# GRUB (Русский)

**Ссылки по теме**

- **GRUB Legacy (Русский)**
- **Arch Boot Process (Русский)**
- **Boot loaders**
- **Master Boot Record (Русский)**
- **GUID Partition Table**
- **Unified Extensible Firmware Interface (Русский)**
- **GRUB EFI Examples**

Загрузчик - первая программа, которая загружается с диска при старте компьютера, и отвечает за загрузку и передачу управления ядру ОС. Ядро, в свою очередь, запускает остальную часть операционной системы.

GRUB2 – **мультисистемный** модульный **программируемый** кроссплатформенный загрузчик, с поддержкой сети, множества файловых систем, таблиц разделов, логических томов, образов и архивов.

**GRUB2** - не путать с **GRUB Legacy** - это новая ветка загрузчика GRUB, GRand Uniform Bootloader. GRUB2 не использует код старой ветки GRUB (GRUB Legacy), и создан на основе проекта **PUPA**.

**Contents**

[hide]

# Предисловие

Релиз GRUB-2.00 уже вышел, но разработка продолжается. В репозиториях ArchLinux появляются самые свежие, в том числе и бета-версии GRUB, поэтому переустанавливать загрузчик (командой grub-install) и особенно обновлять файл конфигурации (командой grub-mkconfig) следует с осторожностью.

Если вы хотите стабильности - не переустанавливайте без необходимости загрузчик и не запускайте генератор конфига. Пакет **grub** сам по себе содержит лишь утилиты и файлы, необходимые для установки и настройки загрузчика, и обновления пакета не затрагивают загрузчик. Однако, существует некоторый риск в отношении файла grub.cfg, см. **Защита конфигурации**.

Если вы не хотите использовать GRUB2, можно установить **GRUB Legacy** из AUR.

### Замечание для текущих и бывших пользователей GRUB Legacy

- Для пользователей, непривычных к шелл-коду, в GRUB2 существует утилита **grub-mkconfig** уровня ОС, для автоматического создания файла конфигурации загрузчика с типовым меню.
- Однако и без автоматического конфигуратора **настройка GRUB2** не сложнее, чем GRUB Legacy.

- Обозначения устройств в GRUB2 отличаются от таковых в GRUB Legacy. Разделы теперь **нумеруются с 1** (как в Linux), а диски по-прежнему с 0. Например, первый раздел первого диска в GRUB2 обозначается как `hd0,1`.

- Существуют различия между командами GRUB Legacy и GRUB2. Подробнее о командах можно прочесть в этой статье и в **официальном руководстве по GRUB2**.

- GRUB2 теперь *модульный*, и не имеет постоянных образов stage1_5 и stage2. Роль stage1_5 в GRUB2 играет образ core.img/core.efi, при каждой **установке загручика**собираемый из ядра GRUB2 и модулей, необходимых для доступа к файловой системе. Остальные модули загружаются из ФС по мере надобности, расширяя функциональность загрузчика.

# Требования к диску

### Загрузчик в BIOS

Сама по себе BIOS, *как правило*, не требует наличия на диске каких-либо таблиц разделов.

**Примечание:** Некоторые реализации BIOS могут проверять содержимое MBR, чтобы определить тип носителя *(особенно это относится к USB-дискам и флешкам)*. Носитель без MBR, или носитель с очень маленьким первым *(по номеру)* разделом типа FAT, может быть ошибочно принят некоторыми версиями BIOS за USB-флоппи-дисковод, что может сделать невозможной нормальную загрузку с него.

**Примечание:** Некоторые реализации BIOS могут отказаться запускать код загрузчика, если ни один из разделов MBR не отмечен флагом "bootable". Несмотря на это, BIOS может запустить код загрузчика только из бут-сектора диска *(сектор MBR)*, но не раздела, а после запуска кода GRUB уже не важно, на какой из разделов установлен флаг.

Участие BIOS в процессе загрузки сводится к считыванию с диска его начального сектора размером в 512 байт, проверке наличия в конце сектора сигнатуры `55AA`, и запуску содержимого как исполняемого кода. Так как в 512 байт невозможно уместить сколь-нибудь сложную программу, единственное, что может сделать код начального сектора – загружать с диска другие сектора, на чтение файлов он не способен.

Поэтому, для успешной загрузки в режиме BIOS, используемый способ разметки диска должен предусматривать:
1) встраивание кода загрузчика в начальный сектор;
2) выделение на диске специальной, не занятой файловыми системами области (группы секторов), из которой будет загружаться стартовый образ загрузчика, умеющий читать хотя бы одну файловую систему.

Программа установки GRUB2 умеет встраивать код BIOS-версии загрузчика на диски с таблицами разделов **MBR** и **GPT**, которые удовлетворяют обоим требованиям.

В обоих случаях, для встраивания используются два образа:

- **boot.img**, код которого встраивается в начальный сектор диска,
- **core.img**, собираемый программой установки из ядра GRUB и модулей доступа к используемой на диске таблице разделов и файловой системе.

**Требования GRUB2-BIOS к MBR**

Сама по себе таблица разделов на диске с разметкой **MBR** занимает 68 байт в конце начального сектора диска, последние два байта занимает "загрузочная" сигнатура `55AA`. Оставшиеся 442 байта с начала сектора – и есть то место, в которое встраивается код загрузчика из образа `boot.img`.

Для встраивания образа загрузчика `core.img` на диск с MBR, используется промежуток между начальным сектором с MBR и самым первым *(по расположению)* разделом. Если ближайший к началу диска раздел начинается с сектора 63 или больше,

```
# fdisk -l /dev/sda
...
Device    Boot     Start       End    Blocks  Id System
/dev/sda1             63   2120579   1060258+ 82 Linux swap / Solaris
...
```

то в большинстве случаев этого места (31КБ) вполне достаточно для встраивания GRUB2. Размер образа `core.img` зависит от размера модулей для чтения используемых таблицы разделов и ФС. В частности, для комбинации MBR+ext4 он составляет всего 25КБ.

Однако, в некоторых более сложных случаях, размер образа может оказаться больше 31КБ, и тогда отступа в 63 сектора уже не хватит. Поэтому современные версии fdisk по-умолчанию предлагают создавать первый раздел с гораздо большим отступом (1МБ), начиная с 2048 сектора.

**Требования GRUB2-BIOS к GPT**

Начальный сектор диска с разметкой **GPT** зарезервирован, как ни странно, для MBR, которая обычно используется для совместимости, но точно так же оставляет в секторе те же 442 байта, достаточные для встраивания кода загрузчика из `boot.img`. Сама GPT располагается на следующих секторах.

В отличии от MBR, GPT предусматривает возможность создания на диске специального раздела для встраивания BIOS-загрузчика. Раздел **BIOS boot partition** имеет `GUID=21686148-6449-6e6f-744e656564454649`, и может быть создан в **fdisk** как раздел типа **4**, или в **gdisk** как раздел типа **EF02**. Этот раздел **не должен содержать никакой файловой системы**, иначе она будет затёрта при установке загрузчика. Номер раздела может быть любым, расположение также практически любым. На больших

дисках рекомендуется располагать BIOS boot partition в пределах первых 2ТБ, поскольку средства BIOS, скорей всего, не позволят прочесть более дальние сектора.

Если такой раздел на диске с GPT создан, программа установки GRUB2 автоматически найдёт его и использует для встраивания стартового образа BIOS-версии загрузчика. Минимальные требования к размеру раздела те же, что и для просвета перед первым разделом в случае MBR – на это место должен поместиться образ `core.img`. Так как расположение BIOS boot partition **не привязано к началу диска**, создать его с размером порядка 1МБ будет несложно, и более чем достаточно в любом случае.

## Загрузчик в UEFI

В отличии от BIOS, **UEFI** загружает образ загрузчика из файла, и поэтому предъявляет определённые требования к таблице разделов, разделу, файловой системе, и к содержимому загружаемого файла.

**Таблица разделов**

Большинство реализаций UEFI поддерживает таблицы разделов **GPT** и **MBR**. Некоторые, однако, могут **не** поддерживать MBR, а некоторые могут поддерживать другие таблицы разделов, специфичные для производителя.

**Загрузочный раздел**

Загрузочный раздел UEFI называется "**EFI System Partition**", он же ESP, он же EFISYS. На каждом диске может быть не более одного такого раздела.

- В GPT он должен иметь тип **EF00** (`GUID=C12A7328-F81F-11D2-BA4B-00A0C93EC93B`), и может располагаться в любом месте диска под любым номером.
- К загрузочному разделу в MBR требования более жесткие: он должен одновременно иметь тип **EF**, флаг "**bootable**", и быть **первым**.

**Файловая система**

Большинство реализаций UEFI работают с загрузочным разделом с файловой системой FAT любой разрядности – FAT12, FAT16 и FAT32. Некоторые реализации могут требовать только FAT32, некоторые могут поддерживать другие ФС, специфичные для производителя.

**Загрузочный образ**

Загружаемый файл должен быть оформлен как EFI-приложение, а значит иметь унаследованный от DOS и Windows бинарный формат MZ/PE (Portable Executable) и **соответствовать архитектуре UEFI**. Все реализации UEFI для платформы PC имеют архитектуру **x86_64**, а значит и сборка загрузчика обязательно должна быть **под эту архитектуру**. Сборка загрузчика под i386 может быть полезна только на некоторых специфических машинах, в основном это старые компьютеры фирмы Apple, а также некоторые миниатюрные устройства на мобильных SoC производства Intel.

**Имя файла по-умолчанию**

UEFI архитектуры x86_64 автоматически находит на загрузочном разделе и запускает файл `\EFI\BOOT\BOOTX64.EFI`, который может быть стартовым образом загрузчика.

**Загрузочные записи**

На загрузочный раздел можно записать больше одного EFI-приложения, каждое под своим именем и в свой каталог. Чтобы можно было запускать их при старте, UEFI обычно предоставляет через специальный программный интерфейс доступ к загрузочным записям – особым переменным (**UEFI Variables**), хрянящимся в энергонезависимой памяти материнской платы (NVRAM). Каждая загрузочная запись содержит:

- видимый в пользовательском интерфейсе UEFI заголовок,
- тип таблицы разделов,
- ID таблицы разделов,
- расположение и размер загрузочного раздела,
- путь к исполняемому файлу EFI-приложения.

Кроме самих загрузочных записей, в NVRAM задаётся порядок их проверки при загрузке, который может быть изменён пользователем.

Программа установки GRUB2 будет пытаться создать загрузочную запись с помощью пакета **efibootmgr**, сохранив в ней параметры для запуска созданного ей образа загрузчика.

# Рекомендации по установке загрузчика

## Выбор раздела

Проще всего установить GRUB2 в корневой раздел. По-умолчанию для этого используется каталог с загрузочными образами ядра `/boot`, но можно установить каталог `grub/` в корень `/`, или в другое место в пределах раздела. GRUB2 обладает средствами для чтения нескольких типов таблиц разделов, логических томов, образов дисков, множества файловых систем и архивов. В большинстве случаев он может читать свои файлы и загружать ОС практически с любого раздела и диска.

- Только в тех случаях, когда ArchLinux установлен на диск, недоступный для чтения средствами BIOS или **UEFI**, которыми пользуется загрузчик, либо если для корневого раздела использована слишком новая, экзотическая, или зашифрованная файловая система, не читаемая GRUB2, может потребоваться вынос каталога `/boot` на отдельный диск либо раздел, и установка загрузчика на него.
- Если загрузчик в состоянии прочесть корневой раздел, создание отдельного раздела для `/boot` не нужно и **не рекомендуется**.

Если на компьютере установлено несколько операционных систем или диструбутивов, может оказаться удобным сделать один системонезависимый загрузчик, и установить его на отдельный диск или раздел. В этом случае стоит учесть следующее:

- Не требуется монтировать этот раздел в `/boot` или переносить в него образы ядра – GRUB2 и так может загрузить их практически из любого раздела.
- Не следует переустанавливать загрузчик без особой необходимости – от него зависит загрузка всех ОС на машине.
- Не рекомендуется использовать для общесистемного загрузчика генератор конфигурации – даже если забыть о его "сырости" и ненадёжности, скорей всего, конфигуратор будет работать только в одной системе, а в остальных всё равно придётся редактировать конфиг загрузчика. Вносить правки гораздо легче в простой и понятный конфиг, чем в громоздкий продукт деятельности конфгуратора.

