Системы хранения и загрузка ОС

Размещение ФС

- Unix:
  - Используя схему размещения (MBR/GPT)
  - Используя весь диск под одну ФС
- Windows:
  - Используя схему размещения (MBR/GPT)

Схемы размещения

- Старое:
  - CHS (Cylinder (0-1023) Head (0-254) Sector (1-63))
  - BIOS
  - MBR, MBR с LBA
- Новое:
  - LBA (Logical block addressing)
  - UEFI (Unified Extensible Firmware Interface)
  - GPT (GUID Partitional Table)

Устаревшее

- CHS: < 8GiB
- MBR: таблица разделов и загрузчик (~400) в одном секторе
- MBR+LBA: начало и длина в 4 байтах (проблема при больше чем 2^32 секторах, при 512 байт на сектор это 2TiB ~ 2.2TB)
- На первой дорожке только MBR, первая партиция начинается (скорее всего) с 63 LBA сектора (в CHS S=1-63, в LBA S=0...)
- Extended BR: запись похожая на MBR, но заполнена только первая и следующая записи партиций
- Идентификатор ФС: 1 байт

UEFI

- GPT:
  - 64 бита на номер сектора (LBA)
  - Таблица размещения: 128 записей (минимум)
  - Начало партиции: 34 (40 для 4k)
  - 0 — Protect MBR
  - 1 — GPT заголовок
  - 2-.. (128B*128 записей) — 32 записи
  - Идентификаторы:
    - Диска: 16 байт (UUID)
    - Партиции: 16 байт тип (GUID), 16 байт идентификатор (GUID)

Загрузка ОС

- legacy:
  - BIOS загружает MBR, передает управление на загрузчик (16 bit, макс 1MiB памяти)
  - Загрузчик находит boot-партицию, загружает первый сектор, передает управление
- UEFI:
  - Найти EFI System partition (ESP)
  - Выбрать загрузчик (файл)
  - Загрузить, передать управление

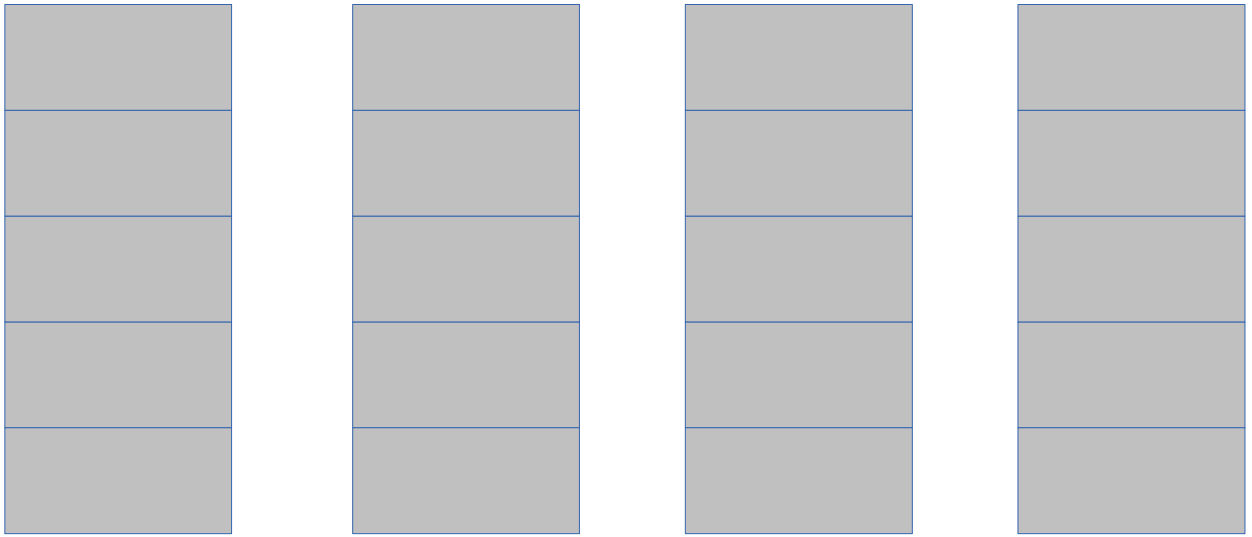
Этапы загрузки (unix-like)

