

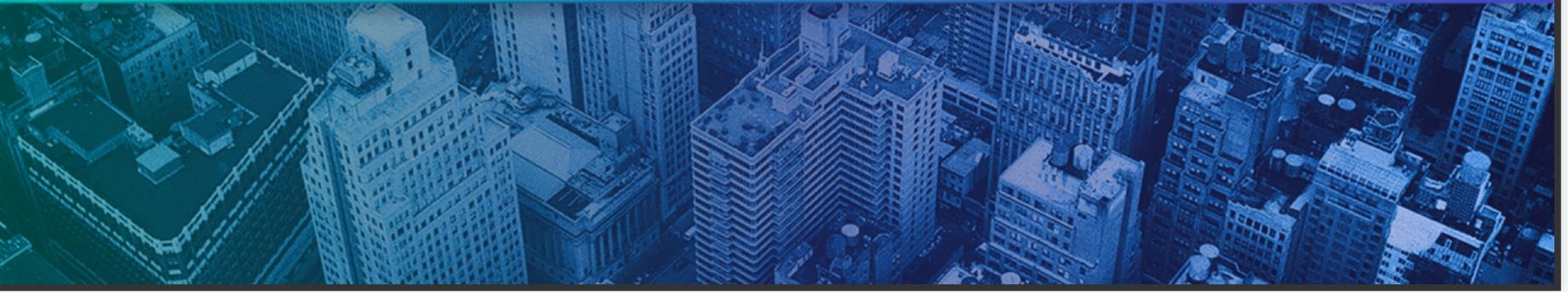


Онлайн-образование



Меня хорошо видно && слышно?

Ставьте  , если все хорошо
Напишите в чат, если есть проблемы



НЕ ЗАБЫТЬ ВКЛЮЧИТЬ
ЗАПИСЬ!!!

Загрузка ОС Linux

Цели и задачи

- Рассмотреть процесс загрузки
- Понять основные различия между загрузкой BIOS/MBS vs UEFI/GPT
- Параметры загрузки ядра
- Работа с загрузчиком GRUB, изменение параметров загрузки
- Разобрать роль initrd
- Рассмотреть пример добавления модуля initrd

Шаги процесса загрузки

- Включение питания
 - Загрузка BIOS/UEFI из NVRAM
 - Тест “железа”
 - Выбор загрузочного устройства (диск, сеть)
- Запуск загрузчика
 - GRUB
 - Выбор в какое ядро грузиться
- Загрузка ядра
 - Загрузка initrd
 - Создание структуры данных ядра
 - Старт PID 1 (init/systemd)

BIOS/MBR vs UEFI/GPT

Basic Input-Output System (BIOS)

- Остался только в виртуалках (SeaBIOS)
- Требует полуторный загрузчик из-за недостатка флеш памяти. BIOS должен работать в 16-битном режиме процессора и ему доступен всего 1 Мб памяти.
- Проблемы с одновременной инициализацией нескольких устройств

