Загрузка ОС и EFI

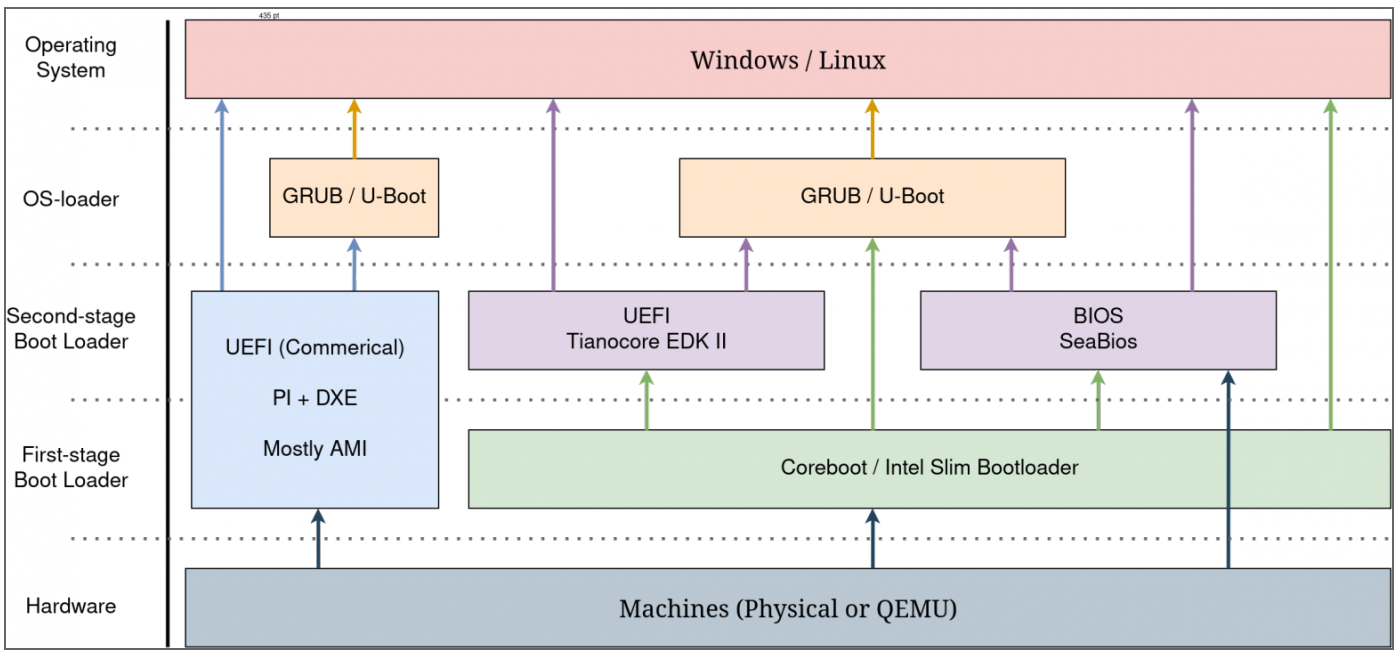
Первая лекция курса «Практическое использование Linux»

Задача загрузки | Legacy BIOS | MBR | UEFI | GPT | ESP | GRUB

> Цель лекции: познакомиться с тем, как работает загрузка ОС, что такое UEFI и в чём его отличие от BIOS.

