Файловые системы

Д.В. Иртегов

Курс "Операционные системы"

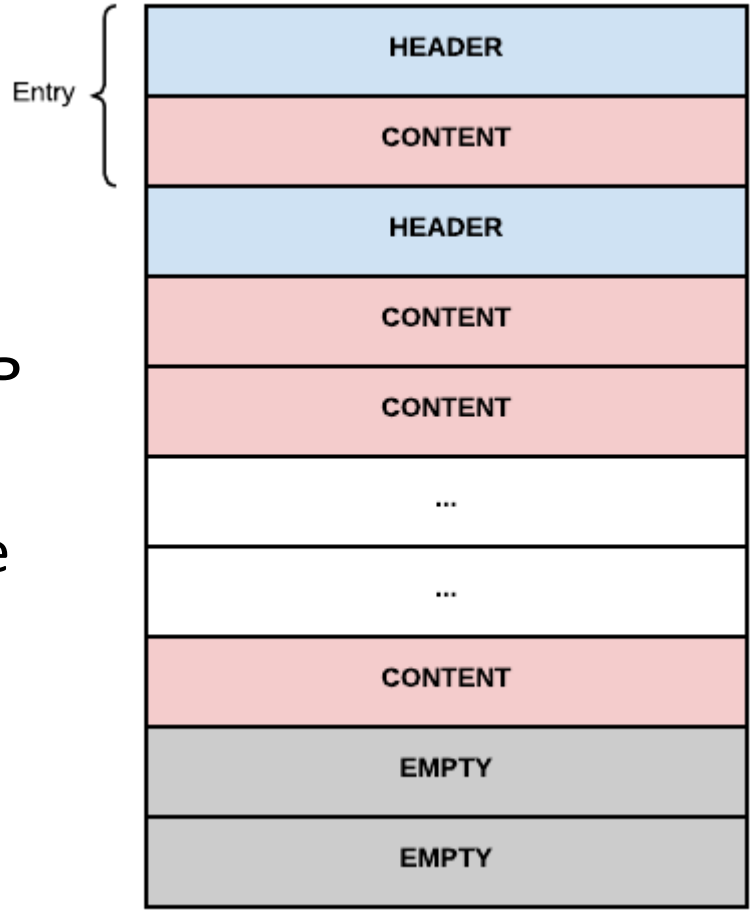
ФИТ/ФФ НГУ

Определение файла

- Совокупность данных, к которой можно обращаться по имени
- В курсе Unix мы видели файлы, которые
  - Не являются совокупностями данных (/dev/null)
  - Не имеют имен (трубы, сокеты)
- Но это определение лучше всего покрывает понятие файла в большинстве существующих ОС
- Обращение по имени подразумевает таблицу имен (каталог)

Простейшие файловые системы: tar

- Tape ARchive
- Формат архивов, первоначально разработанный в Unix
- Рассчитан на максимальную переносимость
- Каждый файл начинается с заголовка
  - Заголовок содержит имя файла, длину и другие атрибуты
  - Тело (содержимое) файла идет за заголовком
  - Потом идет заголовок следующего файла

