Операционные системы

Файловые системы [1]

Соловьев А. В.

ПетрГУ – КИИСиФЭ

(Rev. 2018 06 07)

[1] По материалам «Таненбаум Э., Бос Х. Современные операционные системы. СПб.: Питер, 2015.»

Понятие файловой системы

Долговременное хранилище информации:

- Предоставляет возможность хранения большого количества инф-ции.
- Информация должна пережить прекращение работы использующего её процесса.
- К инф-ции должны иметь одновременный доступ несколько процессов.

Устройство долговременного хранения информации логически представляется линейной последовательностью блоков фиксированного размера, которое поддерживает две операции: чтение блока k и запись блока k. Эти операции неудобны при решении задач ОС. Вводится абстракция – *файл* – логический информационный блок, создаваемый процессом.

Структура файлов, их имена, доступ к ним, их использование, защита, реализация и управление определяется подсистемой операционной системы, называемой *файловой системой*.

Также *файловая система* – это метод организации файлов на устройствах хранения данных.

Пользовательский интерфейс файловой системы

Имена файлов

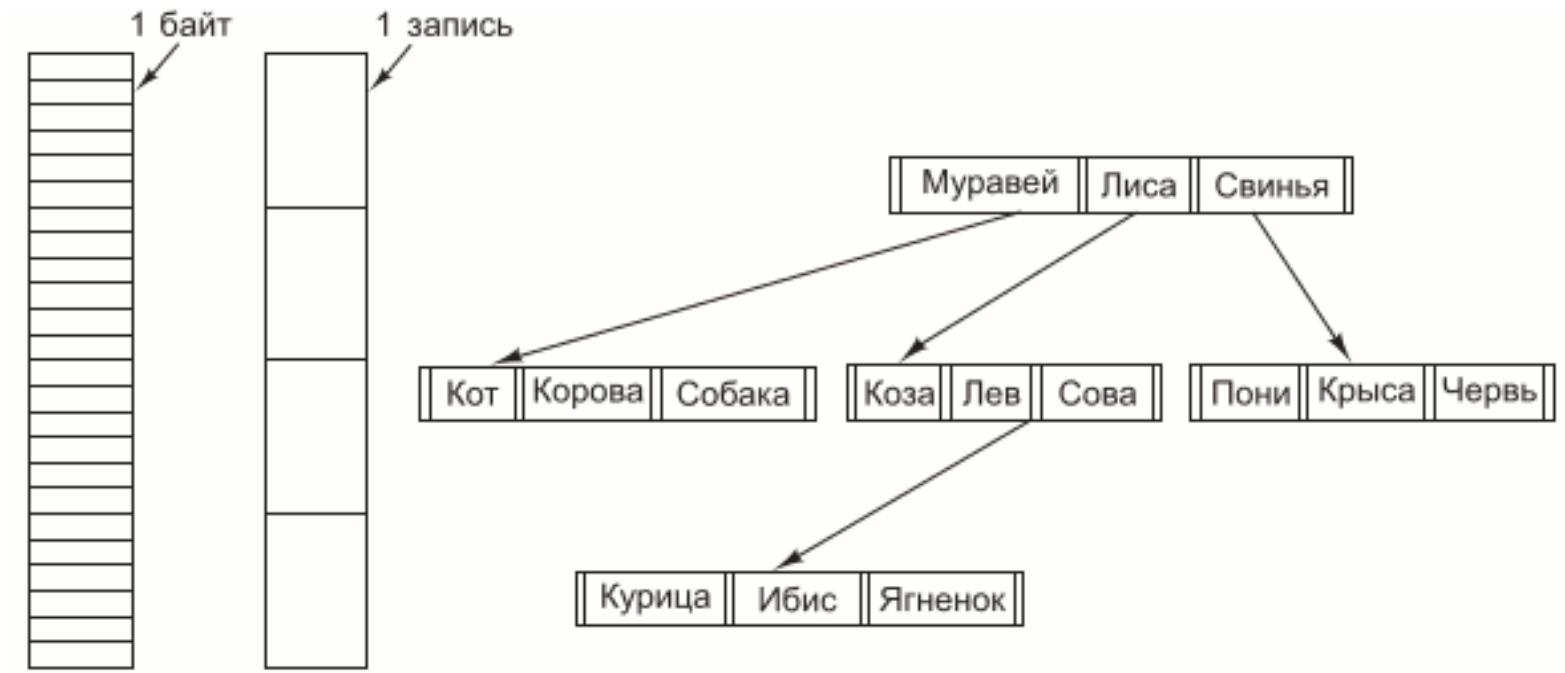
При создании файла процесс присваивает ему имя. Когда процесс завершается, файл продолжает существовать, и к нему по этому имени могут обращаться другие процессы.

