# 

[Типы загрузки](#_t81dfge40dlp)

[BIOS/MBR](#_zfjh2oa2s3fw)

[UEFI/GPT](#_5brjqb9aceo3)

[Пример миграции от MBR к EFI](#_xugf8jgh73y3)

[Boot Loader Specification](#_mg4jm3i24op2)

[Файлы которые используются при загрузки](#_hswjklatvauy)

[Загрузчик GRUB2](#_10tlq73akoqo)

[Конфигурация загрузчика GRUB2](#_6k6nzjemnr4k)

[Поменяем картинку при загрузке](#_cthwj4jpv4md)

[Утилита для редактирования конфигурации загрузки](#_50jo7u5282co)

[Initial ramdisk](#_z7u3r85zzc5b)

[Сбрасываем пароль root](#_shtebjpns1m1)

[Защита паролем menu GRUB](#_67aiyqgkwgm)

[Меняем графическую заставку](#_f622cfkmj8h6)

[Добавляем модуль в initramfs](#_967w5biew4ik)

[Загрузка без загрузчика EFISTUB Kernel](#_odme4e9w1eg3)

[Документация](#_mydxuygwpbbw)

# Типы загрузки

## BIOS/MBR

the program is written into read-only, permanent memory and is always available for use

Применяется для совместимости со старым оборудованием. Встречается в системах виртуализации.

* the processor looks at the end of system memory for the BIOS program, and runs it.
* the BIOS tests the system, looks for and checks peripherals, and then locates a valid device with which to boot the system.
* the order of the drives searched while booting is controlled with a setting in the BIOS
* BIOS loads into memory whatever program is residing in the first sector of this device, called the Master Boot Record (MBR). The MBR is only 512 bytes in size and contains machine code instructions for booting the machine, called a boot loader.
* BIOS finds and loads the boot loader program into memory, it yields control of the boot process to it.
* **first-stage boot loader** is a small machine code binary on the MBR. Its job is to locate the **second stage boot loader** (GRUB) and load the first part of it into memory

https://datacadamia.com/data\_storage/mbr

initramfs-4.18.0-193.14.2.el8\_2.x86\_64.img

## UEFI/GPT

EFI - Extensible Firmware Interface

UEFI

контролирует загрузочный процесс

умеет монтировать разделы и читать файловые системы

ESP (EFI system partition)

системный раздел на котором содержатся приложения скомпилированные под архитектуру EFI

это могут быть загрузчики, утилиты или ядро ОС

ESP раздел EFI firmware может быть как GUID так и MBR. Разделы должны содержать идентификатор.

EFI system partition C12A7328-F81F-11D2-BA4B-00A0C93EC93B (уточнить???)

MBR partition-table scheme is [0xEF](https://en.wikipedia.org/wiki/Partition_type#PID_EFh)

|  | BIOS | UEFI |
| --- | --- | --- |
| System switched on, | the [power-on self-test (POST)](https://en.wikipedia.org/wiki/Power-on_self-test) is executed. | |
| After POST, | BIOS initializes the hardware required for booting (disk, keyboard controllers etc.). | UEFI initializes the hardware required for booting (disk, keyboard controllers etc.). |
| BIOS | BIOS launches the first 440 bytes ([the Master Boot Record bootstrap code area](https://wiki.archlinux.org/title/Partitioning#Master_Boot_Record_(bootstrap_code))) of the first disk in the BIOS disk order. | Firmware reads the **boot entries in the NVRAM** to determine which EFI application to launch and from where.  Firmware looks for an [EFI system partition](https://wiki.archlinux.org/title/EFI_system_partition) on that disk and tries to find an EFI application in the fallback boot path \EFI\BOOT\BOOTx64.EFI (BOOTIA32.EFI on [systems with a IA32 (32-bit) UEFI](https://wiki.archlinux.org/title/Unified_Extensible_Firmware_Interface#UEFI_firmware_bitness)). This is how UEFI bootable removable media work. |
| boot loader | The boot loader's first stage in the MBR boot code then launches its second stage code (if any) from either:   * next disk sectors after the MBR, i.e. the so called post-MBR gap (only on a MBR partition table). * a partition's or a partitionless disk's [volume boot record (VBR)](https://en.wikipedia.org/wiki/Volume_boot_record). * the [BIOS boot partition](https://wiki.archlinux.org/title/BIOS_boot_partition) ([GRUB](https://wiki.archlinux.org/title/GRUB) on BIOS/GPT only). | Firmware launches **the EFI application.**   * This could be a [boot loader](https://wiki.archlinux.org/title/Arch_boot_process#Boot_loader) or the  [kernel](https://wiki.archlinux.org/title/Kernel) itself using [EFISTUB](https://wiki.archlinux.org/title/EFISTUB). * It could be some other EFI application such as the  [UEFI shell](https://wiki.archlinux.org/title/UEFI_shell) or a [boot manager](https://wiki.archlinux.org/title/Arch_boot_process#Boot_loader) like [systemd-boot](https://wiki.archlinux.org/title/Systemd-boot) or [rEFInd](https://wiki.archlinux.org/title/REFInd). |
| boot loader | The actual [boot loader](https://wiki.archlinux.org/title/Arch_boot_process#Boot_loader) is launched. |  |
| boot loader | The boot loader then loads an operating system by either **chain-loading** or directly loading the **operating system kernel.** |  |
|  |  |  |
|  |  |  |