## Выбор таблицы разделов

- Если на диске уже есть таблица разделов, и её возможностей для ваших целей достаточно, то нет никакого смысла её менять.

- Если вы устанавливаете систему на чистый диск, и выбираете таблицу разделов, то с точки зрения установки GRUB2 предпочтительнее GPT, которая позволяет выделить для встраивания загрузчика раздел в любом месте диска. Это общая рекомендация, она действительна как для BIOS, так и для UEFI.

- Если на этот диск планируется установка ОС, не поддерживающих GPT, но поддерживающих MBR, вам придётся выбрать MBR, или в крайнем случае, гибридную разметку диска MBR+GPT.
  **Важно:** Ни одна из версий Windows в режиме BIOS **не поддерживает** загрузку с GPT. Версии Windows, поддерживающие UEFI, имеют по два варианта загрузки (и загрузчиков): только с MBR в режиме BIOS, и только с GPT в режиме UEFI. См. **Windows**.

- Если на диске уже имеется MBR, но в ней недостаточно места перед первым разделом для встраивания GRUB2, и освобождение этого места проблематично, одним из выходов может стать преобразование MBR в GPT, например, с помощью программы **gdisk**. После этого на диске можно в любом доступном месте создать **BIOS boot partition**, и использовать её для встраивания загрузчика.

- Если UEFI вашей машины поддерживает загрузку только с GPT, выбор очевиден.

- Если вы устанавливаете ArchLinux на переносной носитель (флешку или USB-диск), то для универсальности вы можете использовать на ней GPT, и установить в неё сразу две сборки GRUB2 – i386-pc и x86_64-efi. В этом случае на носителе понадобится создать два загрузочных раздела – **EFISYS** для UEFI, и **BIOS boot partition**для BIOS. Файлы со сборками для каждой из архитектур установятся в отдельные каталоги внутри `grub/`, и не помешают друг другу. Файл конфигурации тоже можно использовать общий, но тогда для некоторых специфических настроек и действий понадобятся дополнительные проверки, описанные ниже.

- Если требуется сохранить возможность использования переносного носителя в Windows, включая XP, после создания на нём GPT и установки UEFI-загрузчика, можно использовать для этого гибридную разметку GPT+MBR, с описанием в последней только FAT-раздела с данными, и обязательно под номером 1. Остальное место на носителе в MBR должно быть покрыто "защитными" разделами с кодом 0xEE. Сделать всё это можно программой **gdisk**. GRUB2 в этом случае будет устанавливаться всегда только на GPT.

# Установка

## Установка GRUB2 в процессе установки Arch Linux

Чтобы установить GRUB2 в процессе установки, предварительно требуется смонтировать корневой раздел устанавливаемого Arch, *(а в него boot-раздел, если требуется)*, и выполнить команду **arch-chroot**.

## Установка пакета

Файлы и утилиты для установки GRUB2 содержатся в пакете **grub**, и устанавливаются командой

```
pacman -S grub
```

## Установка загрузчика

### Установка BIOS-версии загрузчика

### Полноценная установка для BIOS

- Для установки GRUB2 нужно выполнить команду от root:

```
grub-install /dev/sda
```

где `/dev/sda` это устройство *(не раздел!)* для установки загрузочных образов GRUB. Файлы загрузчика будут установлены в каталог `/boot`. Код GRUB (`boot.img`) будет встроен в начальный сектор, а загрузочный образ `core.img` – в просвет перед первым разделом **MBR**, или BIOS boot partition для **GPT**.

- Если при выполнении команды происходит ошибка, попробуйте добавить **--recheck** в аргументы как показано ниже:

```
grub-install --recheck /dev/sda
```

- Если требуется установить файлы загрузчика в другой каталог, его можно указать в опции **--boot-directory** . С этой опцией можно легко установить GRUB2 на диск с другой системой *(устанавливаемой или исправляемой)* **без чрута**, достаточно правильно указать

текущий путь к смонтированному каталогу для установки и текущее имя устройства, примерно так:

```
grub-install --boot-directory=/mnt/boot /dev/sdb
```

- Если нужно установить BIOS-версию загрузчика из-под системы, загруженной в режиме **UEFI**, требуется принудительно задать программе установки нужную сборку GRUB:

```
grub-install --target=i386-pc /dev/sda
```

**Генерация загрузочного образа для BIOS без установки**

Новые версии GRUB2 генерируют загрузочный образ `core.img`, который может быть загружен не только кодом бут-сектора, но и **другими загрузчиками** *(GRUB2, GRUB Legacy, **syslinux**)*, в качестве ядра, совместимого со стандартом Multiboot. Например, другой экземпляр GRUB2 может запустить его командой `multiboot`, а GRUB Legacy командой `kernel`.

Чтобы заставить программу установки GRUB2 сгенерировать готовый образ `grub/i386-pc/core.img`, но не устанавливать его в таблицу разделов, можно применить вот такой хак:

```
grub-install --grub-setup=/bin/true /dev/sda
```

Обычно программа установки вызывает сначала `grub-mkimage` (передавая ему множество параметров, что неудобно делать вручную), чтобы сгенерировать образ, а потом запускает `grub-bios-setup`, чтобы установить boot.img и core.img в таблицу разделов. С помощью ключа **--grub-setup** можно подсунуть программе установки вместо grub-bios-setup заглушку (команду **true**), которая всегда возвращает код успешного завершения.

**Установка UEFI-версии загрузчика**

**В режиме UEFI**

Если Arch x86_64 уже загружен в режиме UEFI, **системный загрузочный раздел EFI** уже создан и смонтирован в `/boot/efi`, а **efibootmgr** уже установлен и работает, для установки загрузчика остаётся выполнить команду

```
grub-install
```

Программа установки сгенерирует стартовый образ GRUB2, оформленный в виде EFI-приложения, скопирует его в файл `/boot/efi/EFI/arch/grubx64.efi`, файлы загрузчика будут записаны в каталог `/boot/grub/`, в том числе модули в `/boot/grub/x86_64-efi/`, после чего будет сделана попытка с помощью `efibootmgr` создать в переменных UEFI загрузочную запись "arch" со ссылкой на файл `\EFI\arch\grubx64.efi`, которую можно будет выбрать при следующей загрузке и установить её по-умолчанию.

- Опция **--boot-directory** задаёт путь, по которому будет установлен каталог `grub/`, и где `grub-install` будет искать каталог `efi/` с загрузочным разделом EFI. По-умолчанию этот путь равен `/boot`. Если вы хотите установить каталог с файлами загрузчика в другое место, например в корень, используйте команду вида

```
grub-install --boot-directory=/
```

В этом случае файлы загрузчика установятся в каталог `/grub`, а загрузочный раздел раздел должен быть предварительно смонтирован в `/efi`.

- Опция **--efi-directory** принудительно задаёт каталог, в который смонтирован загрузочный раздел EFI. С её помощью можно указать точку монтирования этого раздела, не привязанную жестко к расположению каталога `grub/`.

- Опция **--bootloader-id** задаёт "ID загрузчика" – имя, под которым будет создаваться загрузочная запись GRUB, видимая при выборе варианта загрузки в интерфейсе UEFI. Под этим же именем в загрузочном разделе будет создан каталог с образом GRUB2. По-умолчанию это имя **arch** .

  Используя эти опции вместе, при желании можно **установить файлы загрузчика непосредственно в загрузочный раздел EFI**, например так:

```
grub-install --efi-directory=/boot/efi --boot-directory=/boot/efi/EFI --
bootloader-id=grub
```

  В этом примере загрузочный раздел EFI с каталогом `EFI/` заранее смонтирован в `/boot/efi`, а ID загрузчика полностью совпадает с именем каталога "grub". Поэтому и образ, и файлы загрузчика будут установлены в один и тот же каталог `EFI/grub/` на загрузочном разделе. Соответственно, в этом случае стартовый образ будет установлен в `/boot/efi/EFI/grub/grubx64.efi`, конфиг загрузчика должен быть в `/boot/efi/EFI/grub/grub.cfg`, а модули в каталоге `/boot/efi/EFI/grub/x86_64-efi/`.

  **Без доступа к UEFI**

- Если нужно установить EFI-версию GRUB2, действуя из системы, загруженной в режиме BIOS, либо в режиме UEFI для другой архитектуры, либо на сменный носитель *(флешку или переносной диск)*, требуется принудительно задать программе установки нужную сборку загрузчика с помощью опции **--target** .
- Кроме того, в этом случае не будет работать `efibootmgr`, и создать загрузочную запись со ссылкой на стартовый образ загрузчика из этой системы не удастся. В такой ситуации можно использовать дефолтный загрузочный путь `\EFI\BOOT\BOOTX64.EFI`, по которому UEFI самостоятельно найдёт стартовый образ, с помощью опции **--removable** :

```
grub-install --target=x86_64-efi --removable
```

  Опции **--boot-directory** и **--efi-directory** можно добавлять при надобности, как описано выше, их действие не изменится.

- Аналогично можно установить GRUB2 на флешку для загрузки в режиме UEFI:

```
grub-install --boot-directory=/mnt/sdb2/boot --efi-directory=/mnt/sdb1 --
target=x86_64-efi --removable
```

  В случае переносного носителя создание загрузочной записи в UEFI бессмысленно, поскольку загрузка будет происходить на другой машине, и использование дефолтного пути к загрузочному образу остаётся единственным вариантом.

- Если вы не хотите или не можете использовать для GRUB дефолтный путь на загрузочном разделе, можно использовать опцию **--no-nvram** , чтобы программа установки поместила загрузочный образ в отдельный *(не дефолтный)* каталог, но не пыталась вызывать efibootmgr:

```
grub-install --target=x86_64-efi --no-nvram
```

**Примечание:** В этом случае вам придётся самостоятельно создать в переменных UEFI загрузочную запись с актуальным путём к загрузочному образу GRUB, с помощью **UEFI Shell** или иных средств, либо запускать этот образ из другого загрузчика.

**Из 32-битного Arch**

Если на машине есть UEFI, но установленный ArchLinux имеет архитектуру i686, установить UEFI-версию GRUB можно, но есть дополнительное затруднение – сборка загрузчика **x86_64-efi** отсутствует в пакете "grub" для архитектуры i686. В пакете же для архитектуры x86_64 присутствуют все три сборки:

- **i386-pc** для BIOS,
- **i386-efi** для 32-битных прошивок UEFI *(встречается редко, в основном это старые машины фирмы Apple, некоторые HDMI-стики)*,
- **x86_64-efi** для 64-битных UEFI.

Перед тем, как устанавливать UEFI-версию GRUB из системы i686, придётся сначала **скачать пакет grub для x86_64**, и распаковать из него *(под рутом)* недостающую сборку загрузчика:

```
tar xvf /путь/grub-версия-x86_64.pkg.tar.xz -C / usr/lib/grub/x86_64-efi
```

После этого можно устанавливать загрузчик, как описано в предыдущей главе "**Без доступа к UEFI**".

**Установка в LVM**

Установка GRUB2 на диск с **LVM** происходит в целом так же, как и на диск без LVM.

GRUB2 читает файловые системы из логических томов LVM, как и из обычных разделов, поэтому никакого отдельного раздела /boot **вне LVM** для файлов GRUB2 и загружаемых им образов ядра не требуется. Однако, для встраивания стартовых образов GRUB2 по-прежнему нужна таблица разделов.

Для загрузки в режиме BIOS, в **MBR** никаких дополнительных разделов создавать не нужно, достаточно единственного раздела на весь диск, полностью отданного под LVM, и оставленного свободного места перед разделом для встраивания загрузчика.

Для загрузки в режиме **UEFI**, или загрузки с **GPT** в любом режиме, требуется как минимум один загрузочный раздел вне LVM для стартового образа GRUB2, но не для файлов загрузчика, и не для образов ядра *(не /boot )*.