ФС определяет правила именования:

- Максимальная длина имени (8/11, 255, 65535, ...)
- Допустимый набор символов в имени (A-Z0-9, unicode, \/:*?"<>|+ .%!@).
- Регистрозависимость имени.
- Структура имени («расширение»).

Структура файлов

Примеры структур файлов: последовательность байтов; последовательность записей; дерево.



Типы, формат файлов, доступ

Типы файлов:

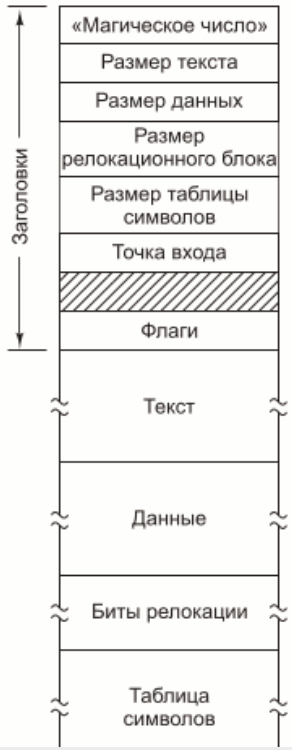
- обычный;
- каталог;
- символическая связь («ярлычок»);
- блочное устройство;
- символьное устройство;
- ...

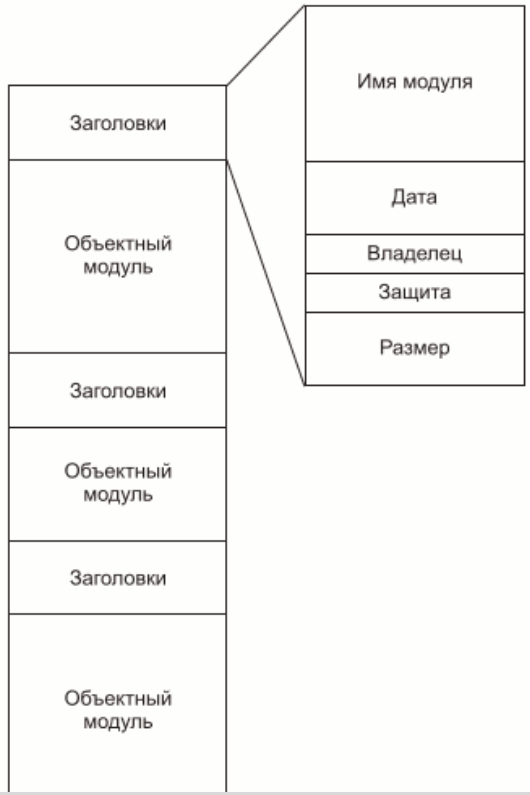
Доступ к файлам:

- последовательный,
- произвольный.

Формат (обычных) файлов:

- ASCII-файлы («текстовые»);
- двоичные файлы (исполняемые, библиотеки, архивы, документы, ...).





Атрибуты файлов (метаданные)

- Атрибуты защиты.
- Пароль.
- Создатель.
- Владелец.
- Флаг «только для чтения».
- Флаг «скрытый».
- Флаг «системный».
- Флаг «архивный».
- Флаг «ASCII/двоичный».
- Флаг произвольного доступа.
- Флаг «временный».
- Флаги блокировки.
- Длина записи.
- Позиция ключа.
- Длина ключа.
- Время создания.
- Время последнего доступа.
- Время внесения последних изменений.
- Текущий размер.
- Максимальный размер.