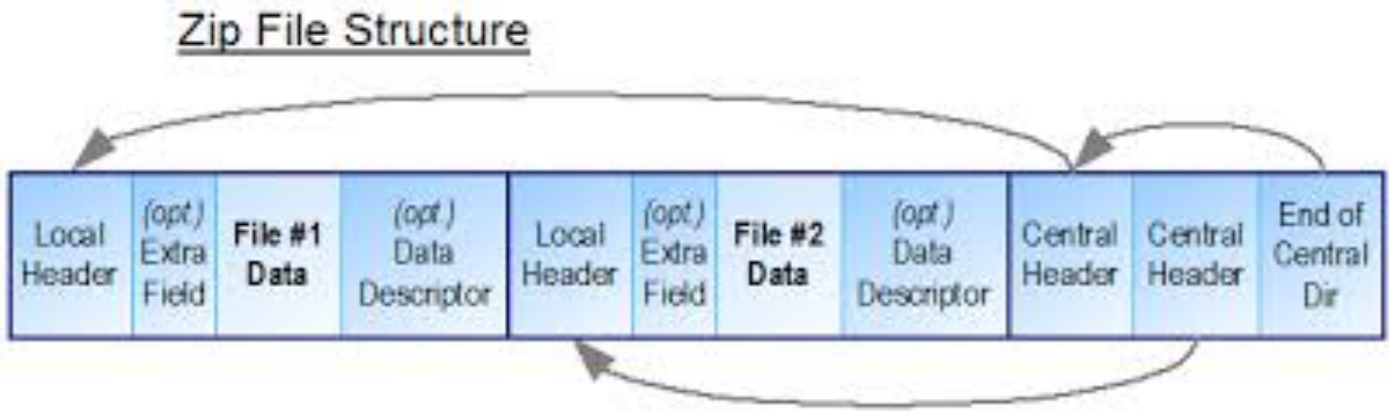


TAR

- Центрального каталога нету
- Каталогов вообще нету. В заголовках лежат иерархические имена с символами /
- Для поиска файла по имени, надо прочитать первый заголовок, промотать до конца содержимого, прочитать второй заголовок и т.д.
- Нельзя стирать файлы (для этого надо пересоздать весь архив)
- Но можно положить файл с тем же именем в конец (рудиментарные версии файлов)

Усовершенствования этой идеи

- Zip: добавлен центральный каталог
- Имена по-прежнему хранятся иерархические
- Заголовки перед файлами сохраняются
- Аналогичную структуру имеют другие архивные файлы: arj, 7zip



RT-11

- Центральный каталог
- Имена фиксированной длины без иерархии (5+3)
- Есть понятие стертого файла (пустого пространства)

Имя файла 1	**Длина файла 1**	**Др. аттрибуты**
Имя файла 2	Длина файла 2	Др. аттрибуты
дырка	Длина дырки	Не используется
Имя файла 3	Длина файла 3	Др. аттрибуты

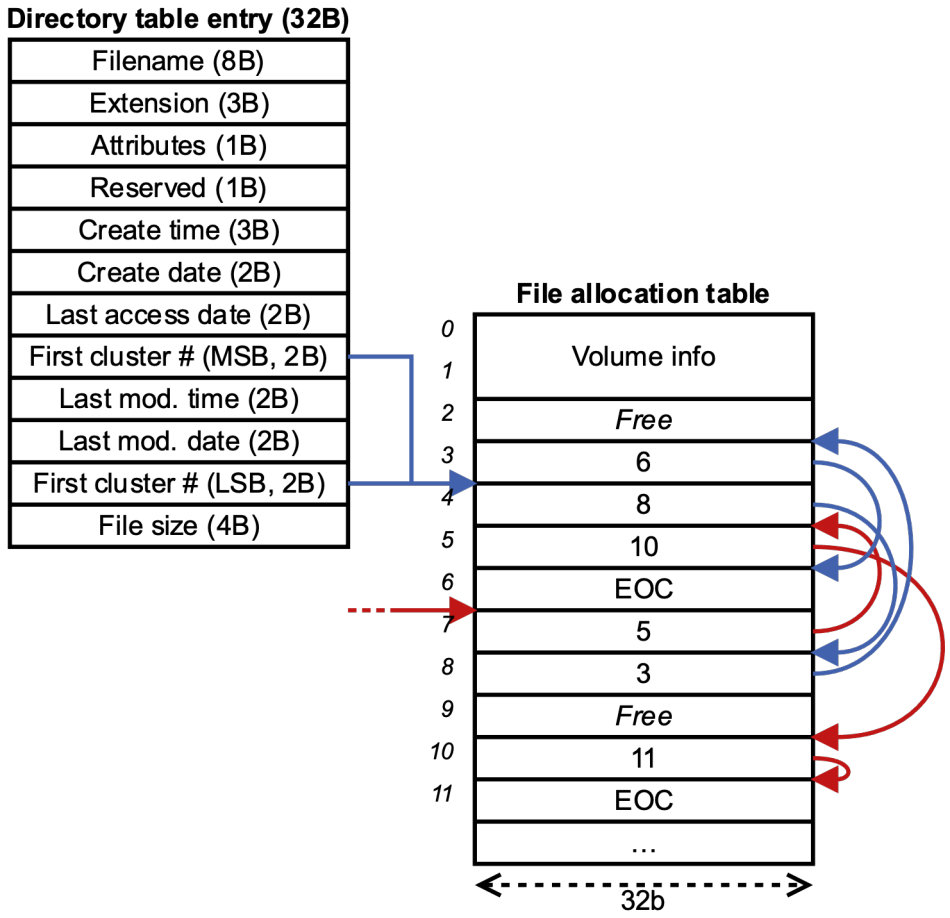
ISO-9660 (CDFS)