- Примеры:
  - Linux:
    - disk: BIOS → MBR → extlinux/grub/... → vmlinuz+initrd → init →(fork)→ остальные процессы
    - PXE: BIOS → сетевая карта → DHCP (получили «адрес загрузчика») → pxeinux/pxegrub (по TFTP) → vmlinuz (TFTP)+initrd → NFS → init → …
  - FreeBSD:
    - disk: BIOS → MBR → loader (FFS) → kernel → init → ...
    - PXE: … → pxeboot (по TFTP) → kernel (по TFTP) → ...

RAID

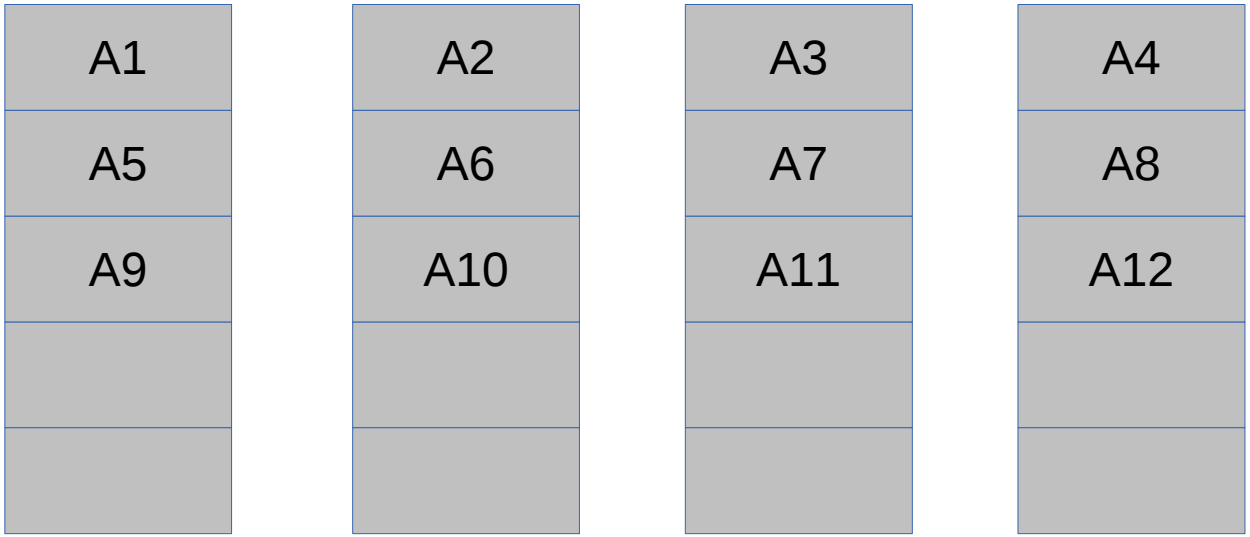
- Redundant Array of Independent Disks
- Можем достичь:
  - Большего объема
  - Большей скорости чтения и/или записи
  - Большей надежности
- В примерах:
  - Мы размещаем данные A1, A2, A3,...
  - В наличии 4 диска
  - Каждый диск подключен независимо от другого