Например:

parted /dev/sda -l

file /boot/efi/EFI/efistub/bootx64.efi

## 

Виртуальные машины поставляются с загрузкой MBR.

Причины? совместимости с гипервизорами, по умолчанию используют MBR

Попробуем сделать современную ОС миграцией.

Как сделать UEFI инсталляцию сразу или через Packer?

Процесс миграции.

находим диск в системе у нас это /dev/sdb

создаем на диске раздел ESP

## Пример миграции от MBR к EFI

1 **Создаем раздел с типом ESP**

lsblk

fdisk -l /dev/sdb

Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors

Units: sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: gpt

Disk identifier: CB31299F-0910-C84B-984B-20D2C3D36E9F

Device Start End Sectors Size Type

/dev/sdb1 2048 1026047 1024000 **500M EFI System**

partprobe

для автоматизации  
yum install -y gdisk

sgdisk -n 1: -t 1:ef00 /dev/sdb

2. **Форматируем в FAT32**

yum -y install dosfstools

mkfs.fat -F32 /dev/sdb1

Добавить /boot в /etc/fstab

blkid /dev/sdb1

echo $UUID /boot vfat defaults 0 2

/etc/fstab

UUID=21B7-C118 /boot vfat defaults 0 2

UUID=$(blkid /dev/sdb1 | cut -f 2 -d ' ' | tr -d '"')

echo $UUID /boot vfat defaults 0 2

ls /boot # check boot files

mount /boot

ls /boot # empty directory

??? сначала директорию смонтировать или сначала пакеты?

3. **Установить grub на ESP раздел**

yum install

yum -y install efi-filesystem grub2-efi-x64-modules efibootmgr grub2-efi-x64.x86\_64

umount /boot/

yum install grub2-efi-x64.x86\_64

mount /boot

yum reinstall grub2-efi-x64.x86\_64

rpm -ql grub2-efi-x64.x86\_64

ls -l /etc/grub2-efi.cfg

grub2-mkconfig -o /etc/grub2.cfg

grub2-mkconfig -o /etc/grub2-efi.cfg

4. **Установить ядро в /boot**

yum -y update kernel-core

или yum -y reinstall kernel-core

В данный момент инструкции содержат ошибки

grub2-install --target=x86\_64-efi --efi-directory=/boot/ --bootloader-id=grub2

Как workaround переустановим пакеты

sudo dnf reinstall shim-\* grub2-\* # <https://bugzilla.redhat.com/show_bug.cgi?id=1917213>

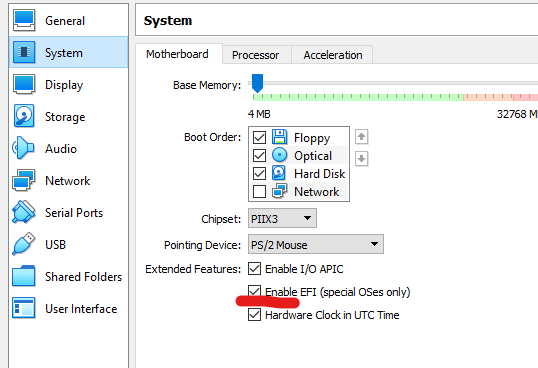
shutdown -h now

офишибки

<https://bugzilla.redhat.com/show_bug.cgi?id=1497918>

<https://bugs.launchpad.net/diskimage-builder/+bug/1893029>

# Поменять тип загрузки на EFI



4. Создадим конфиг

grub2-mkconfig -o /etc/grub2.cfg

5. **Изучаем команды в UEFI shell**

help

help map

map -r

map -u

FS0:

ls

cat

cd

bcfg

Добавить загрузочную запись из EFI shell

bcfg boot add 0 fs1:\EFI\grub\grubx64.efi grub

Вывести список загрузочных записей (-v verbose, -b paged)

bcfg boot dump -v -b

6. **Добавить загрузочную запись из Linux**

efibootmgr -c -d /dev/sdb1 --label "grubmenu" --loader '\EFI\grub2\grubx64.efi'

efibootmgr -c -d /dev/sdb1 --label "grub" --loader '\efi\centos\grubx64.efi'

<https://www.redhat.com/sysadmin/bios-uefi>

<https://www.golinuxcloud.com/update-grub2-grubby-grub2-editenv-rhel-8/>

7. шаги для загрузки ядра вручную

предварительно нужно загрузить blscfg   
 insmod blscfg

загрузить grub вручную.

выполнить команды

set prefix=... - путь где лежит grubenv файл

load\_env

insmod blscfg

blscfg

boot with empty rootfs

<https://mirrors.edge.kernel.org/pub/linux/utils/boot/dracut/dracut.html#_troubleshooting>

можно загрузится из dracut emergency shell если /dev/root будет указывать на корневую файловую систему

1. make a symbolic link to the unlocked root volume  
    # ln -s /dev/mapper/luks-$UUID /dev/root
2. With the root volume available, you may continue booting the system by exiting the dracut shell  
    # exit

# Boot Loader Specification

описывает схему которая позволяет разным операционным системам (дистрибутивам) управлять конфигурационной директорией загрузчика. Включая записи в загрузочном меню. Предлагается использовать единый формат конфигурационного файла для всех реализаций загрузчика, операционных систем, утилит пользователя.

Содержимое директории $BOOT

* $BOOT/loader/ is the directory containing all files needed for Type #1 entries
* $BOOT/loader/entries/ is the directory containing the drop-in snippets. This directory contains one .conf file for each boot menu item.

find /boot/loader/

/boot/loader/

/boot/loader/entries

/boot/loader/entries/ac4fce7ddccf4acb8ed1057791830358-4.18.0-193.6.3.el8\_2.x86\_64.conf

Имя файла уникальное. Предлагается использовать файл /etc/machine-id

(/etc/machine-id or the D-Bus machine ID for OSes that lack /etc/machine-id), the kernel version (as returned by uname -r) and an OS identifier (The ID field of /etc/os-release).

cat /etc/machine-id

852614d49f89479eba3849aece013419

Файл содержит поля title version linux initrd options

title CentOS Linux (4.18.0-193.6.3.el8\_2.x86\_64) 8 (Core)

version 4.18.0-193.6.3.el8\_2.x86\_64

linux /boot/vmlinuz-4.18.0-193.6.3.el8\_2.x86\_64

initrd /boot/initramfs-4.18.0-193.6.3.el8\_2.x86\_64.img $tuned\_initrd

options

kernel-install add $(uname -r) /lib/modules/$(uname -r)/vmlinuz

man 8 kernel-install

https://access.redhat.com/solutions/3766391

# Файлы которые используются при загрузки

1. Модули grub   
    ls /boot/grub2/i386-pc/\*.mod
2. Модули ядра find /lib/modules/ или так найти путь для модуля   
   modinfo libata 2>&1 | grep filename
3. Модули для initrd (initramfs )   
   dracut --list-modules либо здесь   
   ls /usr/lib/dracut/modules.d

# Загрузчик GRUB2

## Конфигурация загрузчика GRUB2

Настройки содержатся здесь:

/etc/default/grub

/etc/grub.d/40\_custom

Команда обновления конфига

/usr/sbin/grub2-mkconfig

Сгенерированный конфиг

Имя конфига зависит от типа загрузки

EFI

grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg

или BIOS

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

Пример: отключаем ipv6 (CentOS7)

vim /etc/default/grub

# cat /etc/default/grub

GRUB\_TERMINAL\_OUTPUT="console"

GRUB\_CMDLINE\_LINUX=" rhgb quiet ipv6.disable=1"