Операции с файлами

- Create – объявить о появлении нового файла и установить ряд атрибутов.
- Delete – удалить файл и освободить дисковое пространство.
- Open – перед использованием файла процесс должен его открыть (дать возможность системе извлечь и поместить в ОЗУ атрибуты и перечень адресов на диске, чтобы ускорить доступ к ним при последующих вызовах).
- Close – после завершения всех обращений к файлу файл должен быть закрыт, чтобы освободить место во внутренних структурах.
- Read – считать данные из файла (обычно байты поступают с текущей поз.). Вызывающий процесс должен указать объём и буфер.
- Write – записать данные в файл (обычно с текущей поз.). Если эта позиция – EOF, то его размер увелич., иначе новые данные замещают существующие.
- Append – добавить данные в конец файла.
- Seek – переметить указатель файла к определенной позиции в файле. После завершения этого вызова данные могут считываться или записываться с этой позиции. (Только для файлов произвольного доступа).
- Get attributes – получить атрибуты.
- Set attributes – установить атрибуты.
- Rename – изменить имя существующего файла.

Иерархические системы каталогов

Для упорядочения файлов в файловой системе используются каталоги (папки), которые сами по себе являются файлами. Без использования каталогов совокупность файлов представляется одним уровнем иерархии (корневой каталог). Более распространены иерархические системы.

Для указания имени файла может использоваться *абсолютное имя*, содержащее полный путь к файлу от корневого каталога:

```
\\??\C:\WINDOWS\SYSTEM32\DRIVERS\ETC\HOSTS
/etc/hosts
>usr>local>etc>hosts
```

В реализации ФС возможны ограничения на глубину иерархии и длину абсолютного имени.

Другой разновидностью имени является *относительное имя*.
Один из атрибутов процесса – рабочий (текущий) каталог. Все неабсолютные имена рассматриваются относительно рабочего каталога.

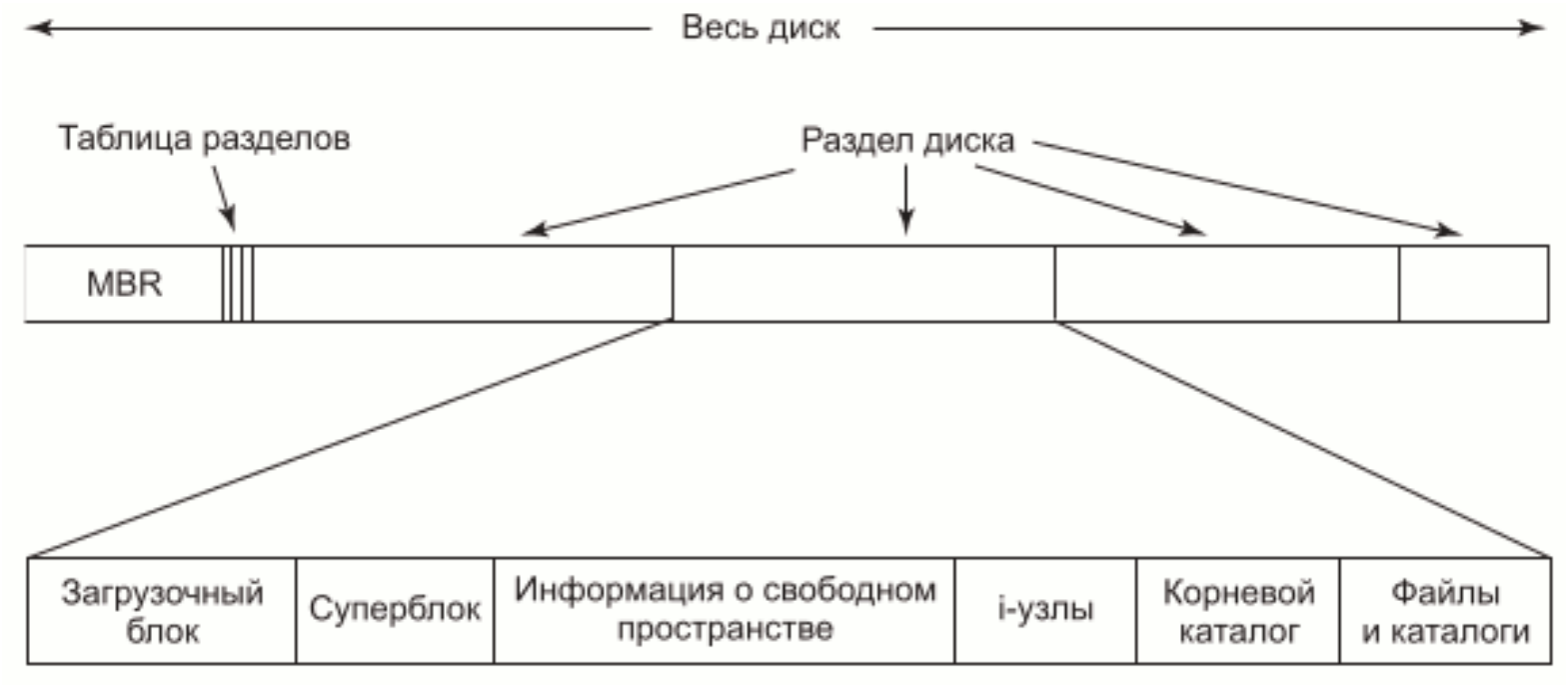
Большинство ФС, поддерживающих иерархию, имеют в каждом каталоге специальные элементы «.» и «..».

