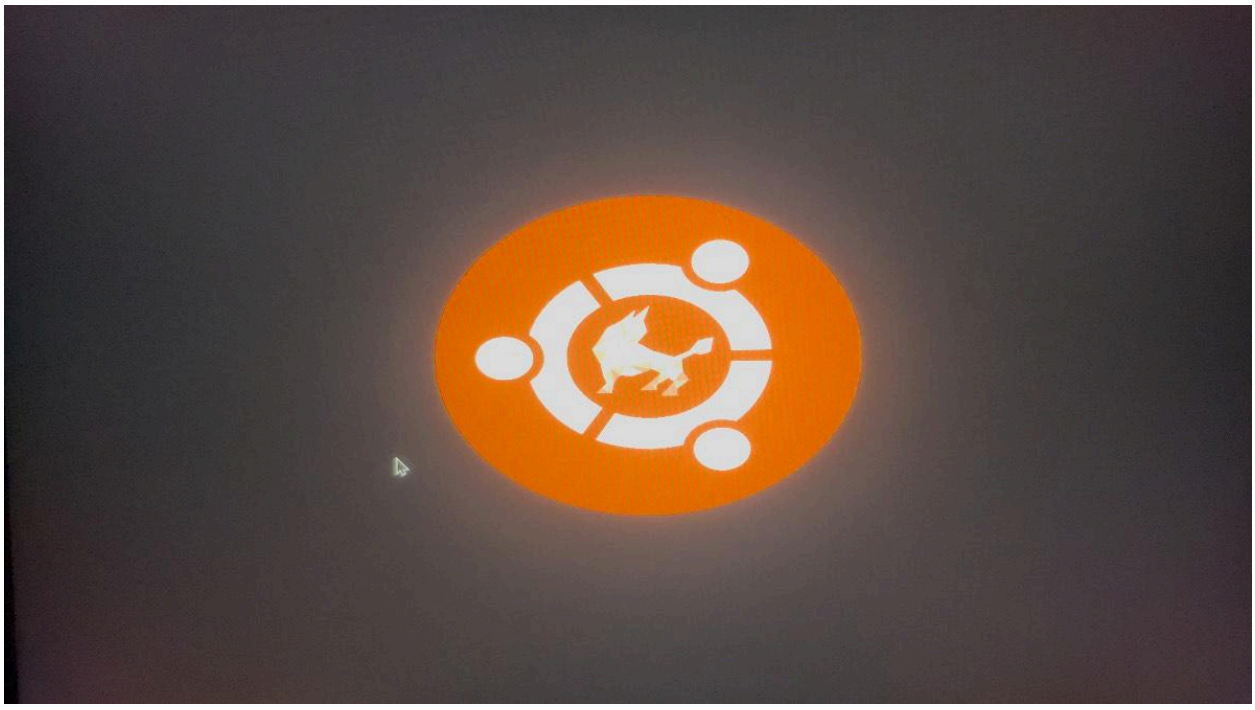
```
#!/bin/bash

# Display ASCII art
echo -e "\n\n"
echo -e "          \033[34m/\\ \033[0m"
echo -e "         \033[34m/  \\ \033[0m"
echo -e "        \033[34m/    \\ \033[0m"
echo -e "       \033[34m/______\\ \033[0m"
echo -e "      \033[32m|      | \033[0m"
echo -e "      \033[32m|  ☀   | \033[0m"
echo -e "      \033[32m|      | \033[0m"
echo -e "      \033[32m|______| \033[0m"
echo -e "     \033[33m/        \\ \033[0m"
echo -e "    \033[33m/          \\ \033[0m"
echo -e "   \033[33m/            \\ \033[0m"
echo -e "\033[31m--------------------------------\033[0m"
echo -e "Custom boot script executed at $(date)"
echo -e "\033[31m--------------------------------\033[0m"

# Hand over to normal init process
exec /sbin/init
```

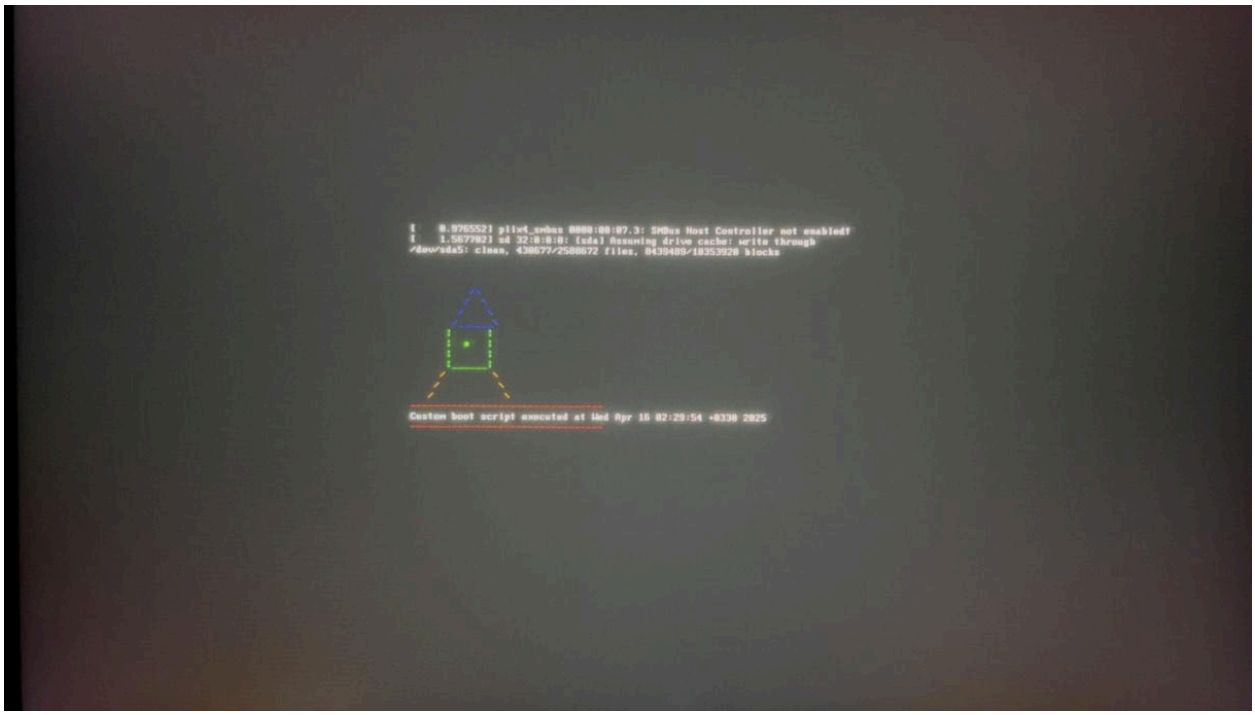
GRUB_BACKGROUND="/boot/grub/background.png" shows a picture as background during the boot process.