GRUB\_DISABLE\_RECOVERY="true"

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

cat /boot/grub2/grub.cfg

Пример: добавляем пользовательское меню

/etc/grub.d/

10\_linux - поиск ядра на разделе по умолчанию

40\_custom - пользовательское меню

41\_custom - пользовательское меню из внешнего файла

grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg

<https://wiki.centos.org/HowTos/Grub2>

40\_custom

menuentry ‘OTUS Kernel’ {

set root=’(hd1,msdos1)’

linux /otus\_kernel root=/dev/sdb1 ro quiet

initrd /initrd\_otus\_kernel.img

}

40\_custom

menuentry "May be run by any user" --unrestricted {

set root=(hd0,1)

linux /vmlinuz

}

## Поменяем картинку при загрузке

vagrant ssh-config > config

scp -F config ~/Pictures/2017-12-13\ 19.35.32.jpg otus\_efi:./image.jpg

1. Установим пакет grub2-efi-x64-modules

2. Создаем директорию

mkdir /boot/efi/EFI/centos/x86\_64-efi/

3. Проверим что модуль gfxterm\_background.mod установлен

find /boot/ | grep gfxterm

/boot/grub2/x86\_64-efi/gfxterm\_background.mod

4. Добавим загрузку модуля в конфигурационный файл

cat /etc/default/grub

GRUB\_TIMEOUT=4

#GRUB\_TERMINAL\_OUTPUT="console"

GRUB\_BACKGROUND="/boot/grub2/image.jpg"

GRUB\_PRELOAD\_MODULES="gfxterm\_background"

5. Rebuild grub2 config  
grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg

6. Проверить пути и модули "insmod gfxterm\_background" (можно проверить и в генераторах /etc/grub.d/00\_header)

## Утилита для редактирования конфигурации загрузки

grubby --help

grubby --default-kernel

grubby --default-title

grubby --default-index

grubby --info=ALL

grubby --remove-args="rhgb quiet" --update-kernel /boot/vmlinuz-4.18.0-147.5.1.el8\_1.x86\_64

grubby --info /boot/vmlinuz-4.18.0-147.5.1.el8\_1.x86\_64

grubby --update-kernel=/boot/vmlinuz-4.18.0-147.5.1.el8\_1.x86\_64 --args="rhgb quiet"

grubby --copy-default --add-kernel=/boot/vmlinuz-4.18.0-147.5.1.el8\_1.x86\_64 --title="Otus Menu Kernel" --initrd="/boot/initramfs-4.18.0-147.5.1.el8\_1.x86\_64"

ls /boot/loader/entries/

Дополнительная информация по загрузчикам

<https://systemd.io/BOOT_LOADER_SPECIFICATION/>

Boot loader specification

# Initial ramdisk

## Сбрасываем пароль root

* Дописываем в конец строки rd.break (опционально убираем quiet)
* Загружаемся нажимаем Ctrl - x
* Смотрим список смонтированных файловых систем. Ищем /sysroot  
  mount
* Перемонтируем в режиме rw /sysroot.   
  mount -o remount,rw /sysroot
* Проверяем командой mount что /sysroot примонтирован в rw режиме (чтение-запись)  
  mount
* Заходим в образ системы   
  chroot /sysroot
* Изменяем пароль пользователя командой   
  passwd
* Т.к. разрешен selinux - создаем файл .autorelabel. (по умолчанию разрешен) Выполняем команду touch /.autorelabel
* Команда exit выходим из chroot.
* Перемонтируем в ro (read only)   
  mount -o remount,ro /sysroot

## Защита паролем menu GRUB

grub2-setpassword # пароль в открытом виде поэтому используем

grub2-mkpasswd-pbkdf2

cat /etc/grub.d/01\_users

cat /boot/grub2/user.cfg

Пункт меню доступный любому пользователю

40\_custom.cfg

menuentry "May be run by any user" --unrestricted {

set root=(hd0,1)

linux /vmlinuz

}

https://www.thegeekdiary.com/centos-rhel-7-how-to-password-protect-grub2-menu-entries/

## Меняем графическую заставку

yum install plymouth\\*

plymouth-set-default-theme spinfinity

plymouth-set-default-theme --list

dracut -f

## Добавляем модуль в initramfs

mkdir /usr/lib/dracut/modules.d/01test

cd /usr/lib/dracut/modules.d/01test

cat >module-setup.sh<<EOF

#------------------------------------------------

#!/bin/bash

check() {

return 0

}

depends() {

return 0

}

install() {

inst\_hook cleanup 00 "${moddir}/test.sh"