RAID

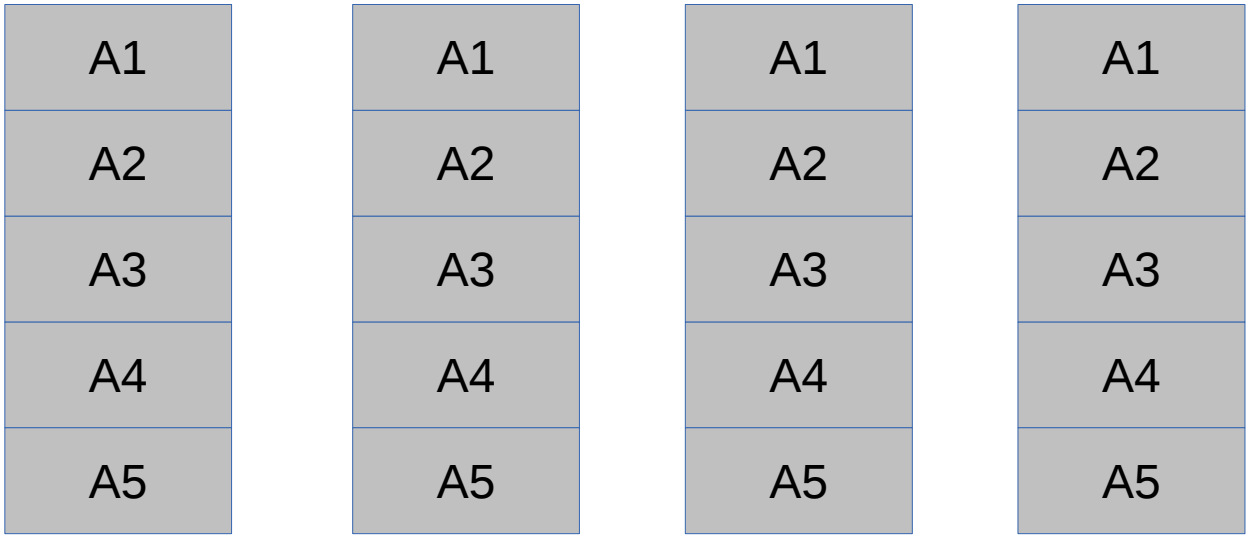


A1, A2,
A3, A4,
A5, A6,
A7, A8, ...