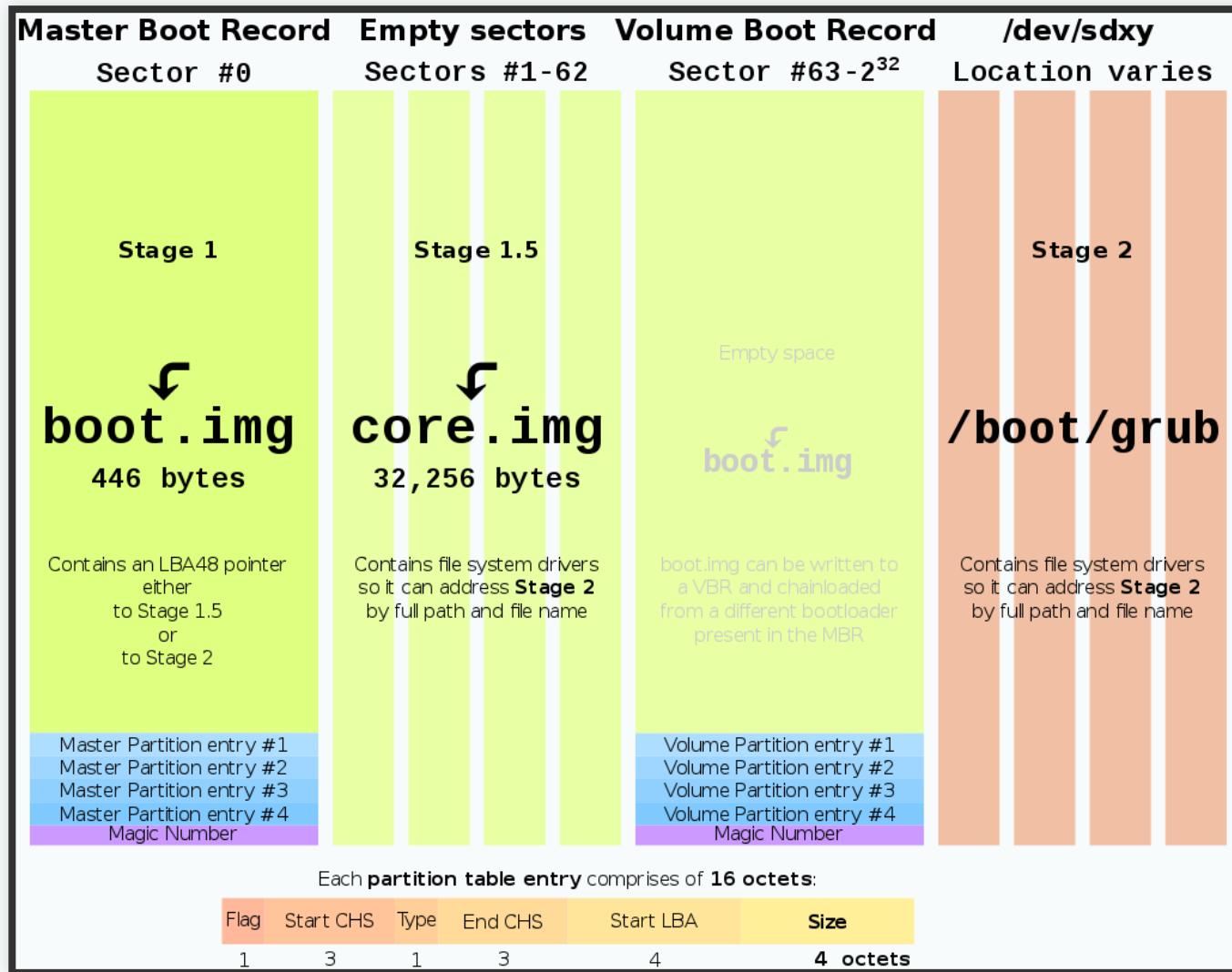
Загрузка из MBR

- BIOS Загружает запись из MBR в NVRAM и приступает к ее исполнению
 - 1-st stage
 - 1.5 stage
 - 2-st stage
- Ограничения
 - первый этап вмещается в 446 байт (1 сектор на диске). Малый объем - простая логика
- Задача
 - обнаружить и загрузить этап 1.5, который располагается в пространстве между загрузочной записью и первым разделом на диске.

Загрузчик GRUB

- Разработан проектом GNU
- GRUB Legacy и GRUB2 (текущий)
- GRUB является эталонной реализацией загрузчика, соответствующего спецификации Multiboot
- GRUB умеет передавать управление другому загрузчику - т.н. multichain booting
- GRUB позволяет пользователю при загрузке задавать произвольные параметры и передавать их в ядро Multiboot-совместимой ОС для дальнейшей обработки. (пример чуть позже)

Расположение на диске



Файлы в /boot

- **/boot** - основной каталог. Отдельный затем, чтобы обеспечить работу с, например, рейдами. Сам по себе "полуторный" загрузчик не понимает софтверных рейдов
- **/boot/initramfs-*** - образ initrd. При обновлении ядра каждый раз собирается новый.
- **/boot/vmlinuz-*** - ядро Linux
- **/boot/System.map-*** - это ссылки на все экспортанные функции ядра. Нужен для утилиты insmod чтобы правильно скомпоновать ядро.
Генерируется при сборке ядра