Загрузка ОС как задача

- Начальная инициализация аппаратуры
- Запуск кода из энергонезависимой памяти на плате
- Поиск доступных вариантов загрузки
- Выбор источника загрузки
- Загрузка следующего компонента
- Передача параметров запуска



Прошивка платформы и её роль

Свойства

- Находится вне обычной файловой системы ОС
- Доступна сразу после включения
- Готовит базовую среду для запуска ОС

Задачи

- Минимальная инициализация аппаратуры
- Поиск источников загрузки
- Запуск встроенного менеджера загрузки

BIOS

Legacy BIOS

- Историческая модель загрузки IBM
- Прошивка выбирает устройство, читает первый сектор и передаёт ему управление
- Зависимость от архитектуры и исторических соглашений

MBR

- **Структура MBR**
- Первый сектор диска: начальный код + таблица разделов
- Четыре записи фиксированного формата
- Сигнатура корректного загрузочного сектора в конце
- Extended/logical partitions — обход ограничения формата

UEFI — современная модель загрузки

Цели

- Избавиться от жёсткой архитектурной зависимости
- Улучшить масштабируемость legacy-схемы
- Устранить legacy формата «сектор + прыжок по адресу»

Итог

- Формализованная предзагрузочная среда
- Стандартные сервисы и протоколы
- Запуск EFI-приложений как файлов

Предзагрузочная среда UEFI

- Менеджер загрузки
- Переменные NVRAM
- Работа с разделами и файловыми системами
- Device paths
- Стандартный способ запуска EFI-приложений

UEFI Boot Manager и boot variables

Общая идея

- Загрузка задаётся не только содержимым диска
- Часть конфигурации хранится в NVRAM прошивки

Boot variables

- `Boot####` — отдельные варианты загрузки
- `BootOrder` — порядок перебора
- `BootNext` — одноразовый выбор
- `BootCurrent` — текущая запись
- `Timeout` — время ожидания

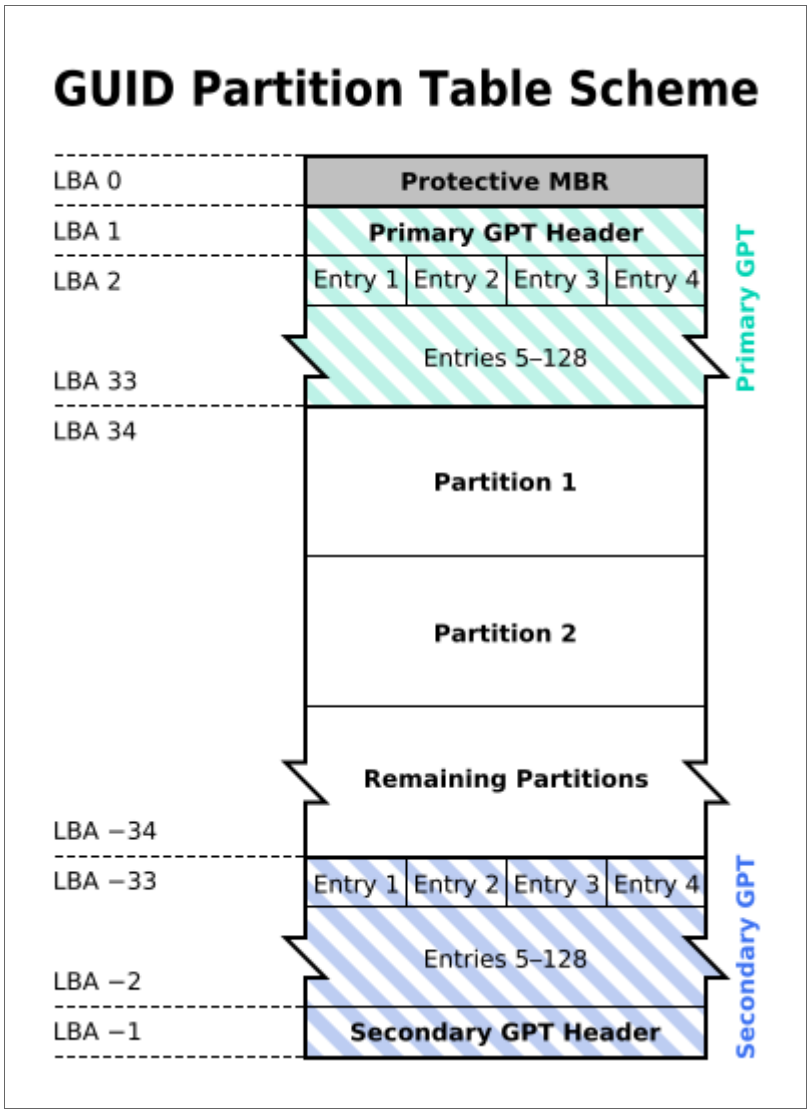
GPT — GUID Partition Table

Проблемы Master Boot Record

- Плохая масштабируемость
- Ограниченное количество записей о разделах
- Слабая защита от повреждений
- Ограничения по размеру диска