}

#------------------------------------------------

EOF

chmod +x module-setup.sh

vi test.sh

test.sh

#------------------------------------------------

#!/bin/bash

exec 0<>/dev/console 1<>/dev/console 2<>/dev/console

cat <<'msgend'

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

< I'm dracut module >

-------------------

\

\

.--.

|o\_o |

|:\_/ |

// \ \

(| | )

/'\\_ \_/`\

\\_\_\_)=(\_\_\_/

msgend

sleep 10

echo " continuing....

#------------------------------------------------

chmod +x test.sh

mkinitrd -f -v /boot/initramfs-$(uname -r).img $(uname -r)

lsinitrd -m /boot/initramfs-$(uname -r).img | grep test

note: remove rhgb quiet from configuration

After reboot look through:

less -R /var/log/boot.log

# Загрузка без загрузчика EFISTUB Kernel

Проверяем что ядро собрано с EFISTUB

zgrep CONFIG\_EFI\_STUB /boot/config-$(uname -r)

CONFIG\_EFI\_STUB=y

(минуя загрузчик)

EFIPATH=/boot/efi/EFI/efistub

mkdir $EFIPATH

cp /boot/vmlinuz-$(uname -r) $EFIPATH/bootx64.efi

cp /boot/initramfs-$(uname -r).img $EFIPATH/initramfs.img

Смотрим параметры загрузки

cat /proc/cmdline

Добавляем новую загрузочную запись

efibootmgr -c -d /dev/sdb1 --label "Linux" --loader '\efi\EFI\efistub\bootx64.efi' -u "root=UUID=eeefc623-b66d-4e42-ad2e-2802985dbb32 ro resume=UUID=eb47d436-f867-4113-a4bd-be84b0996b31 initrd=\\efi\\EFI\\efistub\\initramfs.img"

efibootmgr -c -d /dev/sda1 --label "Linux" --loader '\efi\efistub\bootx64.efi' -u "root=/dev/mapper/cl-root ro crashkernel=auto resume=/dev/mapper/cl-swap rd.lvm.lv=cl/root rd.lvm.lv=cl/swap initrd=\\EFI\\efistub\\initramfs.img"

или

EFI\grub\grubx64.efi

efibootmgr -c -d /dev/sda1 --label "Linux" --loader '\efi\EFI\efistub\bootx64.efi' -u "root=UUID=06c92b59-a126-4c60-8962-70d085e15a5b ro no\_timer\_check crashkernel=auto initrd=\\EFI\\efistub\\initramfs.img"

Меняем порядок загрузки

efibootmgr

efibootmgr --bootorder 0004,0000,0001,0002,0003,0005

delete boot number 4

efibootmgr -b 4 -B

efibootmgr --verbose # show with parameters

Пример добавления опций ядра через bcfg

<https://superuser.com/questions/1035522/how-to-add-booting-parameters-to-the-kernel-with-bcfg-from-the-efi-shell>

Дополнительная информация по EFISTUB

https://bmccarthy.net/2017/05/15/booting-efi-stub-fedora-25/

<https://habr.com/ru/post/197438/>

# Документация

загрузчики

<https://wiki.ubuntu.com/EFIBootLoaders>

<https://bugzilla.redhat.com/show_bug.cgi?id=1525228>

EFI Secure Boot

<https://habr.com/ru/post/308032/>

старая статья, но более менее точная

<https://habr.com/ru/post/113350/>

Пример миграции BIOS to EFI

<https://oded.blog/2017/11/13/fedora-bios-to-uefi/>

Пример установки/добавления EFI

<https://systemd.io/BOOT_LOADER_SPECIFICATION/>

<https://www.rodsbooks.com/efi-bootloaders/installation.html>

Про [BootLoaderSpec/](https://www.freedesktop.org/wiki/Specifications/BootLoaderSpec/) grub2-switch-to-blscfg

<https://www.freedesktop.org/wiki/Specifications/BootLoaderSpec/>

<https://tiantiankan.me/a/5d582695a7c6c7fc974bbc4b>

https://linuxconfig.org/how-to-remove-old-unused-kernels-on-centos-linux