Операции с каталогами

- Create – создать каталог. Каталог создается пустым (за искл. «.» и «..»).
- Delete – удалить (пустой) каталог.
- Opendir – открыть каталог. Каталоги могут быть прочитаны, например, для вывода имен всех файлов, содержащихся в каталоге.
- Closedir – закрыть каталог. Когда каталог прочитан, он должен быть закрыт, чтобы освободить пространство во внутренних структурах ОС.
- Readdir – прочитать следующую запись (в стандартном формате) из открытого каталога.
- Rename – переименовать каталог.
- Link/unlink – манипулирование жёсткими ссылками.

Реализации файловой системы

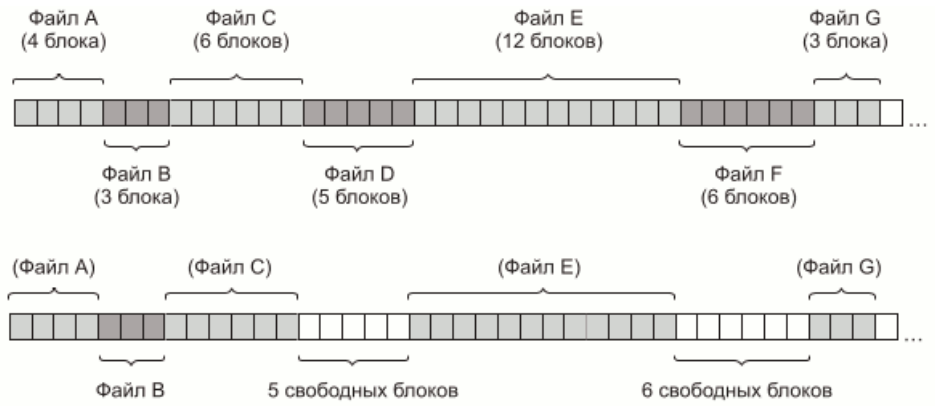
Размещение ФС на диске



BIOS – MBR (CHS) – до 2 ТБайт.
EFI – GPT (LBA) – до 9 ЗБайт.

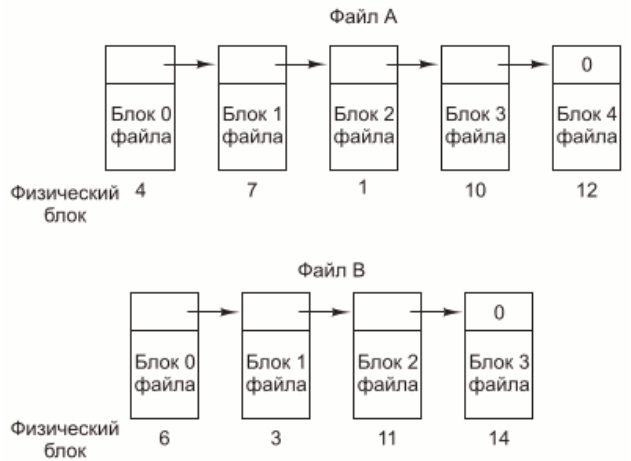
Реализация файлов (1)

Непрерывное размещение



- Просто реализуется.
- Весь файл можно считать за одну операцию.
- Не подходит для изменяемых ФС.