Демо загрузка в режиме BIOS/MBR

UEFI преимущества

Придумали в IBM для своей же архитектуры Power в конце 80-х. По сути эта та же маленькая ОС со своей спецификацией

Улучшения по сравнению с BIOS:

- Стартует в защищенном режиме
- Знает что такое ФС сразу и знает про GPT. Умеет грузиться с дисков > 2ТВ
- Имеет модульную архитектуру - можно использовать свои приложения и загружать свои драйвера
- Встроенный менеджер загрузки

EFI system partition

Что нужно для загрузки UEFI:

- Таблица разделов GPT
- Система загрузки ищет специальный раздел ESP (EFI system partition)
- Спецификация UEFI определяет имена и расположение файлов на ESP
 - например \efi\boot\boot[название архитектуры].efi в FAT

Найдем раздел и название файлов

```
fdisk /dev/sda
sfdisk /dev/sda --part-type 1
```

UEFI Secure Boot

- технология защищает от выполнения неподписанного кода не только на этапе загрузки, но и на этапе выполнения ОС, например, как в Windows, так и в Linux проверяются подписи драйверов/модулей ядра, таким образом, вредоносный код в режиме ядра выполнить будет нельзя. Но это справедливо только, если нет физического доступа к компьютеру, т.к., в большинстве случаев, при физическом доступе ключи можно заменить на свои.
- Secure Boot призван защитить от буткитов, от атак типа Evil Maid.

UEFI Secure Boot и Linux

- Кто виноват?
 - Windows 8 certification program - настрольные компьютеры должны поставляться с включенным Secure Boot. Поэтому на нужно включать ключи MS. Следовательно, только boot loaders - подписанные MS будут работать с такими компьютерами.
- Что делать?
 1. Отключить secure boot
 2. Использовать подписанный MS загрузчик
 3. Использовать свои ключи

Демо загрузка в режиме UEFI/GPT

Параметры ядра

- Параметры, обрабатываемые самим ядром
 - Список и описание параметров ядра
 - <https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt>
 - `man 7 bootparam`
 - Пример: `root= nomodeset ipv6.disable=1`
- Параметры, обрабатываемые системой инициализации
 - <https://www.freedesktop.org/software/systemd/man/kernel-command-line.html>
 - Пример: `quiet rhgb single`
- Параметры загрузки доступны через файл `/proc/cmdline`

Параметры ядра. Частный случай

Восстановление пароля

- https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/terminal_menu_editing_during_boot#sec-Changing_and_Resetting_the_Root_Password

параметр **rd.break** запускает шелл перед pivot_root().

Кореневая файловая система будет находиться в /sysroot

Параметры ядра. Частный случай

Восстановление пароля, способ 2:

```
добавляем параметр init=/sysroot/bin/sh
```

Параметры ядра. Частный случай

SystemD Single User Mode:

```
systemd.unit=emergency.target  
systemd.unit=rescue.target
```

Демо передаем
параметры ядру во
время загрузки

Запишем параметры
ядра в конфигурацию
загрузчика

Файлы конфигурации в /boot/grub2

- **/boot/grub2/** - основной каталог загрузчика
- **/boot/grub2/device.map** - список дисков/разделов в читаемом для GRUB-а формате
- **/boot/grub2/grub.cfg** - скрипт для подгрузки модулей, рисования менюшки пользователю. Файл генерируется автоматом. Править можно, но следует помнить что при обновлении ядра - будет перезаписан.
- **/boot/grub2/grubenv** - файл с небольшим кол-вом сохраненных состояний - переменными окружения. Во время загрузки в нее сохраняются эти переменные, а из работающей системы их можно использовать для редактирования окружения grub

Изменяем параметры загрузки

Шаблоны для конфигурационных файлов grub

- **/boot/grub2/i386-pc/** - директория с разнообразными модулями
- **/etc/grub2.cfg** - симлинка в /boot/grub2/grub.cfg
- **/etc/grub.d/** - скрипты для формирования конфига GRUB-а
- **/etc/default/grub** - переменные для формирования grub.cfg. Те переменные что будут добавляться к параметрам ядра при генерации конфига, например, при обновлении самого ядра.
- **/etc/grub.d/40_custom** - с помощью этого скрипта можно создавать свои пункты меню с запуском кастомных ядер