Особенности

- GUID как идентификаторы
- Структурированные записи о разделах
- Дополнительная система контроля целостности записей
- Поддержка работы с, вообще говоря, **любыми** объектами



Структура записи GPT

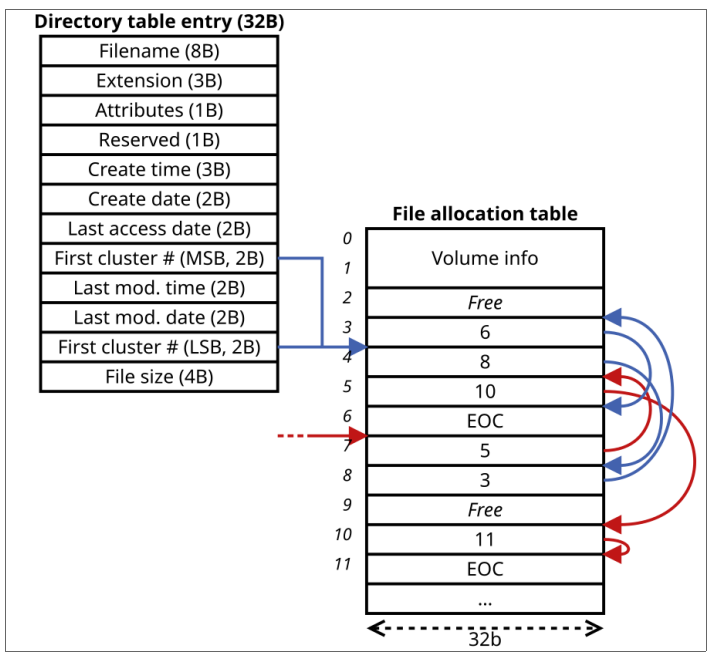
- **Структура GPT**
- Тип раздела
- Уникальный идентификатор раздела
- Начальный адрес
- Конечный адрес
- Атрибуты
- Имя раздела

EFI System Partition

- Специальный раздел, из которого UEFI запускает EFI-образы
- Единая точка размещения файлов загрузки

Структура

- Файловая система FAT (FAT32)
- Каталоги `EFI/...`
- Файлы загрузчиков
  - Ядро Linux с EFI stub



GPT описывает, что раздел **существует**.
ESP — это конкретный раздел с файлами.
Запись `Boot####` указывает, какой файл на нём запускать.

GRUB

Зачем нужен

- Выбрать ядро
- Передать параметры
- Загрузить initramfs
- Поддержать разные сценарии загрузки

Почему удобен

- Работает и с legacy BIOS, и с UEFI
- Удобен для нескольких ОС
- Даёт понятное меню выбора

Конфигурирование

- `/etc/default/grub` — общие параметры
- `/etc/grub.d/` — сценарии генерации
- `grub.cfg` — итоговая сгенерированная конфигурация

Можно ли обойтись без GRUB?

- Да, в UEFI возможен прямой запуск ядра Linux
  - `EFI stub` позволяет ядру выглядеть как EFI-исполняемый образ
  - Ядро размещается на ESP
  - В NVRAM добавляется запись `Boot####`

Плюсы

Цепочка короче, меньше промежуточных звеньев.

Минусы

Меньше гибкости для управления несколькими ядрами и сложными сценариями загрузки.

Speaker notes