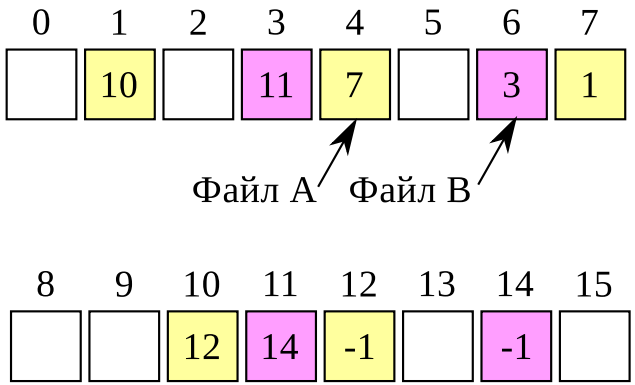
Связанный список



- Подходит для изменяемых ФС.
- Медленный произвольный доступ.
- Объём блока хранилища не кратен 2.

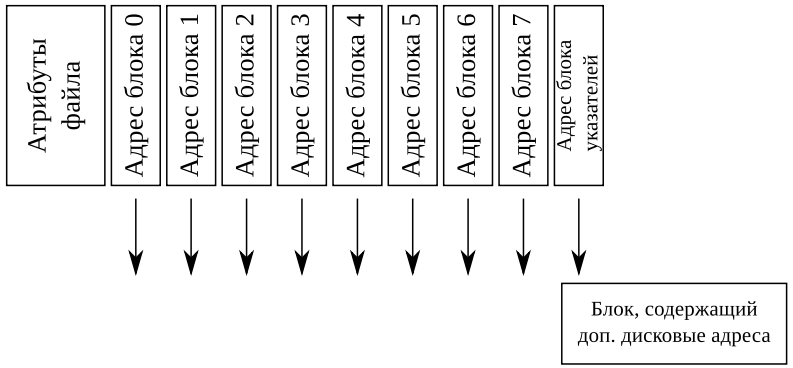
Реализация файлов (2)

Связанный список с таблицей



- Возможен быстрый произвольный доступ.
- Простой элемент каталога (стартовый блок).
- Вся таблица должна находиться в памяти (плохо масштабируется).

Индексные узлы (i-node)



- Возможен быстрый произвольный доступ.
- I-узел хранится в памяти, только когда файл открыт.
- Более сложная структура для больших файлов.

Реализация каталогов

Основная функция системы каталогов – преобразование ASCII-имени файла в информацию, необходимую для определения местоположения данных.
Где и как следует хранить атрибуты?

- Атрибуты хранятся в записи каталога (фиксированного размера).
- Атрибуты хранятся в i-узлах, запись каталога = имя + номер i-узла (запись может быть переменного размера).

При удалении элемента каталога переменного размера необходимо производить уплотнение записей.

Для ускорения поиска по каталогу могут применяться хэш-таблицы (усложняется администрирование).

Файловые системы с журнальной структурой

Log-structured File System (LFS)
Цель – повышение производительности за счёт группировки массовых операций записи мелких порций данных.
Весь диск – большой журнал.
Все блоки, ожидающие записи, собираются в один непрерывного сегмент и в таком виде записываются на диск в конец журнала. Отдельный сегмент может вперемешку содержать i-узлы, блоки каталога и блоки данных. В начале каждого сегмента находится сводная информация, в которой сообщается, что может быть найдено в этом сегменте (для поиска i-узлов ведётся массив i-узлов, проиндексированный по i-номерам). Сводная информация фиксируется в *контрольный момент* (checkpoint), обычно каждые 30 сек.
LFS использует *очищающий поток* (сборщик мусора), который занимается тем, что осуществляет круговое сканирование журнала с целью уменьшения его размера.
Недостатки:

- Возможна серьёзная фрагментация (для редко и мало меняющихся файлов), что сказывается на производительности.
- Существенно падает производительность, когда свободное место подходит к концу (постоянно запускается сборщик мусора).