After making changes to */etc/default/grub*, I ran this command to apply them:

```
erfan@erfan-virtual-machine:~/Desktop$ sudo update-grub
Sourcing file `/etc/default/grub'
Generating grub configuration file ...
Found background: /boot/grub/background.png
Overriding foreground/background text colors (/etc/grub.d/06_local_colors)
Found linux image: /boot/vmlinuz-5.15.0-136-generic
Found initrd image: /boot/initrd.img-5.15.0-136-generic
Found linux image: /boot/vmlinuz-5.15.0-134-generic
Found initrd image: /boot/initrd.img-5.15.0-134-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
```

Now the boot process is like this:




[0.976552] piix4_smbus 0000:00:07.3: SMBus Host Controller not enabled!
[1.567782] sd 32:0:0:0: [sda] Assuming drive cache: write through
/dev/sda5: clean, 438679/2560672 files, 8439489/10353920 blocks
Custom boot script executed at Wed Apr 16 02:29:54 +0330 2025

Before making the changes, this is the result of running these following commands:

```
erfan@erfan-virtual-machine:~/Desktop$ systemd-analyze
Startup finished in 1.629s (kernel) + 34.450s (userspace) = 36.080s
graphical.target reached after 32.334s in userspace
erfan@erfan-virtual-machine:~/Desktop$ systemd-analyze blame
28.053s elasticsearch.service
 4.695s plymouth-quit-wait.service
 2.698s postfix@-.service
 2.564s snapd.seeded.service
 2.400s snapd.service
 2.097s NetworkManager-wait-online.service
 1.453s dev-sda5.device
  843ms dev-loop10.device
  842ms dev-loop12.device
  834ms dev-loop15.device
  828ms dev-loop14.device
  810ms dev-loop8.device
  810ms dev-loop13.device
  796ms dev-loop20.device
  792ms dev-loop19.device
  792ms dev-loop9.device
  787ms networkd-dispatcher.service
  774ms dev-loop18.device
  771ms dev-loop11.device
  771ms fwupd.service
  770ms dev-loop17.device
  767ms dev-loop21.device
  763ms dev-loop16.device
  712ms dev-loop2.device
  706ms dev-loop0.device
  703ms dev-loop3.device
  695ms dev-loop7.device
  685ms dev-loop5.device
  677ms dev-loop1.device
```

And this is the results after the changes:

```
erfan@erfan-virtual-machine:~/Desktop$ systemd-analyze
Startup finished in 1.692s (kernel) + 29.115s (userspace) = 30.808s
graphical.target reached after 29.096s in userspace
erfan@erfan-virtual-machine:~/Desktop$ systemd-analyze blame
24.530s elasticsearch.service
 5.305s plymouth-quit-wait.service
 3.140s postfix@-.service
 2.797s snapd.seeded.service
 2.651s snapd.service
 2.073s NetworkManager-wait-online.service
 1.739s dev-sda5.device
  909ms fwupd.service
  907ms dev-loop2.device
  891ms dev-loop1.device
  881ms dev-loop5.device
  870ms dev-loop4.device
  860ms dev-loop7.device
  842ms dev-loop0.device
  836ms dev-loop6.device
  828ms dev-loop3.device
  766ms networkd-dispatcher.service
  390ms systemd-journal-flush.service
  283ms accounts-daemon.service
  283ms udisks2.service
  261ms apport.service
  204ms dev-loop20.device
  203ms upower.service
  192ms dev-loop14.device
  189ms dev-loop21.device
  188ms dev-loop19.device
  186ms dev-loop8.device
  182ms systemd-udev-trigger.service
  182ms systemd-logind.service
  180ms snapd.apparmor.service
  178ms avahi-daemon.service
  174ms dev-loop18.device
  173ms bluetooth.service
  171ms dev-loop12.device
  171ms dev-loop11.device
  170ms dev-loop16.device
  166ms dev-loop9.device
  164ms systemd-journald.service
  163ms dev-loop13.device
  162ms NetworkManager.service
```