- Файлы занимают непрерывные куски диска
- Каталогов много, поэтому начало файла не может вычисляться как сумма длин предыдущих файлов
- В каталоге, кроме длины и атрибутов, надо хранить ссылку на первый блок файла
- Непрерывные файлы нельзя увеличивать по запросу, как в Unix
- Единственный способ удлинить файл на ISOFS и RT-11 - это стереть/забыть старый файл и создать новый

Как сделать файлы из несмежных кусков диска?

- Сделать из файла связный список?
  - Допустимо при последовательном доступе к файлу
  - Неудобно при произвольном доступе
  - Для поиска любого сектора надо прочитать все предыдущие
- Что если вынести указатели связного списка в отдельную таблицу?
- Создаем таблицу, в которой одна запись соответствует одному блоку диска.
- Если блок принадлежит файлу, запись таблицы - следующий блок этого файла

FAT (File Allocation Table)



Недостатки FAT

- Все метаданные файла (кроме раскладки по диску) хранятся в каталоге
- При поиске файла, мы должны читать метаданные других файлов
- Невыгодно увеличивать объем метаданных
  - Права доступа
  - Расширенные атрибуты

Современные файловые системы

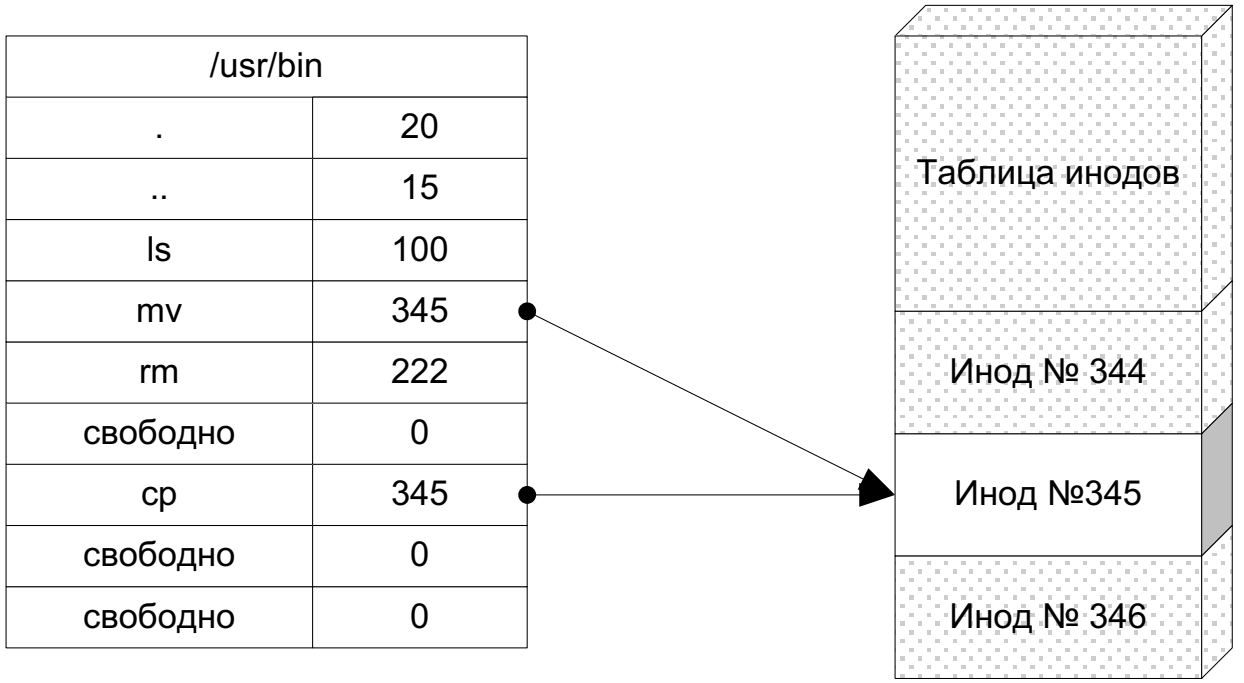
- FAT12/16/32 – разработана лично Биллом Гейтсом для MS DOS в 1982 году
  - В той или иной степени поддерживается всеми современными ОС и рядом «некомпьютерных» устройств (фотоаппараты, телевизоры…)
  - Широко используется на удаляемых носителях (USB-флэшки, SD)
  - Вариант: ExFAT оптимизированный для флэш-накопителей
- NTFS – основная ФС Windows NT; разработана в 1989-1991 годах
- UFS (FFS) – разработана для BSD Unix, также поддерживается в Solaris
- Ext2/ext3 – основная ФС в Linux с 1991 по ~2010 годы
- Ext4 – основная ФС в современных Linux-системах