Подробнее см. **Требования к диску**.

**Установка на образ диска**

В некоторых случаях может понадобиться установить GRUB2 на образ диска, например, для загрузки в виртуальной машине. К сожалению, в нынешних версиях *(на конец 2015г)* программа установки загрузчика **grub-install** по-умолчанию считает loop-устройство, через которое подключается образ, не имеющим разделов, а потому не ищет на нём таблицу разделов, и не включает в начальную загрузку модуль чтения таблицы разделов.

Возможно, в будущем это будет исправлено, а пока для обхода проблемы требуется явно потребовать добавить соответствующий модуль: опцией `--modules=part_msdos`для MBR, или `--modules=part_gpt` для GPT.

Пример установки GRUB2 на образ диска с MBR, и размещением файлов загрузчика в каталоге `/grub` на первом разделе:

```
[root@host ~]# losetup --show -P -f /home/user/VM/disk.img
/dev/loop0
[root@host ~]# ls -1 /dev/loop0*
/dev/loop0
```

```
/dev/loop0p1
/dev/loop0p2
[root@host ~]# mkdir -p /mnt/part1
[root@host ~]# mount /dev/loop0p1 /mnt/part1
[root@host ~]# grub-install --modules=part_msdos --boot-directory=/mnt/part1
/dev/loop0
```

### Переустановка загрузчика

Переустановка пакета **не** переустанавливает загрузчик.

Переустановка GRUB2 выполняется командой **grub-install**, и ничем не отличается от установки.

Переустановка GRUB2 может потребоваться в следующих случаях:

- Если установка другой ОС или другого дистрибутива затёрла стартовый код GRUB2.
- После преобразования таблицы разделов.
- После изменения нумерации раздела, на котором установлены файлы загрузчика.
- После переноса загрузчика на другой раздел или в другую файловую систему.
- Если вас не устраивает текущая версия загрузчика, и вы хотите её обновить.

Переустановка GRUB2 **не** требуется:

- При изменении файла конфигурации загрузчика.
- При обновлении ядра, установке другого ядра, или пересборке initramfs.
- При смене архитектуры установленного Arch с i686 на x86_64 без смены раздела и без форматирования корневой ФС.
- При обновлении пакета "grub".

### Перенос загрузчика

Загрузчик GRUB состоит из двух частей: *(внефайлового)* загрузочного кода в таблице разделов и файлов в каталоге /grub *(/boot/grub)*.

Вне зависимости от того, какие из этих частей переносятся, при переносе обязательно нужно **переустановить загрузчик** с помощью **grub-install** .

Если загрузчик переносится на другой диск, также может потребоваться изменение настроек BIOS/UEFI, чтобы загрузка начиналась с этого диска.

Загрузчик GRUB **не читает** fstab и **игнорирует** флаг bootable в MBR, поэтому любые манипуляции с ними не имеют никакого отношения к установке, переустановке или переносу GRUB.

### После установки

В настоящее время GRUB2, сразу после установки, готов к работе **только** в режиме консоли. *(Это будет продолжаться до тех пор, пока мейнтейнеры пакета "grub" не заменят бессмысленный дефолтный файл конфигурации загрузчика на рабочую статическую или автоматически генерируемую версию.)*

Чтобы получить при загрузке действующее меню GRUB2, требуется заменить файл **grub.cfg** самостоятельно. Речь об этом идёт в следующей части "Настройка".

## Настройка

### Главный файл конфигурации загрузчика

Главный файл конфигурации по-умолчанию находится в `/boot/grub/grub.cfg`.

- Если вы пользуетесь автоматическим генератором конфигурации `grub-mkconfig`, **не редактируйте главный файл конфигурации вручную** – сгенерированный код пригоден для загрузки, но громоздок и неудобен для редактирования, а все изменения будут стёрты при запуске конфигуратора.
- Соответственно, если вы собираетесь редактировать конфиг загрузчика сами – создайте его либо полностью заново, либо на основе примеров, и **не пытайтесь запустить grub-mkconfig** или использовать сгенерированный им код.

## Автоматическая конфигурация (grub-mkconfig)

### Генерация конфига

Команда **grub-mkconfig** может быть использована для генерации файла `grub.cfg`.

Для автоматического обнаружения ОС отличных от Linux установите пакет `os-prober`.

Для настройки конфигуратора используйте файл `/etc/default/grub` и файлы в каталоге `/etc/grub.d/`.

Если Вы хотите добавить свои пункты в меню GRUB, настроить их можно в файле `/etc/grub.d/40_custom`, либо в `/boot/grub/custom.cfg`.

Чтобы применить изменения, запустите команду:

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

Она не только создаст файл `/boot/grub/grub.cfg`, но и проверит конфигурацию на наличие ошибок.

**Важно:** Если вы запускаете конфигуратор при установке ArchLinux, убедитесь, что вы делаете это внутри **arch-chroot**, иначе программа будет читать и записывать файлы конфигурации текущей, а **не** устанавливаемой системы.

### Параметры конфигуратора

Дефолтный файл `/etc/default/grub` содержит параметры конфигуратора с настройками по-умолчанию, снабженные комментариями на английском языке. Ниже перечислены некоторые наиболее общие из них:

- `GRUB_DEFAULT` Номер или заголок пункта меню, выбранного по-умолчанию
- `GRUB_TIMEOUT` Время, после которого будет автоматически загружаться пункт по-умолчанию
- `GRUB_CMDLINE_LINUX` Параметры ядра Linux, добавляемые во все пункты меню.
- `GRUB_CMDLINE_LINUX_DEFAULT` Параметры ядра Linux, добавляемые только в пункты меню, сгенерированные без "recovery". В ArchLinux настройки автоконфигуратора по-умолчанию содержат `GRUB_DISABLE_RECOVERY=true`, поэтому фактически в каждый пункт меню добавляются параметры из обеих упомянутых строк.

Другие параметры конфигуратора вы можете найти ниже, в соответсвующих тематических главах.

## Прямая конфигурация

GRUB2 – программируемый загрузчик, и файл его конфигурации – это не файл настроек, а программа, выполняемая загрузчиком, скрипт. Как всякий скрипт, он может быть как очень простым, не сложнее menu.lst в GRUB Legacy, так и очень сложным, поскольку сложность скриптов практически ничем не ограничена.

Код скрипта конфигурации, генерируемый `grub-mkconfig`, обычно пригоден для загрузки в типовых случаях, но слишком громоздок, избыточен, непригоден для изучения, ограничен в возможностях, и создаёт ложное впечатление о "сложном конфиге GRUB2".

Ещё одно распространённое заблуждение происходит от надписи "DO NOT EDIT ..." в начале кода, генерируемого конфигуратором. Надпись правильная, но её смысл состоит лишь в том, что именно **этот**, автоматически сгенерированный код, действительно нет смысла

редактировать. Надпись стандартна для автоматических конфигураторов, и относится не к grub.cfg вообще, а только к продукту деятельности конфигуратора, безотносительно имени файла, в который его сохранили.

При запуске автоматического конфигуратора, выполняется набор скриптов, работающих на уровне ОС, которые генерируют скрипт конфигурации загрузчика. Уже этот, сгенерированный скрипт, выполняется на уровне загрузчика, и в свою очередь, генерирует меню GRUB. Такая схема предполагает некоторое упрощение типовой настройки GRUB, но ценой загромождения кода, снижения надёжности и гибкости.

Прямое написание скрипта даёт непосредственный доступ ко всем возможностям GRUB2, значительно большую гибкость, надёжность и стабильность, просто за счёт упрощения кода и устранения лишних звеньев в цепочке.

Язык конфигурации GRUB2 – сильно упрощённый UNIX-шелл, из которого убраны возможности перенаправления ввода-вывода, и добавлены команды, специфичные для загрузчика.

Автоматический генератор конфигурации изначально создавался для дистрибутивов Debian и Ubuntu, в которых используются версионные имена образов ядра, что заставило разработчиков этих дистрибутовов создавать автоматические генераторы конфигов вообще для всех используемых там загрузчиков.

В ArchLinux не используется версионное обновление ядер – имена образов ядра и initramfs для каждого пакета с ядром не меняются при обновлении, и файл конфигурации загрузчика не обновляется при обновлении ядра.

Более того, возможности скриптов GRUB2 позволяют средствами самого загрузчика, прямо перед загрузкой ОС, **генерировать меню** с переменным количеством строк, для поиска и загрузки всех установленных ядер ArchLinux, без изменения каких-либо файлов конфигурации. То же самое возможно и **для версионных ядер Debian и Ubuntu**.

Примеры кода конфигурации даны ниже, **подробный справочник** имеется на сайте GRUB.

Прежде чем приступать к написанию конфига, **крайне желательно защитить его** от возможной перезаписи при обновлении/переустановке пакета grub.

## Защита конфигурации

К сожалению, мейнтейнеры пакета **grub** **отказываются** убирать из него бессмысленный, по определению неработающий и никому не нужный "дефолтный" вариант файла grub.cfg, создающий небольшую, но постоянную угрозу перезаписи настоящего рабочего конфига GRUB, особенно в случае ошибок со стороны мейнтейнеров. Эта опасность не зависит от используемого метода конфигурации GRUB – и ручной, и автогенерированный конфиг может быть однажды случайно перезаписан или переименован при очередном обновлении пакета, и выяснится это с большой вероятностью только после перезагрузки.

### Вынос конфигурации в другой файл

Так как GRUB поддерживает модульность конфигурации, можно оставить в файле `grub.cfg` только одну строку со ссылкой на другой файл, например `menu.cfg`

```
/boot/grub/grub.cfg
```

```
. $prefix/menu.cfg
```

и в дальнейшем вместо `grub.cfg` править только `menu.cfg`

Для автоконфигурации в этом случае можно использовать команду

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

### Прямая блокировка grub.cfg

Чтобы защитить файл от любых изменений, присвойте ему атрибут immutable

```
chattr +i /boot/grub/grub.cfg
```

Блокировка снимается командой

```
chattr -i /boot/grub/grub.cfg
```

Если основная конфигурация уже вынесена в другой файл, блокировку grub.cfg достаточно установить однажды и больше не снимать.

Блокировка защитит файл от перезаписи скриптами установки пакетов. Чтобы избежать конфликта с файлом из пакета, добавьте его имя в строку `NoUpgrade` в `/etc/pacman.conf`:

```
/etc/pacman.conf

NoUpgrade = boot/grub/grub.cfg
```

**Перенос каталога grub**

Если каталог `grub/` со всеми файлами загрузчика расположен в корневом разделе, для защиты конфигурации можно переместить его из `/boot` в другое место в пределах раздела, проще всего прямо в корень :

```
mv /boot/grub /
grub-install --boot-directory=/ /dev/sda
```

Вместо ***/dev/sda*** используйте текущее имя диска для установки *(при установке для **UEFI** указывать его не нужно)*. Все файлы загрузчика после этого будут находиться в каталоге `/grub` , в том числе файл конфигурации `/grub/grub.cfg` . Образы ядра и initramfs останутся по-прежнему в `/boot` , и будут загружаться как обычно, правки путей не потребуется.

В модульных и многофайловых *(шрифты, темы)* конфигурациях вместо `/boot/grub` можно использовать в путях к файлам переменную `$prefix`, в этом случае конфиги будут работать правильно вне зависимости от текущего расположения файлов загрузчика.

Если вы используете автоконфигуратор, не забудьте после переноса заменить возможные упоминания `/boot/grub` на `/grub` в файлах `/etc/default/grub` и `/etc/grub.d/*`

Для генерации конфига после переноса можно пользоваться командой

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

## Синтаксис файла конфигурации GRUB

### Пример минимальной работающей конфигурации

Здесь только один пункт меню, загрузчик в корневом разделе, который передаётся ядру **меткой** Arch_root

```
/boot/grub/grub.cfg

set timeout=5
```

```
menuentry "Arch Linux" {
 linux /boot/vmlinuz-linux root=LABEL=Arch_root rw
 initrd /boot/initramfs-linux.img
}
```

**Основные команды и переменные**

Файл конфигурации состоит из команд, выполняемых по порядку, как в обычном шелл-скрипте.