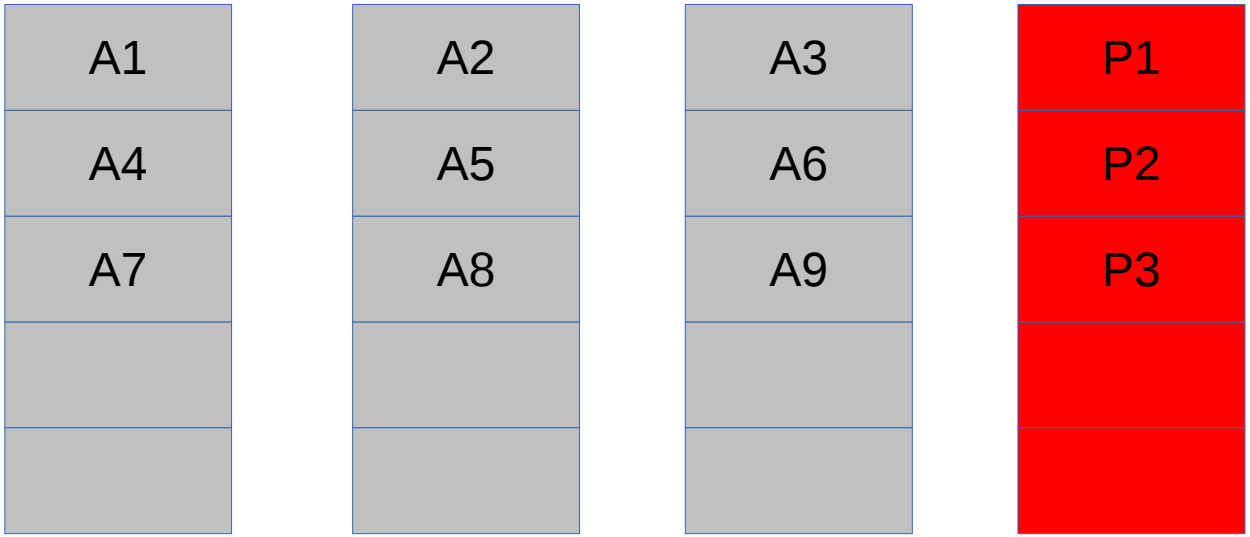
RAID 0



RAID 1

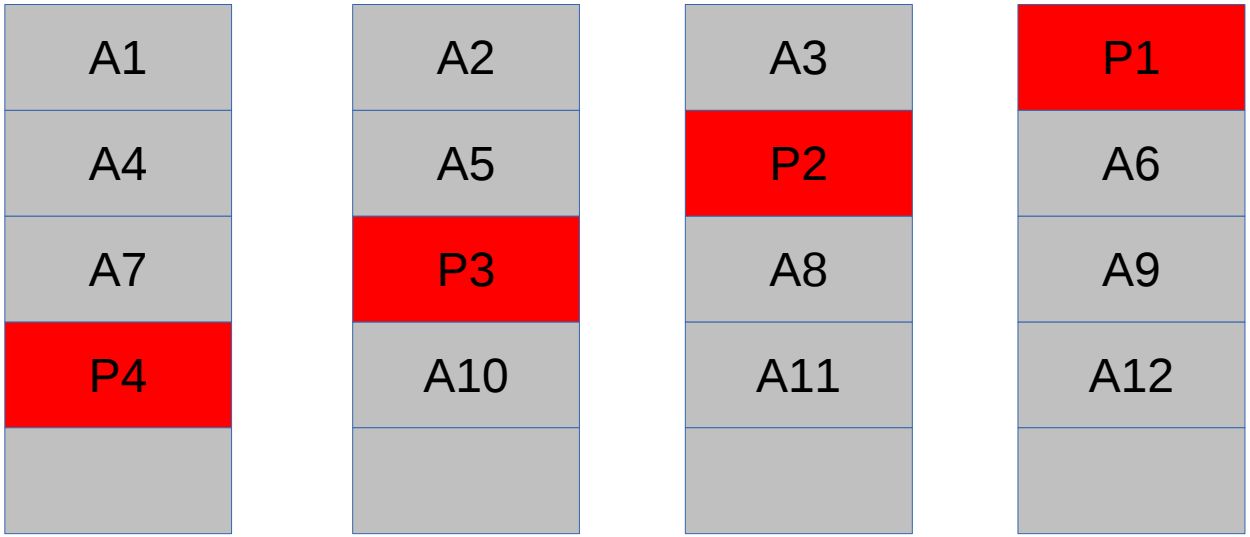


RAID 4



P1=A1+A2+A3
P2=A4+A5+A6
...

RAID 5

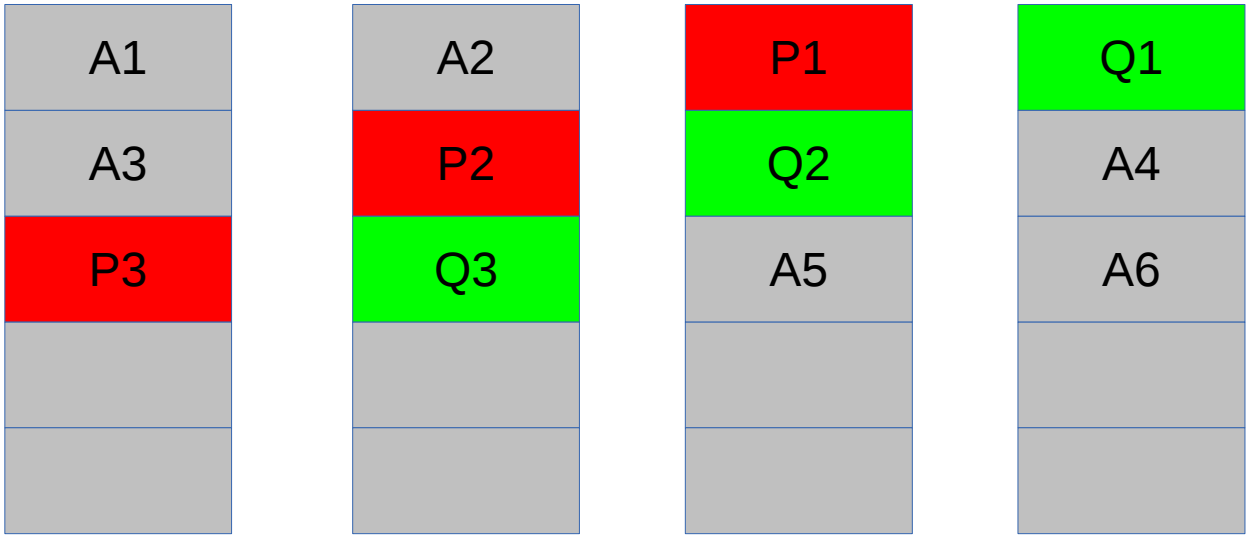


P1=A1+A2+A3
P2=A4+A5+A6
...