Современные ФС (продолжение)

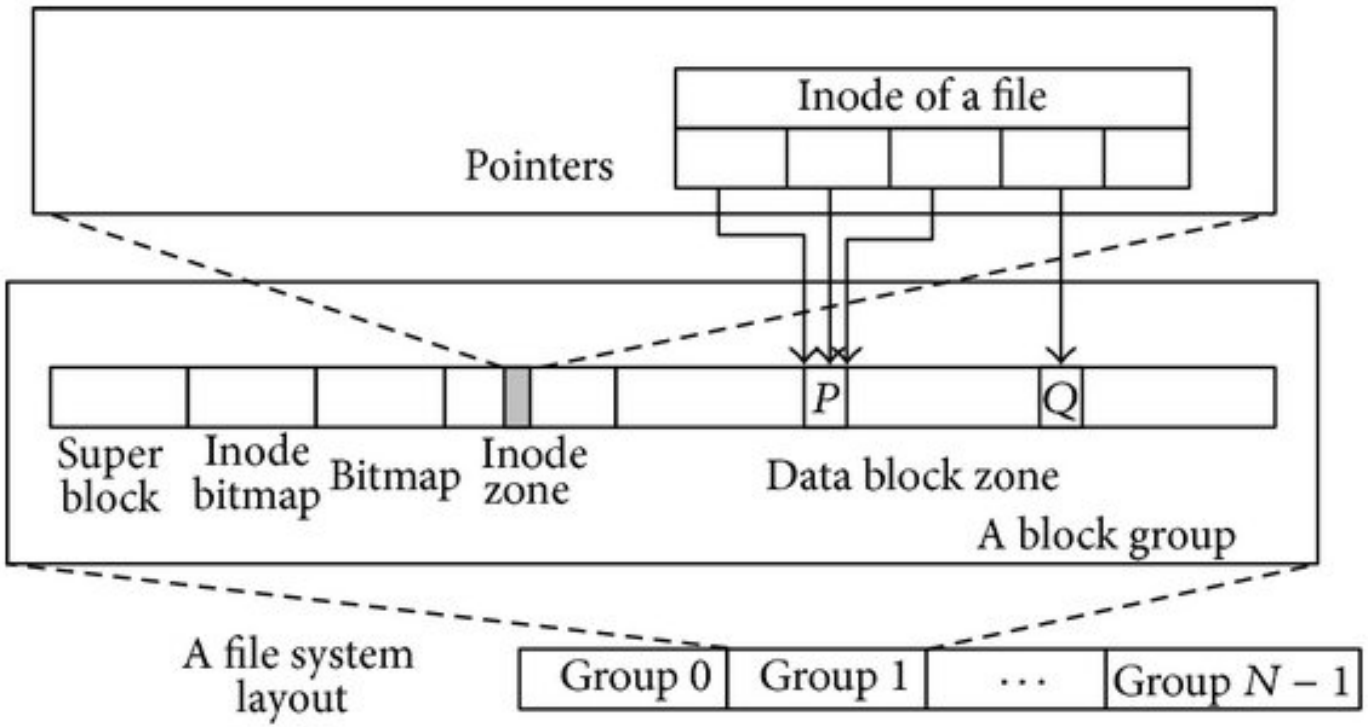
- ReFS – новая ФС, появившаяся в Windows Server 2012, довольно мало открытой информации
- Файловые системы с копированием при записи:
  - NetAPP WAFL, используется в СХД NetAPP
  - Sun/Oracle ZFS – Solaris, есть поддержка в Oracle Linux
  - Linux btr