- Строки или продолжения строк, начинающиеся с символа # , считаются комментариями.

- В GRUB2 используются обозначения дисков вида **hdX** и разделов вида **hdX,Y** , где **X** номер диска, **Y** номер раздела на нём. Нумерация разделов начинается с единицы *(как в ядре Linux)*, нумерация дисков – с нуля. Если обозначение диска или раздела используется само по себе, например в переменной **root**, оно пишется **без** скобок.
- Пути к файлам могут начинаться с обозначения диска или раздела в скобках, например `(hd0,1)/boot/vmlinuz` соответствует файлу `vmlinuz` в каталоге `/boot` на разделе `/dev/sda1`

- Если в пути к файлу не указан диск или раздел, подразумевается текущий диск или раздел из переменной **root** . При запуске GRUB2 эта переменная уже указывает на раздел, где хранятся все файлы загрузчика, и как правило, образы ядра и initramfs. Задавать переменную root в конфиге имеет смысл ТОЛЬКО для загрузки с ДРУГОГО раздела. Пример:

```
set root=hd0,1 # задаём раздел с другой ОС или другим загрузчиком
chainloader +1 # загружаем его бут-сектор
```

- Кроме файлов, GRUB2 позволяет обращаться напрямую к **цепочкам секторов** на диске или разделе, в формате **X+Y** , где **X** номер начального сектора цепочки, а **Y**количество секторов. Если цепочка начинается с первого (номер 0) сектора, её обозначение сокращается до `+Y`. Наиболее употребительная цепочка в конфигах GRUB – бут-сектор раздела или диска, обозначается как **+1**

- В обозначении раздела перед его номером может стоять слово, обозначающее его принадлежность к **таблице разделов** определённого типа, например `hd0,msdos3`означает третий раздел в **MBR** на нулевом диске, а `hd1,gpt2` означает второй раздел в **GPT** на следующем диске. GRUB2 всегда добавляет тип таблицы при выводе разделов командой `ls` и при записи `$prefix` в процессе установки, однако писать тип таблицы вручную не требуется – он определяется автоматически. Исключение может составлять только особо тяжелый случай двух разных таблиц разделов с разной нумерацией на одном диске, причём гибрид GPT+MBR таким случаем **не является** – GRUB2 игнорирует MBR, если на диске обнаружена GPT.

- Команда **menuentry** генерирует один пункт меню. Она задаёт видимый заголовок и список команд, которые выполнятся после выбора этого пункта меню:

```
menuentry "Заголовок" {
# команды
}
```

- Переменная **default** задаёт номер или заголовок пункта меню, выбранного по-умолчанию. Например, `set default=1` выбирает *второй* пункт. Пункты меню нумеруются с нуля.

- Переменная **timeout** определяет время в секундах, по истечении которого будет загружен пункт меню по-умолчанию, например, `set timeout=5`.
  **Примечание:** Если таймаут не задан, GRUB будет показывать меню неограниченно долго, и не начнёт загрузку ОС без вмешательства пользователя.

- Команда **linux** загружает образ ядра и параметры для него.

- Команда **initrd** загружает образ начальной корневой ФС. Если образов указано несколько, они будут загружены по очереди. Например, образ для обновления микрокода процессоров Intel загружается перед основным образом:

```
initrd /boot/intel-ucode.img /boot/initramfs-linux.img
```

- Команда **chainloader** предназначена для загрузки "по цепочке" образа другого загрузчика. В режиме BIOS это, как правило, boot-сектор или его копия в файле, в режиме **UEFI** это может быть любое EFI-приложение, даже не обязательно находящееся в доступном для UEFI разделе – главное, чтобы файл мог быть прочитан средствами GRUB, а переменная **root** при вызове команды указывала на доступный UEFI раздел.

  **Стандартный пример конфигурации**

  В этом примере так или иначе фигурируют три раздела:

- **hd0,1** – раздел Windows, указан явно в последнем пункте меню.

- **hd0,2** – boot-раздел , монтируется в `/boot` , поэтому /boot отсутствует в путях к файлам. При старте раздел уже находится в переменной root, поэтому отдельно никак не задаётся, только подразумевается.

- **hd0,3** – корневой раздел Arch Linux , в Linux отображается как `/dev/sda3` , как и прописан в параметрах ядра.

```
/boot/grub/grub.cfg

# по-умолчанию выбран пункт меню 0
set default=0

# при бездействии пользователя он загрузится через 5 секунд
set timeout=5

# пункт меню номер 0
menuentry "Arch Linux" {
 linux /vmlinuz-linux root=/dev/sda3 rw
 initrd /initramfs-linux.img
}

# пункт меню номер 1
menuentry "Windows XP" {
 chainloader (hd0,1)+1
```

```
}
```

**Примечание:** Если образы ядра и initramfs лежат не в отдельном разделе, а в каталоге `'/boot'`, он должен быть указан в путях к ним:

```
menuentry "Arch Linux" {
 linux /boot/vmlinuz-linux root=/dev/sda3 rw
 initrd /boot/initramfs-linux.img
}
```

**Вторичные конфиги, вложенные меню и смена контекста**

Как и UNIX-шелл, GRUB2 поддерживает три вида переменных: обычные, позиционные параметры и переменные окружения.

- **Обычные переменные** доступны командам конфига, также внутри вызываемых из него функций и пунктов меню, но не наследуются при смене контекста.
- **Переменные окружения** недоступны непосредственно, но наследуются и автоматически импортируются в обычные при смене контекста.
- **Позиционные параметры**, они же "параметры командной строки". Передаются при вызове функции или пункта меню как аргументы команды, внутри функции или пункта меню обозначаются цифрами по порядку, начиная с **$1**

Обычные переменные создаются в момент первого присвоения, командой `set переменная=значение`
При этом само слово **set** можно не писать:

```
head="Arch linux"
```

**Примечание:** В примерах конфигов в этой статье слово **set** использовано при присвоении встроенных переменных GRUB2, чтобы их было проще отличать от обычных, пользовательских переменных. На самом деле, использование `set` при присвоении не требуется ни для каких переменных.

Для уничтожения обычной переменной может быть использована команда **unset**

```
unset timeout
```

Обычные переменные могут быть экспортированы "в окружение" командой **export**

```
export a b c
```

**Примечание:** Некоторые встроенные переменные GRUB2, в частности **root** и **lang**, имеют свойство экспортироваться автоматически.

Так же как в шелл-скриптах, из одного конфига GRUB2 может вызван другой файл конфига.

- Команда **source** *(часто обозначается просто точкой `.`)* выполняет другой конфиг без смены контекста, после чего продолжает выполняться текущий. Пример:

```
. $prefix/custom.cfg
```

- Команда **configfile** запускает новый конфиг в новом контексте:

```
configfile /boot/grub/new.cfg
```

**Примечание:** Переменная **prefix** при запуске GRUB содержит полный путь к каталогу, в который установлен загрузчик. Задавать её явно обычно не требуется, однако `$prefix` удобно использовать при обращении к другим конфигам, так как именно в этом каталоге находится grub.cfg, и как правило, дополнительные файлы конфигурации.

- Команда **submenu** создаёт новый пункт меню, так же как и `menuentry`, и может иметь точно такой же набор параметров. Единственное отличие **submenu** состоит в том, что команды внутри него выполняются **в новом контексте**, так же как при вызове конфига через `configfile`. Соответственно, все новые пункты меню, создаваемые в новом контексте, добавляются в **новое меню**, отсюда и название команды.
- Возврат в старое меню *(и старый контекст, со старыми переменными)* из вложенного, может быть выполнен нажатием клавиши ESC.

**Примечание:** Несмотря на название, можно использовать команду **submenu** и без создания нового меню, просто для смены контекста. В большинстве примеров этой статьи команда `submenu` использована только для того, чтобы локально менять текущий раздел в переменной `$root`, не затрагивая других пунктов меню в случае возврата.

**Примечание:** В принципе, ничто не мешает создавать пункты меню и внутри `menuentry`, однако в этом случае новые пункты будут будут добавляться не в новое, а прямо в текущее меню.

## Постоянное именование устройств

### UUID

**Примечание:** Автоматический конфигуратор использует UUID по-умолчанию. Если вы хотите, чтобы он использовать классические имена устройств *(например, если у вас нет initramfs)*, вы можете запретить конфигуратору использовать UUID, раскомментировав в `/etc/default/grub` строку `GRUB_DISABLE_LINUX_UUID=true`

При загруженном Linux, узнать UUID имеющихся разделов можно с помощью команды **lsblk -f**

Полученный UUID корневого раздела можно вручную вставить в параметры ядра, примерно так: `root=UUID=355ccb5c-99e1-400d-b612-451f9247e35e`, но делать это имеет смысл, **только** если у вас есть отдельный boot-раздел.

Чаще всего и GRUB, и ядро находятся в одном том же корневом разделе, который уже и так находится в `$root`, остаётся получить его UUID в переменную для подстановки в параметры ядра:

```
probe --set=UUID --fs-uuid $root
```

Пример загрузки ядра с автоматической подстановкой UUID:

```
menuentry "Arch Linux" {
 probe -s UUID -u $root
 linux /boot/vmlinuz-linux root=UUID=$UUID rw
 initrd /boot/initramfs-linux.img
}
```

Параметры команды **probe** здесь те же самые, просто в сокращённой форме.

И только если образ ядра находится **не** в том разделе, что GRUB, то есть **не** в корневом, и **не** в boot-разделе, требуется отдельно указать его загрузчику. Чтобы сделать это,

используйте команду **search**. Так мы устанавливаем корневой раздел в переменную загрузчика `$root` через поиск по UUID, и этот же UUID подставляем в параметры ядра, если отдельного boot-раздела нет:

```
menuentry "Arch Linux" {
 UUID=355ccb5c-99e1-400d-b612-451f9247e35e
 search --fs-uuid $UUID --set root
 linux /boot/vmlinuz-linux root=UUID=$UUID rw
 initrd /boot/initramfs-linux.img
}
```

**Метки**

Метки - легко читаемые заголовки, присваиваемые файловым системам и не только:

```
e2label /dev/sda3 Arch_root      # ставим метку на ext2/3/4
swaplabel -L Arch_swap /dev/sda2 # ставим метку свап-разделу
mkswap -L Arch_swap /dev/sda2    # если свап "старый", пересоздаём с меткой
ntfslabel /dev/sda1 WindowsXP    # ставим метку на NTFS
fatlabel /dev/sda5 OTHERDATA     # ставим метку на FAT
```

Метку можно подставить в параметры ядра. Пример:

```
menuentry "Arch Linux" {
 linux /boot/vmlinuz-linux root=LABEL=Arch_root rw
 initrd /boot/initramfs-linux.img
}
```

**Примечание:** Если вы используете метки для поиска разделов при загрузке, позаботьтесь об их уникальности. К примеру, метки Arch, root или my_disk уникальностью не отличаются.

Если загрузчик находится в отдельном от ядра и корня разделе, корневой раздел можно указать через метку и загрузчику:

```
menuentry "Arch Linux" {
 search --label Arch_root --set root
 linux /boot/vmlinuz-linux root=LABEL=Arch_root rw
 initrd /boot/initramfs-linux.img
}
```

## Особые типы устройств

### LVM

- При использовании GRUB2 на **LVM**, как правило, никаких дополнительных команд в конфиге GRUB не требуется, достаточно просто передать ядру в параметре **root=**обозначение группы и тома LVM, принятое в Linux, то есть `/dev/mapper/Group-Name`:

```
menuentry "Arch Linux" {
 linux /vmlinuz-linux root=/dev/mapper/Arch-Root rw
```

```
 initrd /initramfs-linux.img
}
```

или `/dev/Group/Name`:

```
menuentry "Arch Linux" {
 linux /vmlinuz-linux root=/dev/Arch/Root rw resume=/dev/Arch/Swap
 initrd /initramfs-linux.img
}
```

- Только если GRUB находится на разделе, отдельном от раздела с ядрами, то есть **не** на корневом, и **не** на boot-разделе, может потребоваться указать ему нужный раздел с образами ядра и imitramfs. Группа и том LVM в формате GRUB2 задаются так:

```
set root=lvm/Group-Name
```

- И только если GRUB2 установлен **не** на LVM, а образы ядра он должен прочесть с тома LVM, может также потребоваться предварительно загрузить его модуль командой **insmod** :

```
menuentry "Arch Linux" {
 insmod lvm
 set root=lvm/Arch-Root
 linux /vmlinuz-linux root=/dev/mapper/Arch-Root rw
 initrd /initramfs-linux.img
}
```

### Загрузка других операционных систем

GRUB можно использовать для загрузки и других установленных на той же машине дистрибутивов и операционных систем. Для добавления каждого нового пункта загрузочного меню GRUB, в его конфиг добавляется своя секция menuentry. Примеры конфигурации дополнительных пунктов для запуска Linux и Windows приведены ниже.