Скрипты из grub.d

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries. Simply
# menu entries you want to add after this comment. Be careful not
# to the 'exec tail' line above.
menuentry 'OTUS Kernel' {
set root='(hd1,msdos1)'
linux /otus_kernel root=/dev/sdb1 ro quiet
initrd /initrd_otus_kernel.img
}
```

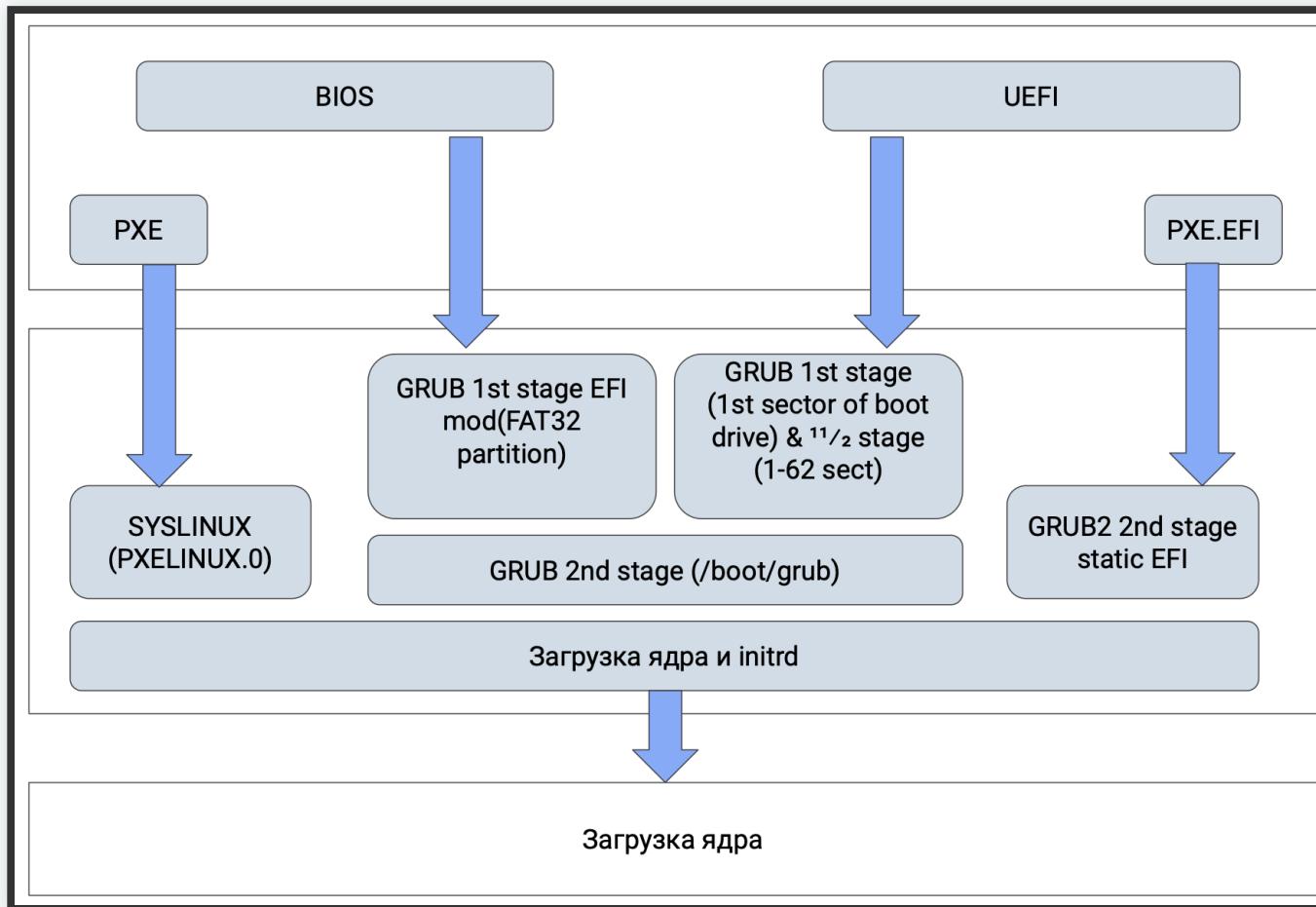
GRUB2 конфигурация

/etc/default/grub Часто модифицируемые опции:

- GRUB_CMDLINE_LINUX - параметры ядра
- GRUB_BACKGROUND - фоновая картинка
- GRUB_DEFAULT - пункт меню “по умолчанию”
- GRUB_DISABLE_RECOVERY - убирает автогенерацию рековери пунктов
- GRUB_PRELOAD_MODULES - перечень GRUB модулей
- GRUB_TIMEOUT - время в секундах до автovыбора

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

Общий принцип



Initrd

Initrd - (`initramfs`, `initramdisk`) временная файловая система, используемая при начальной загрузке для монтирования корневой файловой системы для которой в свою очередь необходим модуль для работы с диком и ФС, а для чтения модуля необходима файловая система, с которой этот модуль читается.

Проблема курицы и яйца в полный рост)

Initrd. Общий принцип

Маленький образ ОС, единственная задача которого

- обработать модули (собрать LVM, собрать RAID),
- перемонтировать rootfs
- передать управление ядру:

Шаги в initrd

- инициализация ядра
- запуск /sbin/init
- загрузка модулей и некоторые этапы инициализации
- монтирование корня mount / pivot_root()

root fs:

- монтирование остальных разделов
- инициализация сети и запуск сервисов

Смотрим что внутри Initrd

Разборка:

```
zcat /boot/initrd-$(uname -r).img | cpio -i
```

Сборка:

```
find. -print0 |  
cpio -o --null --format=newc |  
gzip -q -9 > /boot/initrd-$(uname -r).img
```

Initrd. dracut

- <https://www.kernel.org/pub/linux/utils/boot/dracut/dracut.html>
- В CentOS для управления initrd используется dracut, который позволяет
 - модифицировать
 - просматривать содержимое initrd
 - вставлять свои скрипты в разные этапы загрузки.

Части модуля (функции-хуки)

исполняются в определенные этапы загрузки:

- cmdline - самое начало загрузки initrd
- pre-udev - перед запуском udev-подсистемы
- pre-trigger - в процессе запуска udev'а, возможность с ним взаимодействовать
- pre-mount - перед монтированием файловых систем
mount - смонтировать root-filesystem
- pre-pivot - после монтирования перед pivot_rootcleanup -
перед pivot_root для “подчистки за собой”

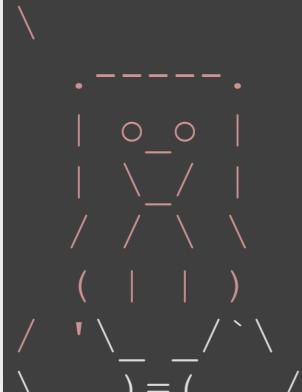
Пример модуля

```
#!/bin/bash
check() {
return 0
}
depends() {
return 0
}
install() {
inst_hook cleanup 00 "${moddir}/test.sh"
}
```

Пример модуля

```
#!/bin/bash
exec 0<>/dev/console 1<>/dev/console
2<>/dev/console
cat <<'msgend'
Hello! You are in dracut module!
```

```
< I'm dracut module >
```



Рефлексия



Отметьте 3 пункта, которые вам запомнились с вебинара



Что вы будете применять в работе из сегодняшнего вебинара?



Заполните, пожалуйста,
опрос о занятии по ссылке в чате

Домашнее задание

1. Попасть в систему без пароля несколькими способами
2. Установить систему с LVM, после чего переименовать VG
3. Добавить модуль в initrd
4. (*) Сконфигурировать систему без отдельного раздела с /boot, а только с LVM Репозиторий с пропатченным grub:
https://yum.rumyantsev.com/centos/7/x86_64/PV
необходимо инициализировать с параметром --bootloaderaresize 1m