Структура типичной современной ФС

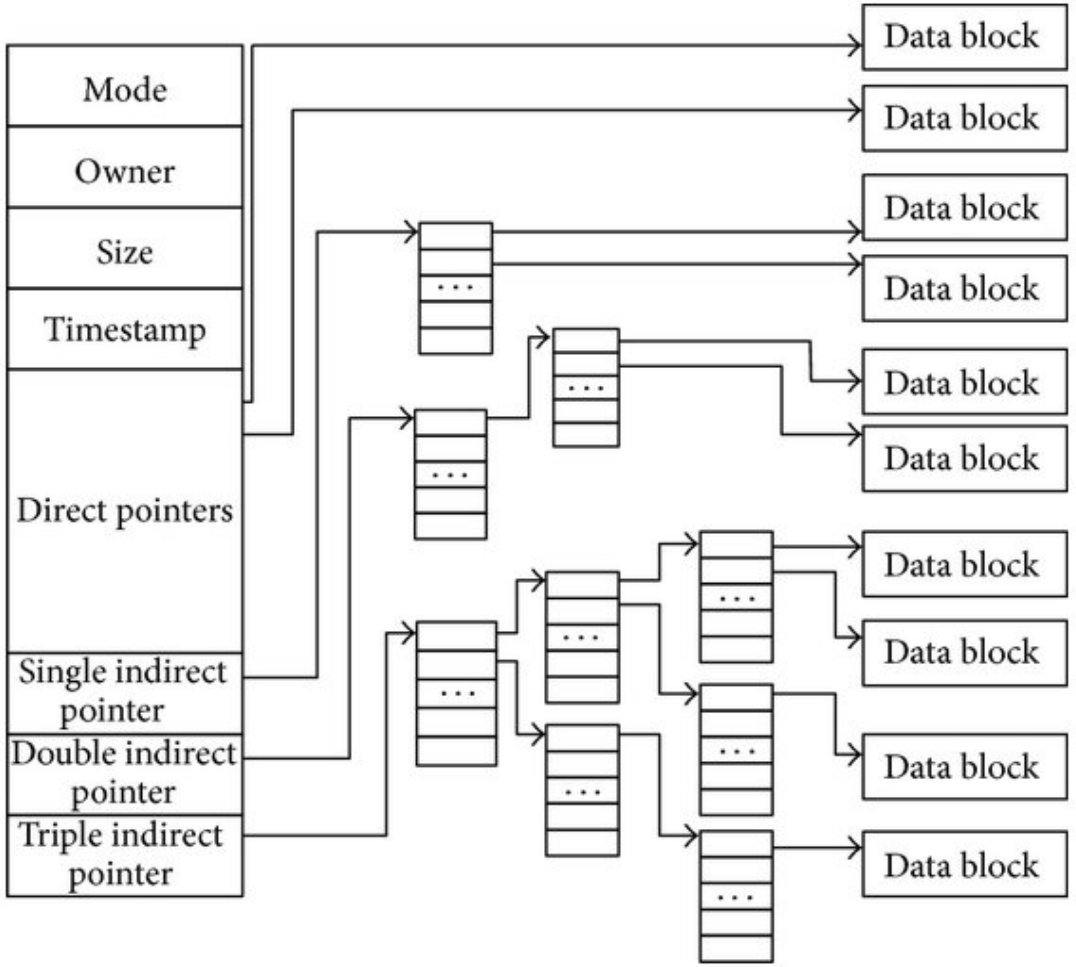
- Каталог содержит только
  - имя файла
  - номер инода (ссылку на блок метаданных)
- Метаданные собраны в отдельной структуре (таблица инодов, MFT в NTFS)