Если вы используете автоконфигурацию GRUB, но хотите, чтобы в генерируемый конфиг добавлялись ваши пункты меню, написанные вручную, отредактируйте файл `/etc/grub.d/40_custom` и допишите их в конец этого файла. Всё его содержимое будет добавлено в конфиг загрузчика при запуске **grub-mkconfig**

**GNU/Linux**

В этом примере другой дистрибутив Linux загружается с раздела `sda2`:

```
submenu "Other Linux" {
 set root=hd0,2
 linux /boot/vmlinuz # добавьте сюда все нужные опции ядра
 initrd /boot/initrd.img # файл initrd, если он используется
}
```

**Windows**

Операционные системы Windows не поддерживают протокол Multiboot, и GRUB не может загружать Windows непосредственно. Чтобы загрузить Windows, нужно запустить её собственный загрузчик, который, в свою очередь, загружает ядро Windows и набор необходимых для старта драйверов, по списку, который он читает из реестра Windows.

Для BIOS и для **UEFI** в одной и той же версии Windows используются две различных версии загрузчика, каждая со своими особенностями. Windows, установленная в режиме BIOS, не имеет UEFI-загрузчика и не стартует в режиме UEFI, и наоборот – будучи установленной в режиме UEFI, Windows не стартует режиме BIOS.

**Windows в режиме BIOS**

Загрузчик Windows для режима BIOS поддерживает только одну таблицу разделов – **MBR**, и не поддерживает GPT, независимо от версии Windows. Кроме того, 32-битная версия Windows XP может быть запущена только в режиме BIOS.

Загрузчик Windows может быть запущен через бут-сектор раздела, на который он установлен:

```
menuentry "Windows" {
 chainloader (hd0,2)+1
}
```

либо напрямую из его файла, специальной командой **ntldr**

```
submenu "Windows XP" {
 set root=hd0,2
 ntldr /ntldr
}
```

В Windows Vista и выше файл загрузчика называется иначе, но может быть загружен той же командой:

```
submenu "Windows 7" {
 set root=hd0,3
 ntldr /bootmgr
}
```

**Windows в режиме UEFI**

Загрузчик Windows, установленной в режиме **UEFI**, поддерживает только одну таблицу разделов – **GPT**, и не поддерживает MBR.

Если GRUB2 установлен в режиме UEFI, его стартовый образ лежит в том же **ESP/EFISYS** разделе, что загрузчик Windows. Полный путь к стартовому образу GRUB2 режима UEFI указывает переменная `$cmdpath` с содержимым вида `(диск,раздел)/EFI/каталог/grubx64.efi`. Чтобы из этого пути получить диск и раздел, можно регулярным выражением обрезать скобки и путь, и записать результат в переменную `$root`. После этого загрузчик Windows запускается командой `chainloader`, как обычное EFI-приложение:

```
submenu "Windows 8" {
 regexp -s root '\((.+)\)' "$cmdpath"
 chainloader /EFI/Microsoft/Boot/bootmgfw.efi
```

```
}
```

**Запуск FreeDOS**

Аналогично командам **linux** и **ntldr**, в GRUB предусмотрена возможность прямой загрузки ядра FreeDOS, командой **freedos**, без использования оригинального загрузочного кода в MBR и бут-секторе:

```
menuentry "FreeDOS" {
 freedos /KERNEL.SYS
}
```

Это может быть удобно, когда требуется скопировать уже установленную, к примеру, на флешку, FreeDOS, используемую для запуска MHDD и подобных инструментов. Такой способ загрузки позволяет под Linux копировать файлы и каталоги однажды установленной FreeDOS на другие носители, в том числе "мультизагрузочные".*(Установка FreeDOS штатным способом, с записью её загрузочного кода в MBR и бут-сектор, требует каждый раз загружать саму FreeDOS, хотя бы в виртуальной машине.)*

**Запуск программ, работающих без ОС**

**Memtest86+**

Тест памяти **Memtest86+** (запускается только в режиме BIOS):

```
menuentry "Memtest86+" {
 linux16 /boot/memtest86+/memtest.bin
}
```

**EFI-приложения**

EFI-приложения можно запускать не только из интерфейса UEFI, но и из других EFI-приложений, в том числе из UEFI-версии GRUB2, с помощью команды **chainloader**. На примере загрузчика Windows это уже показано **выше**.

Чтобы запустить EFI-приложение, вовсе не обязательно класть его на раздел EFISYS, особенно если приложение большое, а на спецразделе очень мало места. Достаточно, чтобы файл приложения читался средствами GRUB, а переменная **root** указывала на EFISYS. В этом примере **UEFI Shell** запускается прямо из `/boot`:

```
submenu "UEFI Shell" {
 archroot=$root
 regexp -s root '\((.+)\)' "$cmdpath"
 chainloader ($archroot)/boot/Shell.efi
}
```

При выходе из UEFI Shell вы снова увидите меню GRUB.

**Debian, Ubuntu и другие дистрибутивы с версионным обновлением ядра**

В дистрибутивах с версионными ядрами, при каждом обновлении ядра автоматически вызывается генератор конфигурации загрузчика, поскольку имена загрузочных образов vmlinuz и initrd меняются при каждом обновлении. Это обстоятельство вынудило разработчиков Debian и Ubuntu сделать автоконфигуратор даже для первой версии GRUB, в которой обычно использовались только статические конфиги.

Если другой дистрибутив установлен на отдельный диск *(в режиме BIOS)*, и имеет собственную установку GRUB2, для его загрузки достаточно запустить другой загрузчик:

```
menuentry "Ubuntu" {
 chainloader (hd1)+1
}
```

В режиме **UEFI** несколько загрузчиков могут быть установлены и на один диск, и запускаться один из другого как EFI-приложения:

```
submenu "Ubuntu" {
 regexp -s root '\((.+)\)' "$cmdpath"
 chainloader /EFI/ubuntu/grubx64.efi
}
```

В некоторых случаях *(не всегда)* другую установку GRUB2 можно запустить в режиме BIOS с помощью команды **multiboot**, минуя встраивание образа в таблицу разделов:

```
sunmenu "Other Linux" {
 set root=hd0,6
 multiboot /boot/grub/i386-pc/core.img
}
```

К сожалению, часто встречается ситуация, когда другой дистрибутив установлен в режиме BIOS на тот же самый диск, в таблицу разделов которого можно установить только один загрузчик, и запустить второй проблематично. В этом случае остаётся либо использовать GRUB из состава того дистрибутива, либо учить штатный загрузчик ArchLinux работать с версионными ядрами.

Самый простой способ это сделать – подсунуть "своему" GRUB конфиг от чужого, в котором всё уже предусмотрено, не забыв перед этим сбросить некоторые переменные, которые могут вызвать проблемы:

```
submenu "Ubuntu" {
 unset lang
 unset gfxmode
 set root=hd0,2
 configfile /boot/grub/grub.cfg
}
```

Однако, есть возможность обойтись без чужих конфигов и загрузчиков, если использовать **динамическую генерацию меню загрузчиком**. В конфиг GRUB2 включается скрипт, генерирующий меню для Ubuntu:

```
. $prefix/ubuntu.cfg
```

Параметры для загрузки чужих ядер здесь вынесены в отдельный файл настроек:

```
usettings.cfg
```

```
uroot=hd0,2
boot=/boot
opts="root=LABEL=Ubuntu_root ro resume=LABEL=SwapU"
hpref="Ubuntu"
```

Сам скрипт:

ubuntu.cfg

```
function usave {
 if [ "$1" != "$usel" ] ; then
  usel="$1"
  save_env usel
 fi
}

function umenu {
 . $prefix/usettings.cfg

 kpref=$boot/vmlinuz-
 ipref=$boot/initrd.img-

 load_env
 default="$hpref $usel"
 if [ -n "$2" ] ; then default="$default $2" ; fi

 kernels=
 for kfile in "$uroot$kpref"* ; do
  k=
  regexp -s k "$kpref"'(.+)' "$kfile"
  kernels="$k $kernels"
 done

 for k in $kernels ; do
  ifile="$uroot$ipref$k"
  if [ -f "$ifile" ] ; then
   head="$hpref $k"
   if [ -n "$2" ] ; then head="$head $2" ; fi
   menuentry "$head" --source="usave $k
linux $uroot$kpref$k $opts $1
initrd $ifile"
  fi
 done
}

submenu "Ubuntu" --hotkey=u {
```

```
 insmod regexp

 submenu "Recovery mode" --hotkey=r {
  umenu "recovery nomodeset" "recovery mode"
 }

 umenu
}
```

Этот скрипт в конфиге GRUB динамически генерирует отдельное подменю со всеми имеющимися ядрами Ubuntu, и отдельно (для Ubuntu) запоминает последнее выбранное ядро.

**Прямая загрузка из образа диска**

GRUB2 может загружать образы ядер ОС в том числе из файлов-образов, отображаемых в псевдоустройство командой **loopback**. Однако следует иметь в виду, что псевдоустройство действует только в пределах загрузчика. В общем случае загрузка из образа выглядит примерно так:

```
loopback loop файл-образа
linux (loop)/путь/к/vmlinuz параметры
initrd (loop)/путь/к/initrd
```

После отображения образа в loop-устройство *(имя может быть любым, не только loop)*, можно средствами GRUB2 обращаться с ним так же, как и с физическими дисками – не только загружать файлы ядра и initrd, что происходит перед загрузкой, а к примеру, прочитать метку ФС образа (требуется для образа Arch):

```
probe -s isolabel -l loop
```

Так как средства загрузки с loopback-устройства в разных дистрибутивах различаются, **для каждого из них требуется передавать при загрузке параметры, специфичные для данного дистрибутива**.

- Установочный образ Arch требует метку ФС образа в параметре ядра `archisolabel=`, и линуксовое устройство раздела, на котором он лежит, в параметре `img_dev=`*(универсальнее всего прочесть и передать его UUID, но можно использовать и метку, если она есть)*.
- Образ Ubuntu довольствуется лишь путём к образу на диске, а остальное находит сам.

В некоторых дистрибутивах средства загрузки из образа могут и вовсе отсутствовать, поэтому не все существующие загрузочные образы могут быть использованы подобным образом.

Во всех приведённых ниже примерах предполагается, что **GRUB2 установлен на тот же раздел диска или флешки, где лежат образы** *(в противном случае см. **ниже**)*. Такой способ установки позволяет записать на одну флешку или внешний диск один или несколько образов, при этом сохранив возможность использования оставшегося места на носителе по прямому назначению, с сохранением предпочтительной файловой системы.