Журналируемые файловые системы

Проблема – файловые операции неатомарны, например, удаление файла:

1. Удалить элемент каталога.
2. Освободить i-узел, поместив его в пул свободных i-узлов.
3. Вернуть все дисковые блоки файла в пул свободных.

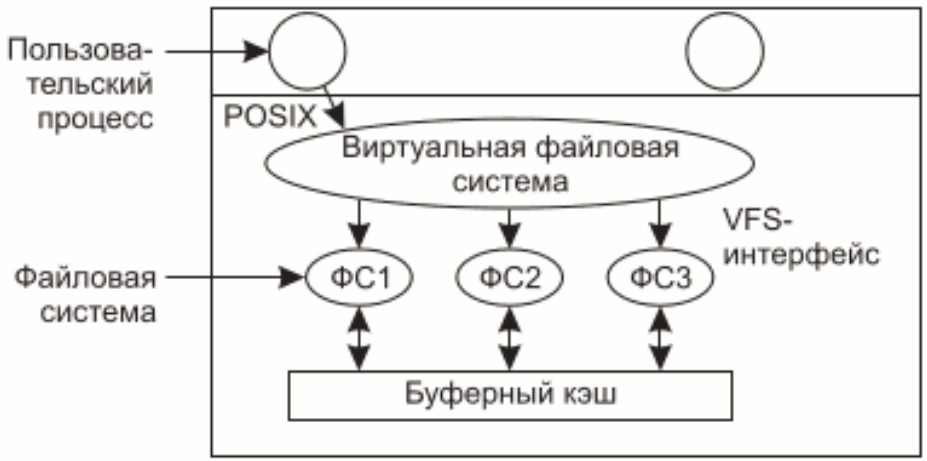
В журналируемой ФС:

1. Делается запись в журнале с перечнем действий (запись фиксируется на диске).
2. Выполняются намеченные операции.
3. Журнальная запись удаляется (помечается как выполненная).

При восстановлении после отказа просматриваются записи журнала о незавершённых операциях.
Важно, чтобы операции, фиксируемые в журнале, были *идемпотентными* (эффект от операции не зависит от количества её повторов).
Эквивалентно концепции *атомарных транзакций*.

Виртуальные файловые системы (1)

Sun (1986), Virtual file system (VFS)



Цель – выделить какую-то часть ФС, являющуюся общей для всех ФС, и поместить её код на отдельный уровень, из которого вызываются расположенные ниже конкретные ФС с целью фактического управления данными.
Верхний интерфейс – POSIX: open, read, write, lseek, ...
Нижний интерфейс – считать/записать блок, структуры: суперблок, v-узел, каталог, ...

Виртуальные файловые системы (2)

При загрузке системы VFS регистрирует корневую ФС. При подключении (монтировании) других ФС они также должны быть зарегистрированы в VFS. При регистрации ФС предоставляет список адресов функций, необходимых VFS, по одному на каждый VFS-объект.

v-узел = i-узел + таблица функций

Размер блока

Почти все ФС разбивают файлы на блоки фиксированного размера, которые не нуждаются в смежном расположении.

