Файловые системы
Архитектура

Объект

- Объект: единица информации (страница, файл, ...)
- Поиск:
  - Имя
  - Содержимое
- Структура:
  - Одноуровневая
  - Иерархическая
- Примеры:
  - Файлы (на текущий момент): имя + иерархия
  - Wiki: структура чаще всего одноуровневая, содержимое

Хранение объекта

- Файловая система:
  - inode (индексный дескриптор)/кластер
    - Размер кратен физическому сектору
    - Особенности 4k-дисков
  - Служебные области
  - Пользовательские области

Потоки файла

- Данные (поток данных)
- Служебные данные (например, размер, владелец, время модификации, …)
- Служебный поток (например, ACL)

Хранение объекта: структура

- Директория:
  - Хранение имен файлов
  - Организация в виде списка/дерева/...
- Поток данных:
  - Основное содержимое (без имени)
  - Организация хранения в виде:
    - Одного «куска» (раздела)
    - Списка
    - Дерева

Устройства хранения и организация хранения

- Ленты:
  - Последовательный доступ
- Жесткие диски (HDD):
  - Произвольный доступ
  - Быстрые операции чтения/записи в пределах одной дорожки
- Flash-диски (SDD):
  - Произвольный быстрый доступ
  - Запись может быть медленнее чтения
  - Ограниченность циклов перезаписи

Примеры организации данных

- Одним «куском»:
  - Лента (tar)
  - CD/DVD (mkisofs)
- Списком (дискеты, жесткие диски):
  - FAT
  - NTFS
- Деревом (HDD, SDD):
  - UFS1-2/EXT2-4: жесткие диски
    BTRFS/ZFS: flash-память

Хранение целым объектом

- Плюсы:
  - За один проход получаем все данные объекта
- Минусы:
  - Изменять файл очень сложно (зажат в своих границах), если на это пойти — большая фрагментация
- Вывод: подходит для архивации
- ISO: служебные данные в конце — можно «эмулировать» перезапись файла/добавление/удаление на RO-дисках

Хранение списком

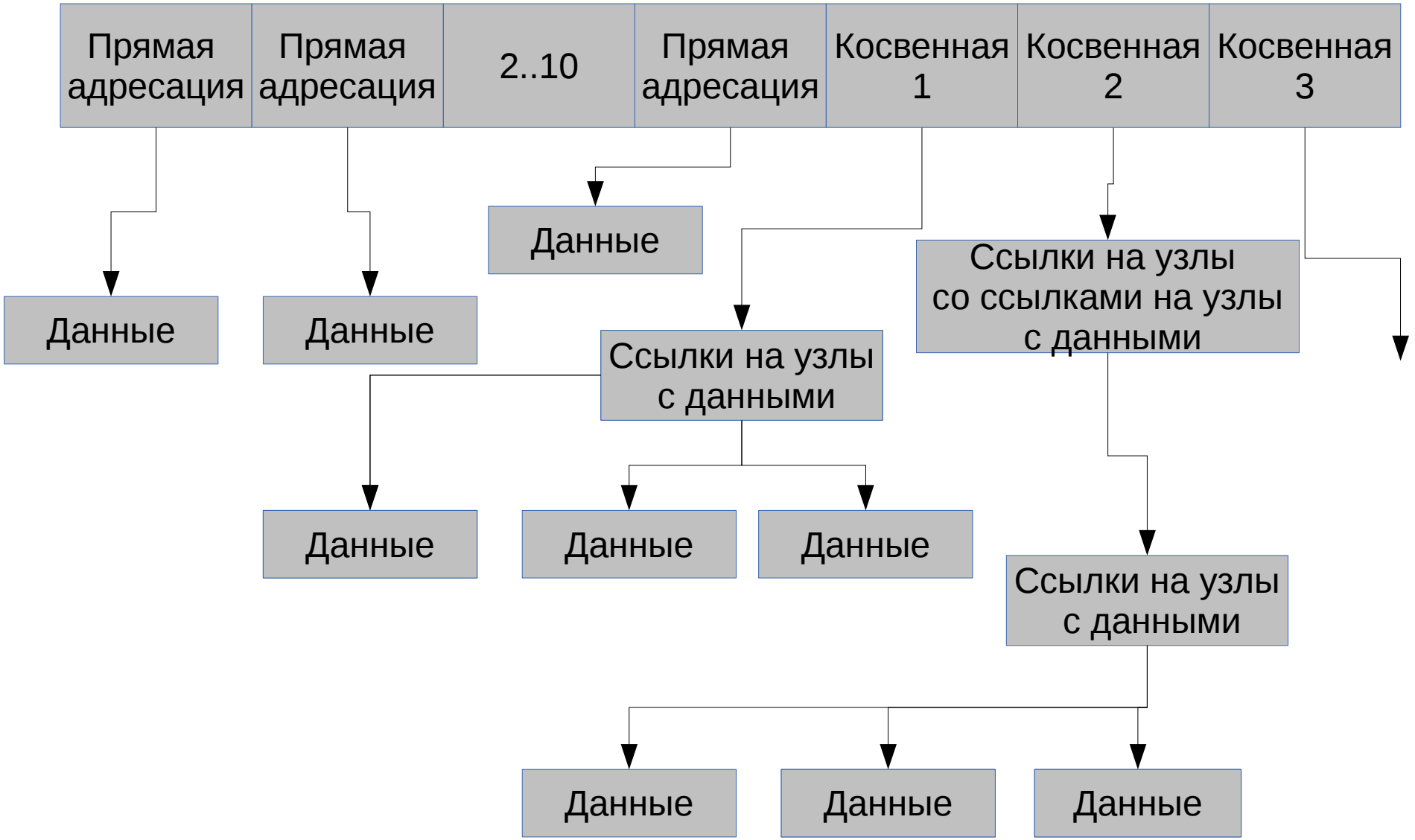
- Файл:
  - Сектор 5
  - Сектор 7
  - Сектор 6
  - Сектор 3
  - Сектор 100
  - ...