**Arch Linux ISO**

В этом примере GRUB2 загружает официальный установочный образ ArchLinux.

```
grub.cfg
```

```
dir=
arch=x86_64
insmod regexp                # для шаблонов * в именах файлов
probe -s root_uuid -u $root # получаем UUID для подстановки в img_dev=

for iso in $dir/archlinux-*-$arch.iso ; do      # ищем образ(ы) Arch по
шаблону
 if [ ! -f "$iso" ] ; then continue; fi        # только если образ существует
 regexp -s build 'archlinux-(.+)-'$arch "$iso"  # получаем из имени образа
дату сборки

  menuentry "Arch Linux ISO $build $arch" --source="\
loopback loop $iso
probe -s isolabel -l loop
linux (loop)/arch/boot/$arch/vmlinuz archisolabel=\$isolabel
img_dev=/dev/disk/by-uuid/$root_uuid img_loop=$iso earlymodules=loop
initrd (loop)/arch/boot/$arch/archiso.img"

done
```

Файл образа с именем вида `archlinux-YYYY.MM.DD-x86_64.iso` должен лежать в корне раздела. В этом случае текст файла конфигурации можно использовать "как есть", без каких-либо правок – загрузчик сам определит конкретное имя образа (или образов), сам прочитает UUID раздела, метку образа, и при загрузке передаст всё это дистрибутиву через параметры ядра.

Если же вы хотите использовать для образа (образов) особый каталог, впишите путь к нему (относительно корня раздела) в строку `dir=`, например

```
dir=/images
```

**Ubuntu desktop ISO**

В этом примере GRUB2 загружает любые ISO-образы Ubuntu, кроме серверных. В сгенерированном меню будут варианты загрузки для всех найденных образов Ubuntu.

```
grub.cfg

dir=
insmod regexp

for iso in $dir/*ubuntu-*.iso ; do         # ищем образ(ы) Ubuntu
 if [ ! -f "$iso" ] ; then continue; fi    # только если образ существует
 regexp -s name '.*/(.+)\.iso' "$iso"                # выделяем только имя
 if regexp server "$name" ; then continue ; fi              # Ubuntu Server
не трогаем, там всё иначе
 if regexp amd64 "$name" ; then efi='.efi' ; else efi= ; fi # для x86_64
образ ядра vmlinuz.efi
```

```
  menuentry "$name" --source="\
loopback loop $iso
linux (loop)/casper/vmlinuz$efi boot=casper iso-scan/filename=$iso noeject --
initrd (loop)/casper/initrd.lz"

done
```

Как и в предыдущем примере, код не требует никаких правок, если файлы образов лежат в корне раздела, но можно вписать отдельный каталог для них в строку `dir=`

**Загрузка образов с отдельного раздела встроенного диска**

Если образы требуется загружать с некорневого раздела встроенного жесткого диска *(например,* `/home` *)*, можно использовать тот же код, создав для него подменю:

```
submenu "ISO boot" {
 dir=
 set root=hd0,5 # здесь нужно вписать настоящий номер раздела
 # или найти его по метке, например Arch_home:
 # search -s root -l Arch_home
 insmod regexp
 probe -s root_uuid -u $root
 arch=x86_64

 # вместо этой строки вставляем основной код, начиная с for
}
```

Если раздел требуется задать по UUID, можно сделать так:

```
submenu "ISO boot" {
 dir=
 root_uuid=ef6daeca-9278-40df-8c3b-55cf093ab215
 search -s root -u $root_uuid
 insmod regexp
 arch=x86_64

 # вместо этой строки вставляем основной код, начиная с for
}
```

## Защита загрузчика паролем

### О защите загрузчика

По-умолчанию GRUB2 предоставляет любому пользователю полный доступ ко всем своим возможностям, включающим в себя не только выполнение любых пунктов меню, но и изменение их кода перед выполнением, а также терминал, позволяющий вручную выполнить любые команды загрузчика.

Всё это может помочь при настройке и восстановлении системы, однако те же самые инструменты существенно облегчают любому посетителю взлом системы. Изменение параметров ядра может быть использовано для получения полномочий root без ввода пароля,

а загрузка с внешнего носителя – для получения полного доступа ко всем незашифрованным данным.

GRUB2 включает в себя средства ограничения доступа к загрузчику. С их помощью, в сочетании с ограничением доступа к BIOS, и запретом в BIOS на загрузку с любых внешних носителей, можно сильно затруднить несанкционированный доступ к системе, кроме случая вскрытия корпуса для извлечения дисков или сброса настроек BIOS.

**Реализация паролей в GRUB2**

Пользователи в GRUB2 делятся на три категории:

- **Гости**, они же неавторизованные пользователи. Могут выполнять только не защищённые паролем пункты меню.
- **Авторизованные пользователи**. Могут выполнять разрешенные для них пункты меню.
- **Администраторы**. Имеют полный доступ – могут выполнять любые пункты меню, редактировать их перед выполнением, и открывать командный терминал.

Для управления доступом к пунктам меню, команды **menuentry** и **submenu** поддерживают следующие опции:
`--users=` позволяет задать список пользователей, которым разрешено выполнять этот пункт меню
`--unrestricted` разрешает выполнять этот пункт меню без авторизации.

**Важно:** После активации парольной защиты загрузчика, любые пункты меню, не имеющие ни одной из этих опций, будет доступны только администраторам.

Список администраторов задаётся в переменной **superusers**, например так:

```
set superusers=root
```

Пароль для каждого пользователя отдельно может быть задан в открытом виде командой

```
password пользователь пароль
```

либо в зашифрованном (хешированном) виде, командой

```
password_pbkdf2 пользователь хеш
```

Для хеширования пароля используется утилита **grub-mkpasswd-pbkdf2**. Запустив её и введя (дважды) пароль, можно получить его хеш, пригодный для вставки в конфиг GRUB2.

Чтобы защитить пароли и хеши от просмотра, можно задать права на файл с конфигом GRUB в виде root:root 600, либо вынести команды с паролями в отдельный файл

```
. $prefix/secret.cfg
```

и ограничить доступ только к нему.

**Внедрение паролей в генерируемый конфиг**

Конфигуратор `grub-mkconfig` сам не умеет ограничивать доступ к загрузчику, хотя и устанавливает права доступа к конфигу в `-rw------- root root`, что имеет смысл только для сохранения паролей.

Есть возможность вставить вручную написанный фрагмент конфига с паролями, добавив в конец файла `/etc/grub.d/00_header` примерно такие строки:

```
cat << EOF

set superusers=root
password_pbkdf2 root
grub.pbkdf2.sha512.10000.C2DDC47FC5C7341CE73DBD6728E8D29A.AA5A1DEA93E23358E90
8301439DEC488

EOF
```

**Важно:** После генерации такого конфига ВСЕ пункты меню станут доступны только администратору!

Более гибкая настройка доступа к загрузчику, например разрешение штатной загрузки системы без ввода пароля, возможна либо путём добавления своих пунктов меню с опцией `--unrestricted` в конец файла `/etc/grub.d/40_custom`, либо при самостоятельном написании всего конфига GRUB.

**Пример конфига с паролями**

Здесь "Arch Linux" разрешено загружать без авторизации, "Windows" разрешено загружать пользователю second с паролем dnjhjq, а активировать "Boot next disk", позволяющий загрузиться с подключённой флешки, может только администратор по имени root, который никому не сказал свой пароль.

```
set default=0
set timeout=5
set superusers=root

password second dnjhjq
password_pbkdf2 root
grub.pbkdf2.sha512.10000.C2DDC47FC5C7341CE73DBD6728E8D29A.AA5A1DEA93E23358E90
8301439DEC488

menuentry "Arch Linux" --unrestricted {
 linux /boot/vmlinuz-linux root=LABEL=Arch_root rw resume=LABEL=Arch_swap
 initrd /boot/initramfs-linux.img
}

submenu "Windows" --users=second {
 set root=hd0,1
 chainloader +1
}

submenu "Boot next disk" {
 set root=hd1
 chainloader +1
}
```

## Визуальная настройка

GRUB2 предоставляет возможность менять способы отображения и внешний вид меню.

**Цвета меню**

Цвета меню GRUB2 задаются в переменных **menu_color_normal** (общие цвета текста/фона) и **menu_color_highlight** (цвета текста/фона выделенной строки). Например, цвета по умолчанию для Arch задаются в конфиге GRUB (grub.cfg) так:

```
set menu_color_normal=light-blue/black
set menu_color_highlight=light-cyan/blue
```

или в настройках `/etc/default/grub` конфигуратора `grub-mkconfig` так:

```
GRUB_COLOR_NORMAL="light-blue/black"
GRUB_COLOR_HIGHLIGHT="light-cyan/blue"
```

Список доступных в GRUB2 цветов можно найти **на сайте**.

**Скрытое меню**

`grub-mkconfig` умеет "скрывать" меню так, что оно появляется на экране только при нажатии клавиши ESC до истечения таймаута. Чтобы использовать эту его возможность, нужно раскомментировать в `/etc/default/grub` строки

```
GRUB_HIDDEN_TIMEOUT=5
GRUB_HIDDEN_TIMEOUT_QUIET=true
```

В конфиг GRUB (grub.cfg) для получения аналогичного результата можно добавить такой код:

```
set timeout=5

echo -n "Press ESC to see the menu... "
if sleep --verbose --interruptible $timeout
then set timeout=0
else unset timeout
fi
```

Здесь не задаётся отдельный таймаут для скрытого и видимого меню, а используется общий из переменной timeout. После вывода надписи запускается ожидание **sleep** с в выводом обратного отсчёта `--verbose` и прерыванием по ESC `--interruptible` . Если отсчёт не был прерван, таймаут уменьшается до нуля `set timeout=0`, и меню проскакивает без вывода на экран. В противном случае таймаут отключается вообще `unset timeout`, чтобы отсчёт не начался повторно после показа меню.

Если вы используете в загрузчике графический режим с обоями и шрифтами, следует решить, нужно ли их инициализировать **до** начала отсчёта, или только если отсчёт прерван пользователем.

**Настройка параметров режима экрана**

**Проверка доступных режимов экрана**

GRUB2 может работать в текстовом и графических режимах экрана. Актуальный список режимов, доступных средствами BIOS или **UEFI** на конкретной машине, можно можно получить, выполнив команду **videoinfo** в **консоли загрузчика**. На разных машинах, с разными

графическими адаптерами, прошивками и мониторами, этот список может существенно отличаться.

Выбрав нужный режим, желательно предварительно его проверить, с помощью команды **videotest *режим*** в той же консоли, примерно так:

```
videotest 1280x1024x32
```

Тестовая картинка должна отображаться на экране до нажатия на любую клавишу. Если вы видите её, значит тест пройден.

**Примечание:** Тест графики можно запускать **только** из **текстового режима**.

**Важно:** Автоматический генератор конфигурации включает графический режим по-умолчанию, без проверки. Если графический режим уже задан в конфигурации неудачно, и вы не видите на экране интерфейса GRUB2 при загрузке, включите **текстовый режим** в настройках конфигуратора.

**Текстовый режим**

GRUB2 стартует в текстовом режиме. Если нужно вернуться в текстовый режим из графического, можно использовать в скрипте конфигурации, или прямо в **консоли GRUB**, следующие команды:

```
unset lang
terminal_output console
```

Если вы используете автоматический конфигуратор, в настройках которого стоит **auto** по-умолчанию, загрузчик после старта будет переходить графический режим. Чтобы изменить это его поведение, можно задать для загрузчика текстовый режим:

```
GRUB_GFXMODE=console
```

или же задать текстовый режим только для загрузки ядра:

```
GRUB_GFXPAYLOAD_LINUX=text
```

**Графические режимы**

GRUB2 позволяет задать режим экрана для самого загрузчика в переменной **gfxmode** , и отдельно режим, который будет использоваться ядром Linux при загрузке, через переменную **gfxpayload**

Эти режимы могут быть разными:

```
grub.cfg

set gfxmode=1024x768x32
set gfxpayload=1280x1024x16
```

```
/etc/default/grub

GRUB_GFXMODE=1024x768x32
```

```
GRUB_GFXPAYLOAD_LINUX=1280x1024x16
```

или одинаковыми:

```
grub.cfg

set gfxmode=1280x1024x32
set gfxpayload=$gfxmode
```

```
/etc/default/grub

GRUB_GFXMODE=1280x1024x32
GRUB_GFXPAYLOAD_LINUX=keep
```

Также можно задать только один из них, не задавая другой.

Режим экрана для ядра Linux будет установлен в начале загрузки ядра, никаких модулей GRUB для этого специально загружать не требуется.