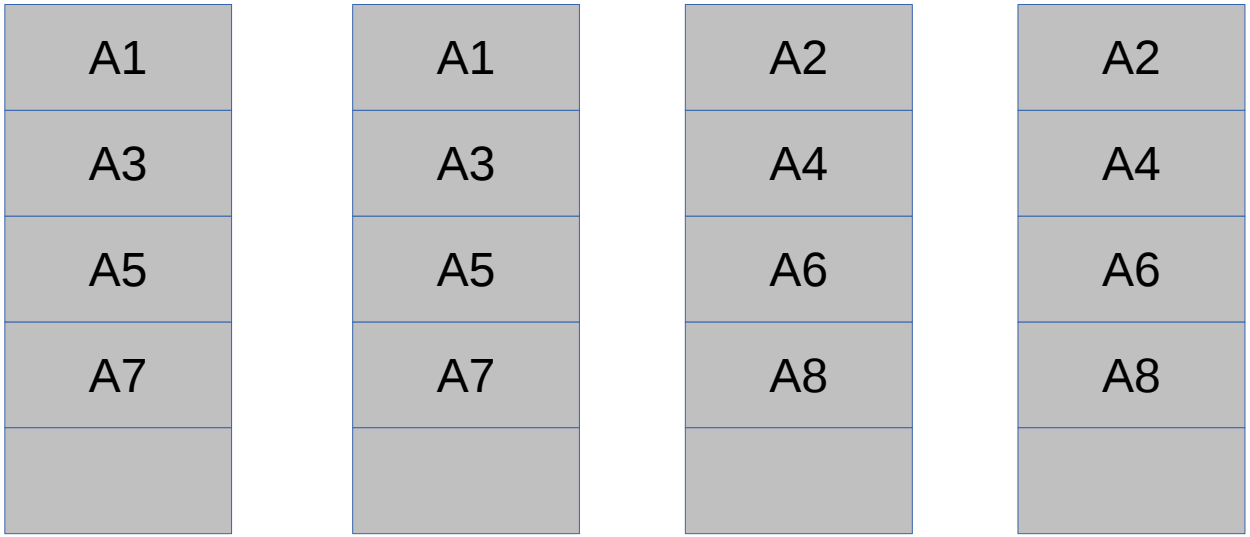
RAID 6

- Два подхода:
  - Две функции:
    - P=P(A1,A2)
    - Q=Q(A1,A2)
  - Два набора данных (двойной набор четности)
    - X1=A1+A2
    - X2=A3+A4
    - Y1=A1+A4
    - Y2=A3

RAID 6



RAID 10/1+0



RAID 2,3

- В отличие от предыдущих работают с байтами/битами.
- RAID2: используется код Хемминга. Для наших 4х надо еще 3 диска (пластины).
- RAID3: аналогичен RAID4

ZFS как RAID

- Типы:
  - pool
  - mirror
  - raidz1-raidz3
- Отсутствие проблемы write hole

Устройства хранения

- Жесткие диски, дискеты, CD-ROM,...:
  - ОС показывает как файлы (особого вида) в /dev
  - Цепочка: устройство ↔ драйвер устройства ↔ (устройства как файлы) ↔ размещение ФС ↔ драйвер ФС ↔ файлы
- Если с устройством работают как с файлом, то и с файлом можно работать как с устройством (только размещаться будет не в /dev)

Зачем файл как устройство

- Снимок ФС (копия, восстановление данных)
- Тестирование драйверов ФС
- Получение «партиции» для записи файла, который больше, чем наши ФС по отдельности