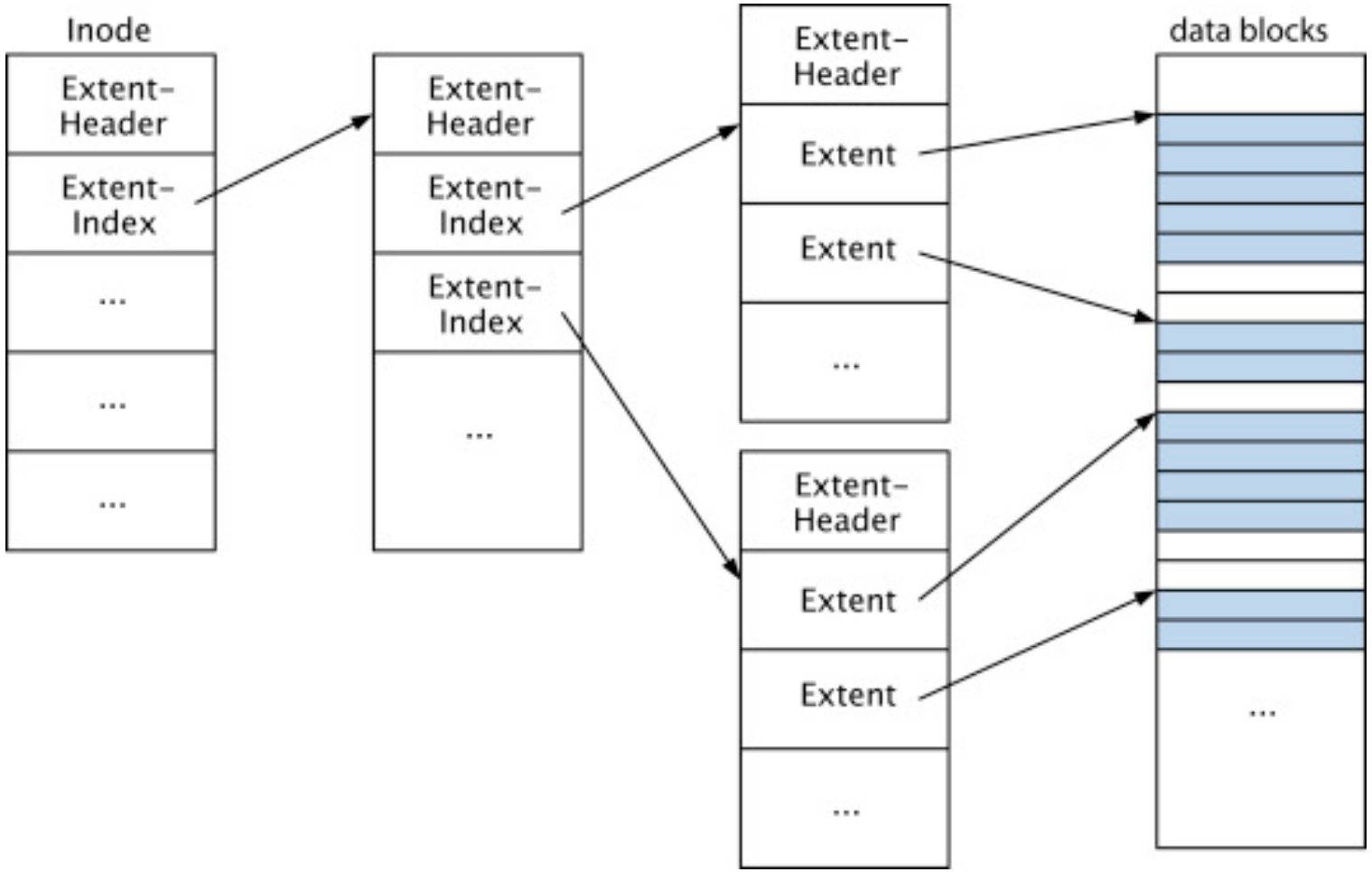
Пример: структура ext3



Пример: структура инода ext3



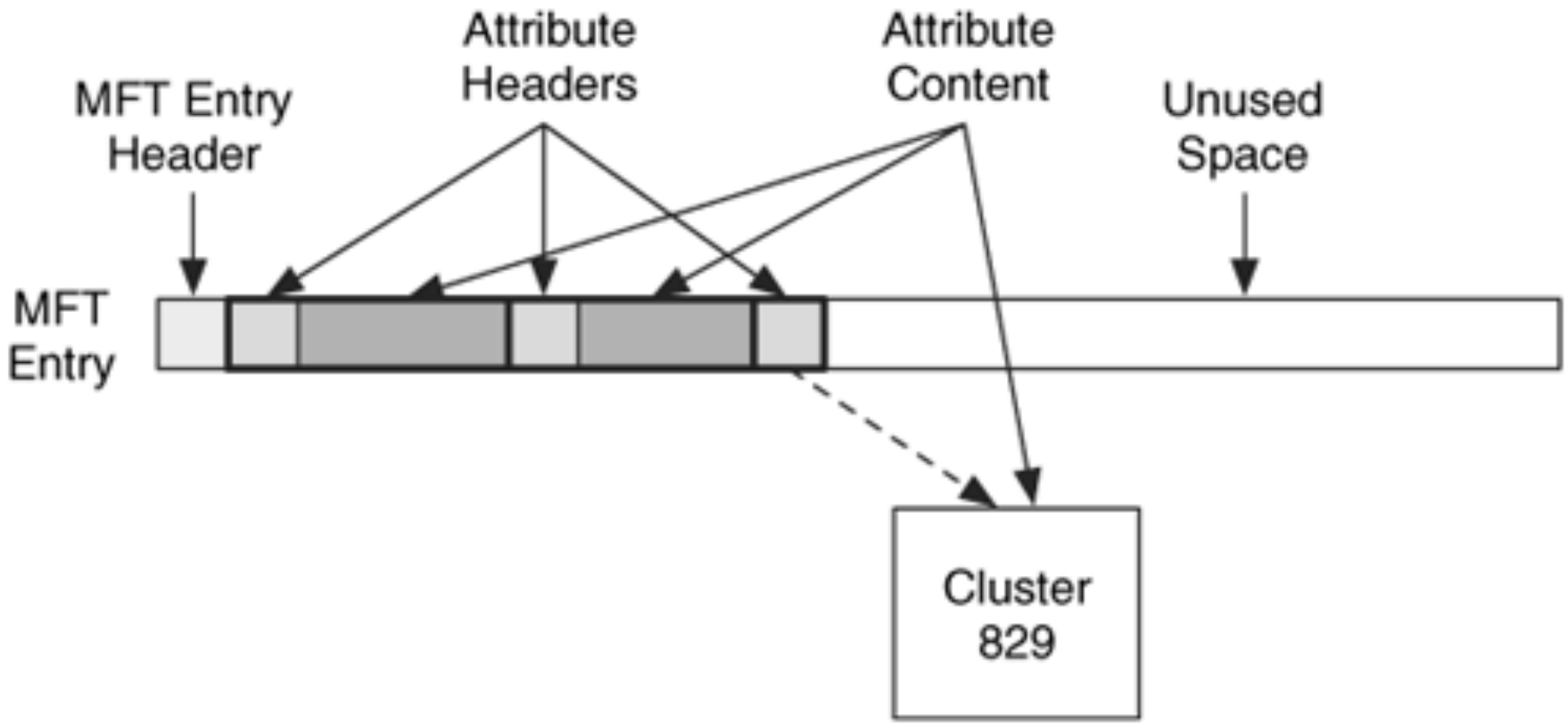
Пример: структура инода ext4



Пример: структура записи MFT NTFS

- Запись MFT состоит из атрибутов.
- Атрибут может быть
  - резидентным (размещен в записи MFT)
  - нерезидентным (под него выделены экстенты данных)
  - Размещение больших атрибутов описывается в виде B-дерева
- Атрибуты:
  - Обязательные атрибуты файла: тип, дата, владелец, security id
    всегда резидентный
  - Имена файла (если у файла есть жесткие ссылки, может быть несколько)
  - ACL (список управления доступом)
  - Потоки данных (может быть несколько)

Структура записи MFT NTFS



Проблемы типичных современных ФС

- Список (обычно битмап) свободных блоков и списки блоков файлов – разные структуры
- При выключении питания в неподходящий момент, могут возникать потерянные или дважды посчитанные блоки
- Старое решение: dirty flag и проверка ФС
- Современное решение - журналирование