Вместо жесткого указания конкретного режима, можно присвоить переменной слово **auto** , и в этом случае режим будет выбран автоматически, исходя из предпочтений BIOS видеокарты и предпочтительного режима монитора. Обычно `auto` соответствует максимальному из штатных режимов монитора, но в некоторых случаях нужный режим приходится выставлять вручную.

Существует также старый, специфический для BIOS, способ задать VESA-режим – через параметр ядра Linux **vga**, например `vga=790`

Возможные значения для параметра vga, в зависимости от количества цветов и пикселей на экране, можно найти в этой таблице

```
+-----+-------------------------------------------+
|     |  640x480    800x600   1024x768   1280x1024 |
+-----+-------------------------------------------+
| 256 | 0x301=769  0x303=771  0x305=773  0x307=775 |
| 32K | 0x310=784  0x313=787  0x316=790  0x319=793 |
| 64K | 0x311=785  0x314=788  0x317=791  0x31A=794 |
| 16M | 0x312=786  0x315=789  0x318=792  0x31B=795 |
+-----+-------------------------------------------+
```

Однако в некоторых версиях BIOS могут использоваться и другие коды режимов. Актуальные значения можно получить из вывода команды **videoinfo** в консоли GRUB или же выполнить в терминале запущенной ОС GNU/Linux команду

```
sudo hwinfo --framebuffer
```

Команда **hwinfo** доступна в репозитории **community**.

**Графический режим, шрифт и обои**

Установка переменной **gfxmode** сама по себе НЕ переключает GRUB в графический режим. Чтобы перейти в графику, требуется:

- задать режим в переменной **gfxmode**
- загрузить командой **loadfont** хотя бы один шрифт
- загрузить модуль поддержки графического режима (**vbe** для BIOS, либо **efi_gop** и **efi_uga** для **UEFI**)
- и модуль графического терминала **gfxterm**
- после всего этого запустить графический терминал командой **terminal_output gfxterm**

GRUB2 поддерживает растровые шрифты в собственном формате pf2. Шрифт Unifont включен в пакет grub под именем `unicode.pf2`, и при установке загрузчика командой **grub-install** автоматически копируется в каталог `/boot/grub/fonts`

В графическом режиме GRUB2 также позволяет установить обои командой **background_image** . Поддерживаются изображения в форматах tga, png и jpeg, для каждого из них требуется предварительно загрузить соответствующий модуль. Максимальный поддерживаемый размер изображения зависит от вашего оборудования.

В настройках конфигуратора `/etc/default/grub` обои можно задать так:

```
GRUB_BACKGROUND=/boot/grub/themes/starfield/starfield.png
```

Пример графического режима с установкой дефолтного шрифта и обоев из дефолтной темы оформления. Нужные модули графического режима выбираются автоматически.

```
if [ "$grub_platform" = "pc" ] ; then
 insmod vbe                          # загружаем модуль для BIOS
else
 insmod efi_gop                      # или для EFI
 insmod efi_uga
fi
loadfont $prefix/fonts/unicode.pf2  # загружаем шрифт
set gfxmode=auto
insmod gfxterm                       # загружаем модуль графического терминала
terminal_output gfxterm              # эта команда запускает графику
insmod png                           # для этих обой требуется модуль png
background_image $prefix/themes/starfield/starfield.png
```

После установки **юникодного** шрифта можно использовать в меню GRUB кириллицу, и даже перевести на русский язык встроенные сообщения загрузчика, присвоив переменной **lang** значение **ru** .

**Важно:** Не пытайтесь использовать русский язык в текстовом режиме! Если требуется возврат в текстовый терминал командой **terminal_output console** , следует предварительно отключить русский язык командой **unset lang** , иначе вместо надписей на русском останутся одни вопросы.

Чтобы использовать в GRUB свои шрифты, нужно предварительно конвертировать их в формат pf2. Для этого в состав пакета grub входит утилита **grub-mkfont** . Лучше всего она работает с растровыми шрифтами в формате BDF

```
grub-mkfont шрифт.bdf -o шрифт.pf2
```

и векторными в формате TTF

```
grub-mkfont шрифт.ttf -s размер -o шрифт.pf2
```

**Установка шрифта на примере Terminus**

Пакет terminus-font для GRUB не подходит, требуется скачать с **официального сайта** его исходники, они как раз в формате BDF. Далее остаётся распаковать архив, конвертировать файл со шрифтом нужного размера, и скопировать в каталог, доступный для GRUB:

```
 $ tar xf terminus-font-4.38.tar.gz
 $ cd terminus-font-4.38/
 $ ls *.bdf
ter-u12b.bdf  ter-u14v.bdf  ter-u18b.bdf  ter-u22b.bdf  ter-u28b.bdf
ter-u12n.bdf  ter-u16b.bdf  ter-u18n.bdf  ter-u22n.bdf  ter-u28n.bdf
ter-u14b.bdf  ter-u16n.bdf  ter-u20b.bdf  ter-u24b.bdf  ter-u32b.bdf
ter-u14n.bdf  ter-u16v.bdf  ter-u20n.bdf  ter-u24n.bdf  ter-u32n.bdf
 $ grub-mkfont -v ter-u16b.bdf -o ter-u16b.pf2
Font name: Terminus Bold 16
Max width: 8
Max height: 16
Font ascent: 12
Font descent: 4
Number of glyph: 879
 $ sudo cp ter-u16b.pf2 /boot/grub/fonts/
```

Пример фрагмента конфига GRUB со шрифтом Terminus и русским языком:

```
loadfont $prefix/fonts/ter-u16b.pf2
set gfxmode=auto
set lang=ru      # включаем русский язык
insmod vbe
insmod gfxterm
terminal_output gfxterm
```

**Проверка загрузки шрифтов**

Если шрифты должны были быть загружены, но на экране выглядят неправильно, проверить это можно в **консоли GRUB**, с помощью команды **lsfonts**, которая выводит список успешно загруженных шрифтов.

Некоторые файлы шрифтов могут быть устаревшими, и несовместимыми с установленной версией GRUB. Чтобы отдельно проверить загрузку каждого шрифта, можно попробовать вручную загружать их в консоли GRUB командами вида

```
loadfont $prefix/themes/тема/шрифт.pf2
```

При попытке загрузить несовместимый шрифт будут выдаваться сообщения об ошибках.

**Графическая тема оформления**

Даже после переключения в графический режим, меню GRUB отображается с помощью символов псевдографики. Альтернативный вариант отображения GRUB – **графические темы оформления**

Тема включает в себя файл описания `theme.txt`, а также может содержать элементы картинок для "рисования" меню, шрифты и обои.

В пакет grub входит дефолтная тема оформления, при установке командой `grub-install` она копируется в каталог `/boot/grub/themes/starfield/` . Тему для GRUB требуется прописывать в виде полного пути к файлу описания темы. В настройках конфигуратора `/etc/default/grub` это делается так:

```
GRUB_THEME="/boot/grub/themes/starfield/theme.txt"
```

В файлах конфигурации GRUB путь к файлу описания темы требуется записать в переменную **theme** ДО переключения в графический режим. Чтобы тема могла использовать указанные в ней элементы, до перехода в графику также нужно загрузить модули для использованных в ней форматов картинок (чаще всего *png*) и загрузить все имеющиеся в ней шрифты.

Пример загрузки темы оформления, входящей в пакет **grub**:

```
dir=$prefix/themes/starfield # каталог с темой
set theme=$dir/theme.txt     # задаём файл описания
insmod regexp                # этот модуль позволяет использовать шаблоны в
именах файлов
loadfont $dir/*.pf2          # загружаем по шаблону сразу все шрифты из темы
insmod png                   # модуль поддержки картинок
set gfxmode=auto
set lang=ru
insmod gfxterm
insmod vbe
terminal_output gfxterm      # включаем графику
```

Некоторые темы оформления GRUB можно найти в **AUR**.

**Важно:** Некоторые темы оформления могут быть устаревшими, и включенные в них шрифты могут оказаться **несовместимыми с текущей версией GRUB**.
Также в некоторых темах могут использоваться шрифты без поддержки кириллицы. Имейте это в виду, включая русский язык в конфигурации загрузчика

## Автоматизация в меню

### Запоминание выбранного пункта меню

В настройках конфигуратора запоминание включается так:

```
/etc/default/grub

GRUB_DEFAULT="saved"
GRUB_SAVEDEFAULT="true"
```

Пример реализации запоминания в конфиге GRUB2 приведён ниже.

### Однократная загрузка заданного без смены дефолта

Существует утилита **grub-reboot**, с её помощью можно из-под ОС запланировать для однократную загрузку другого пункта меню, например так:

```
grub-reboot "Windows XP"
```

В генерированном меню это работает, если перед запуском `grub-mkconfig` в `/etc/default/grub` была строка

```
GRUB_DEFAULT="saved"
```

Пример реализации однократного выбора в конфиге GRUB2 приведён ниже.

**Пример конфига GRUB с реализацией запоминания**

```
set timeout=5
set default=0
load_env # восстанавливаем переменные из файла

if [ -n "$next_entry" ] ; then   # если задан временный выбор
 set default="$next_entry"       # временно подменяем дефолт
 unset next_entry                # и очищаем временный выбор
 save_env next_entry
fi                               # временный дефолт не сохраняем

export default    # на случай использования savedef внутри submenu

function savedef {     # Создаём функцию по имени savedef
 if [ -n "$1" ]               # выбор берём либо
  then def="$1"               # из первого параметра
  else def="$chosen"          # либо из $chosen, переменной GRUB
 fi                           # с заголовком выбранного пункта меню
 if [ "$def" != "$default" ] ; then   # Если выбор отличается
  set default="$def"                  # от текущего дефолта -
  save_env default                    # сохраняем его
 fi
 unset def
}                    # конец функции

menuentry "Arch Linux" {
 savedef
 linux /boot/vmlinuz-linux root=LABEL=Arch_root rw
 initrd /boot/initramfs-linux.img
}

menuentry "Arch Linux fallback" {
 savedef "Arch Linux" # в следующий раз выберется "Arch Linux"
 linux /boot/vmlinuz-linux root=LABEL=Arch_root rw
```

```
 initrd /boot/initramfs-linux-fallback.img
}

submenu "Windows XP" {
 savedef
 set root=hd0,1
 ntldr /ntldr
}
```

В данном примере используется предоставляемая GRUB2 возможность сохранения переменных между сеансами.

Команда **load_env** загружает из файла все сохранённые в нём переменные.

Команда **save_env** сохраняет указанные переменные в файл.

В переменную **chosen** после выбора пункта меню помещается его заголовок. При выборе пунктов вложенных меню, в переменную записывается весь "путь", состоящий из последовательно выбранных заголовков, разделённых знаком "**>**". Например, если в подменю "Fallback" выбран пункт "Arch linux-lts", в переменной будет `"Fallback>Arch linux-lts"`.

Также здесь используется объявление *функции*. Функция вызывается так же, как другие команды GRUB. При каждом вызове эта функция будет сравнивать свой параметр или заголовок выбранного пункта меню с переменной default, и если выбор отличается - записывать его в default и сохранять.

Переменные сохраняются в файле `/boot/grub/grubenv` . Так как GRUB не умеет корректно модифицировать файловые системы, файл должен быть заранее создан, а длина его при перезаписи не должна меняться, поэтому конец файла до нужной длины в 1024 байта всегда заполнен символами ### . Для просмотра и изменения при загруженной ОС сохранённых в этом файле переменных, настоятельно рекомендуется использовать утилиту **grub-editenv**, либо **grub-reboot** и **grub-set-default**.

### Динамическое меню

Конфиг GRUB2 это скрипт, выполняемый загрузчиком. Он действительно генерирует, а не статически описывает меню загрузчика.

В этом примере загрузчик сам, "на лету", находит в /boot/ все установленные ядра Arch Linux и образы initramfs, правильно группирует их и формирует меню для их загрузки, а после выбора пункта меню запоминает его. При загрузке в параметры ядра подставляется автоматически найденный загрузчиком UUID корневого раздела.