Плюсы:
- Файлы могут меняться (расти/уменьшаться)
- Фрагментация данных меньше (если использовали 1Mb, а потом стали 10Kb, то оставшееся место легче использовать)

Минусы (в основном FAT):
- Чтобы дойти до конца, надо знать всю цепочку — страдаем при большой фрагментации
- нельзя создать «дырявые» файлы

NTFS: сжатие цепочек

Хранение деревом (UFS)



Директории и файлы

- FAT/NTFS:
  - Хранение информации о файле в самой директории (файле директории)
- UFS:
  - Директория хранит только имя и inode (для ссылки: имя и ссылку)
  - Информация о файле хранится в самом файле
  - Жесткие/мягкие ссылки (ln)

Особенности ФС из-за носителей

- Жесткие диски:
  - Выгодно размещать данные «рядом», на одной дорожке
  - В FFS: есть цилиндр-группы
- Flash-диски:
  - Перед записью блок надо подготовить (очистить): команда TRIM (ATA)

Скорость/отказоустойчивость

- mount -o sync/async
- Проблема: начали копировать/создавать файл, пропало питание:
  - Образовались кластеры, которые должны быть свободны, но помечены как занятые
  - Аналогично при удалении файла (успели «удалить» из директории, но не освободить место)

Решение

- Журналирование:
  - NTFS
  - Ext3
- Softupdate + фоновый fsck:
  - FFS
- Транзакционые ФС:
  - ZFS/BTRFS

Журналирование

- В NTFS и Ext3 (по умолчанию) применяется частичное журналирование
- Операция записи:
  - Записать в журнал, что хотим писать в кластеры a,b,c
  - Пометить кластеры как занятые (если были свободны)
  - Записать
  - Удалить запись из журнала
- Сохранность данных не гарантируется (был 1Mb 0, выполнили запись 1Mb 1, часть кластеров записалась, часть — нет).

Транзакционные ФС

- Стратегия COW (Copy On Write)
- Пусть у нас есть узлы a,b,c,d и мы делаем запись в b и c
- Операция записи:
  - Находим свободные узлы (b', c') и помечаем в памяти как занятые
  - Пишем в b',c' новые данные
  - Меняем ссылки в файле (одна запись) на a,b',c',d
  - Помечаем b и c как свободные
- Сохранность данных: либо на момент до записи, либо на момент после записи

Следующая лекция

- Размещение ФС на устройстве:
  - MBR
  - GPT
- «Виртуальные» устройства:
  - Устройство = файл, Файл = устройство
  - RAID (объединение нескольких устройств в одно):
    - Увеличить объем
    - Увеличить надежность