В grub.cfg пишем только ссылку. (причину см. **выше**.)

```
/boot/grub/grub.cfg

. $prefix/main.cfg
```

Пользовательские настройки подключаются из отдельного файла `settings.cfg`

```
/boot/grub/settings.cfg

set menu_color_normal=white/blue
set menu_color_highlight=white/black
set timeout=5
```

```
set default=0

boot=/boot
probe -s UUID -u $root    # автоматически определяем UUID корня
opts="root=UUID=$UUID rw" # и подставляем его в параметры ядра
addimg=$boot/intel-ucode.img # включаем обновление микрокода CPU intel

load_env
```

Центральный файл, здесь реализована функция запоминания выбранного пункта меню, сюда же подключены остальные части конфига:

/boot/grub/main.cfg

```
# Подключаем файл настроек
. $prefix/settings.cfg

# Поддержка однократного выбора
if [ -n "$next_entry" ] ; then
 set default="$next_entry"
 unset next_entry
 save_env next_entry
fi

export default

# Описываем функцию запоминания
function savedef {
 if [ -n "$1" ]
 then def="$1"
 else def="$chosen"
 fi
 if [ "$def" != "$default" ] ; then
  set default="$def"
  save_env default
 fi
 unset def
}

# Подключаем динамическое меню
. $prefix/dynamic.cfg

# Подключаем файл для дополнительных пунктов меню, если он есть
c="$prefix/custom.cfg"
if [ -f "$c" ] ; then
 . "$c"
```

```
fi
```

Скрипт поиска ядер и генерации меню:

```
/boot/grub/dynamic.cfg
```

```
insmod regexp

initrd=initrd
if [ -n "$addimg" -a -f "$addimg" ] ; then
 initrd="$initrd $addimg"
fi

kpref="$boot/vmlinuz-"
ipref="$boot/initramfs-"

kernels=
for kfile in "$kpref"* ; do
 k=
 regexp -s k "$kpref"'(.+)' "$kfile"
 kernels="$kernels $k"
done

for ifile in "$ipref"* ; do

 kname=
 for k in $kernels; do
  if regexp "$k" "$ifile" ; then
   if ! regexp "$kname" "$k" ; then continue; fi
   head="Arch $k"
   if regexp -s s "$k"'-(.+).img' "$ifile"
   then head="$head $s"; fi
   kname="$k"
  fi
 done

 if [ -n "$kname" ] ; then
  menuentry "$head" --source="savedef
linux $kpref$kname $opts
$initrd $ifile"
 fi

done
```

## Консоль GRUB2

Загрузчик GRUB2 выполняет свой файл конфигурации как шелл-подобный скрипт, состоящий из команд. Все те же команды он позволяет выполнять и в интерактивном режиме, с помощью консоли.

## Нормальная консоль

### Запуск консоли

Если вы видите в терминале GRUB2 приглашение вида `grub>` , значит вы попали в его нормальную консоль.

GRUB2 открывает нормальную консоль, если:

- при загрузке не найден файл конфигурации `grub.cfg`;
- файл конфигурации найден и выполнен, но в ходе его выполнения так и не было создано меню, и не была выполнена команда **boot** *(см. ниже)*, либо её выполнение завершилось неудачно;
- в меню загрузчика нажата клавиша "**c**".

### Команды, полезные в консоли GRUB2

Кроме уже описанных в части "**Настройка**", в консоли GRUB2 могут пригодиться:

- Переменная **pager** . Присвоение ей единицы командой `pager=1` поможет вам полностью читать вывод следующих команд, если он не помещается на экран.
- Команда **help** . Без параметров выводит список **всех** поддерживаемых в текущей конфигурации команд загрузчика. При вызове вида `help слово` выводит справку по командам, в которых присутствует заданное слово. Например, команда `help normal` покажет справку по командам `normal` и `normal_exit`.
- Команда **ls** без параметров выводит список дисков и разделов, доступных в данный момент загрузчику. С ключом **-l** показывает подробные сведения о каждом из них – размер, метку, UUID и тип файловой системы. С ключами **-lh** выводит размеры в "человекочитаемом" виде. С параметром в виде диска или раздела выводит сведения только о нём, например `ls (hd0,1)` выведет сведения о первом разделе на нулевом диске.
- Команда **ls** с параметром в виде полного пути к каталогу выводит содержимое каталога. С ключом **-l** показывает подробные ведения о каждом файле и каталоге, с ключами **-lh** выводит размеры и даты в "человекочитаемом" фромате. К примеру, команда `ls -lh /boot` подробно выведет содержимое каталога /boot на текущем *(в переменной root)* разделе, а `ls (hd0,1)/` покажет список файлов и каталогов в корневом каталоге раздела hd0,1.
- Команда **set** без параметров. Выводит список всех переменных со значениями.
- Команда **echo** аналогична такой же команде обычного шелла, и выводит всё, что в ней написано. Может использоваться для вывода переменных, например `echo $cmdpath $prefix $root` покажет значения трёх самых важных переменных загрузчика.
- Команда **cat** аналогична одноимённой команде шелла, и выводит содержимое заданного файла в консоль. Так как перенаправления ввода-вывода GRUB2 не поддерживает, использовать её можно только для просмотра текстовых файлов, например, конфига самого загрузчика, fstab, и т.д.
- Команда **boot** запускает образ ядра, другого загрузчика, или EFI-приложения, загруженный перед этим командами `linux`, `initrd`, `ntldr`, `chainloader` и некоторыми другими. В отличии от меню GRUB, где запуск загруженных образов происходит автоматически после завершения кода пункта меню, в консоли выполнение этой команды обязательно – без неё загруженный образ сам не запустится.

В нормальной консоли GRUB2 поддерживает возврат к предыдущим командам и автодополнение команд, каталогов и файлов по нажатию клавиши TAB, как в консоли Linux.

### Пример загрузки ArchLinux из консоли загрузчика

Если вы успешно установили загрузчик в корневой раздел, но забыли создать конфиг загрузчика, вы увидите меню из дефолтного конфига, но оно не будет работать. Чтобы продолжить загрузку, вы можете войти в консоль, нажав клавишу "**c**", и выполнить команды:

```
probe -s UUID -u $root
linux /boot/vmlinuz-linux rw root=UUID=$UUID
initrd /boot/initramfs-linux.img
boot
```

**Пример загрузки с внешнего диска из консоли**

Во многих версиях BIOS нет специального интерфейса для однократной загрузки с флешки. Для этого можно создать отдельный пункт меню, но на один раз проще обойтись консолью GRUB:

```
root=hd1
chainloader +1
boot
```

**Пример конфига с загрузкой без меню**

```
grub.cfg

set timeout=3

probe -s UUID -u $root
linux /boot/vmlinuz-linux rw root=UUID=$UUID
initrd /boot/initramfs-linux.img

if sleep -vi $timeout
then boot
fi
```

В этом примере GRUB2 загружает с диска образы ядра и initramfs, ждёт 3 секунды, и запускает ядро.

Если же в течении этих секунд пользователь нажмёт ESC, он попадёт в консоль загрузчика. Так как ядро уже загружено, для продолжения загрузки достаточно набрать команду **boot** в консоли.

## Аварийная консоль

Если вместо меню или "шапки" нормальной консоли вы видите при старте загрузчика сообщение об ошибке и приглашение вида `grub rescue>`, значит вы попали в аварийную консоль.

Аварийная консоль GRUB2 встроена в стартовый образ загрузчика, и запускается в случаях, когда GRUB2 не может самостоятельно перейти в "нормальный" режим. Такое может случиться, если ядро GRUB2 при загрузке не нашло каталог со своими файлами и модулями по пути, указанному в переменной **prefix**.

Значение этой переменной обычно имеет вид `(диск,раздел)/путь`, например `(hd0,msdos6)/boot/grub`. Обозначение диска подставляется при старте загрузчика, а остальная часть *(начиная с запятой)* "зашивается" в стартовый образ ещё на этапе установки. Это значит, что после изменения любого из содержащихся в `$prefix` параметров *(таблицы разделов, номера раздела, пути к файлам grub)*, загрузчик требуется **переустанавливать**, в противном случае он "вывалится" в аварийную консоль.

В режиме аварийной консоли GRUB2 понимает всего 4 команды: **set** , **unset** , **ls** , и **insmod** . Повтор и автодополнение не поддерживаются, команда **ls** поддерживается в урезанном виде – без ключей и с выводом в сведениях о разделах только типа файловой системы, если она опознана. По-умолчанию в стартовый образ загрузчика включается модуль для поддержки таблицы разделов и файловой системы только для того раздела, на который устанавливается GRUB2. Остальные модули должны загружаться уже из файлов, если загрузчику удастся их найти.

В некоторых случаях, когда встроенных в стартовый образ модулей достаточно для продолжения загрузки в изменившихся условиях, загрузку можно продолжить, изменив в аварийной консоли переменную `$prefix`.

Если вы попали в аварийную консоль, наберите для начала команду `set`. Вы увидите значения трёх главных переменных GRUB2. В переменной **cmdpath** будет обозначение диска, либо полный путь к EFI-файлу, из которого стартовал образ загрузчика. В переменной **prefix** будет тот путь, по которому должен был быть каталог с остальными файлами загрузчика. В переменной **root** будет текущий диск или раздел, скорей всего совпадающий с тем, что в `$prefix`. Попробуйте команды

```
ls
ls $root
ls $prefix
```

Если вы знаете, что делалось с диском перед тем, как GRUB показал ошибку, то возможно, вы уже догадываетесь, в чём ошибка и удастся ли её исправить.

Если в переменной `$root` оказался не тот или несуществующий раздел, можно проверять командами вида `ls (диск,раздел)/` каждый раздел из тех, что вывела первая команда, пока не найдётся нужный. Если он нашелся – к примеру, оказался hd0,msdos5 вместо hd0,msdos6, и читается – запишите его в переменные:

```
root=hd0,5
prefix=($root)/boot/grub
```

Если раздел правильный, а неправильный каталог *(команда `ls $prefix` выдаёт ошибку или не то, что нужно)*, запишите в переменную правильный путь, допустим такой:

```
prefix=($root)/grub
```

Если у вас получилось найти правильный диск, раздел и каталог, и вы успешно прописали из в переменные, остаётся загрузить модуль "normal" и выполнить одноимённую команду *(она станет доступна после загрузки модуля)*, чтобы перейти в "нормальный" режим загрузчика:

```
insmod normal
normal
```

Если же из аварийной консоли не удаётся получить доступ к нужному каталогу с файлами загрузчика – увы, но больше она ничем не сможет помочь, и вам придётся обратиться к другим способам загрузки, например с внешних носителей.

## Запуск GRUB2 из других загрузчиков

- Загрузочный EFI-образ GRUB2 в режиме **UEFI** может быть запущен так же, как любое EFI-приложение.
- Загрузочный образ BIOS-сборки GRUB2 новых версий может быть запущен по стандарту Multiboot из других загрузчиков. См. также главу "**Генерация загрузочного образа для BIOS без установки**".

### Загрузка из старых версий GRUB

Код конфига для **GRUB Legacy**, **NeoGRUB** *(и возможно, **grub4dos**)*, с загрузкой GRUB2:

```
menu.lst

default 0
timeout 1

title       Chainload into GRUB v2
root        (hd0,7)
kernel      /boot/grub/i386-pc/core.img
```

### Загрузка из syslinux

**Пример загрузки GRUB2** из syslinux приведён в **статье о нём**.

## Примеры исправления проблем

### Сообщение о невозможности встраивания в MBR

```
grub-setup: warn: This msdos-style partition label has no post-MBR gap;
embedding won't be possible!
grub-setup: warn: Embedding is not possible. GRUB can only be installed in
this setup by using blocklists.
            However, blocklists are UNRELIABLE and its use is discouraged.
grub-setup: error: If you really want blocklists, use --force.
```

Эта ошибка может возникнуть, когда вы попытаетесь установить в виртуальную машину VMware. Читайте больше об этом **здесь**.

Это также может случится, если первый раздел начинается сразу после **MBR**, без необходимого места в 60 блоков перед первым разделом.