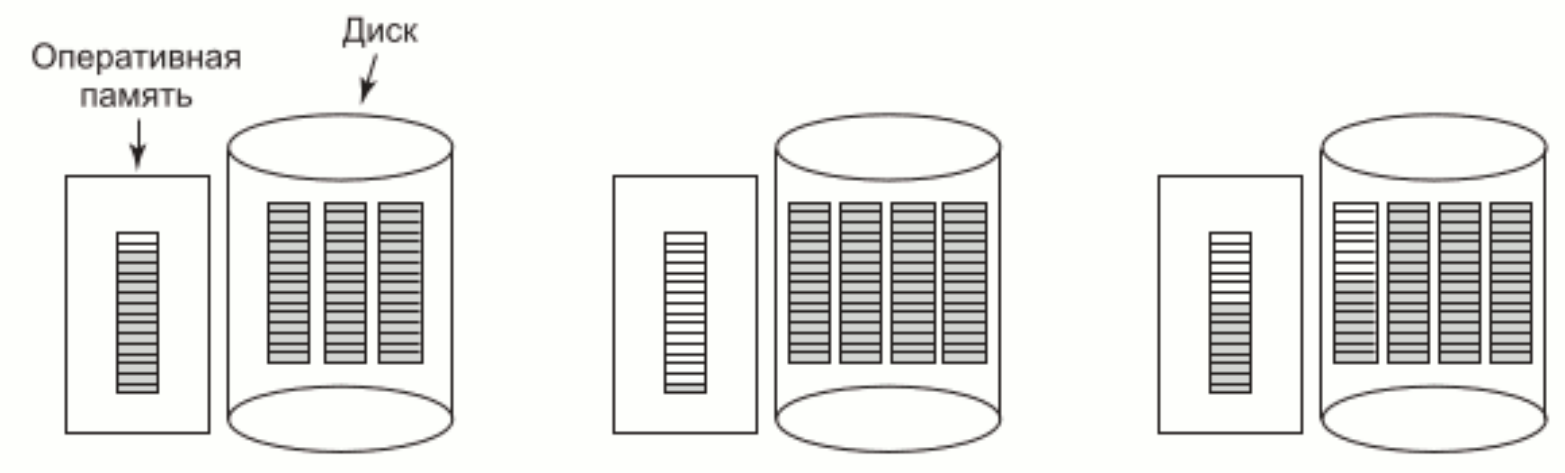
- Крупный размер блока приводит к неэффективному размещению маленьких файлов (место тратится впустую).
- Маленький размер блока приводит к увеличению количества операций позиционирования при чтении/записи блоков файла (время тратится впустую).

Традиционно в ФС используют блоки размером 1-4 Кбайт, иногда до 64 Кбайт.

Отслеживание свободных блоков

- Связанный список номеров свободных блоков. Как правило, для его хранения используются сами свободные блоки (т. е. хранение обходится «бесплатно»). Модификация: список последовательностей свободных блоков (если диск сильно фрагментирован – неэффективно).
- Битовая карта свободных блоков.

В памяти можно хранить, например, только один блок списка или карты.



Дисковые квоты

Чтобы не дать пользователям возможности захватывать слишком большие области дискового пространства, многопользовательские ОС часто предоставляют механизм навязывания дисковых квот. Сисадмин назначает каждому пользователю максимальную долю файлов и блоков, а ОС гарантирует невозможность превышения этих квот.

В числе атрибутов любого файла имеется запись, сообщающая о том, кто является владельцем файла. Любое увеличение размера файла будет засчитано в квоту владельца (определяется для открытого файла в процессе работы с ним). Попытка добавить что-нибудь к файлу, когда достигнут лимит, приведет к ошибке.

Лимиты бывают жёсткие и мягкие, на количество блоков и на количество i-узлов. Мягкий лимит может быть превышен в течение сеанса. Жесткий лимит не может быть превышен ни при каких условиях.

Резервное копирование файловой системы

Физическая архивация – поблочное копирование диска без учёта структуры ФС (включая неиспользуемые и дефектные блоки, а также блоки временных файлов и файлов подкачки).
Логическая архивация – пофайловое копирование данных.
Аспекты:

- Все ли файлы копировать? (Можно пропустить временные файлы, специальные файлы и т. п.)
- Возможно *инкрементное резервное копирование* – не копируются файлы, которые не изменились со времени предыдущего резервного копирования.
- Использовать ли сжатие? (Экономит место, но затратнее по времени и меньше надёжность).
- Как копировать активную ФС?
- Как обеспечить безопасность резервных копий?

Непротиворечивость файловой системы

На примере fsck.

1. Проверки блочной непротиворечивости.

Создаётся две таблицы, каждая из которых состоит из счётчика для каждого блока, изначально установленного в нуль. Счётчики в первой таблице отслеживают количество присутствия каждого блока в файлах, а счётчики во второй таблице регистрируют количество присутствий каждого блока в списке свободных блоков (или в битовом массиве свободных блоков).

Если у ФС нет противоречий, у каждого блока будет 1 либо в первой, либо во второй таблице.

Обнаружение потерянных свободных блоков, дублирования свободных блоков в списке, пересекающихся файлов.

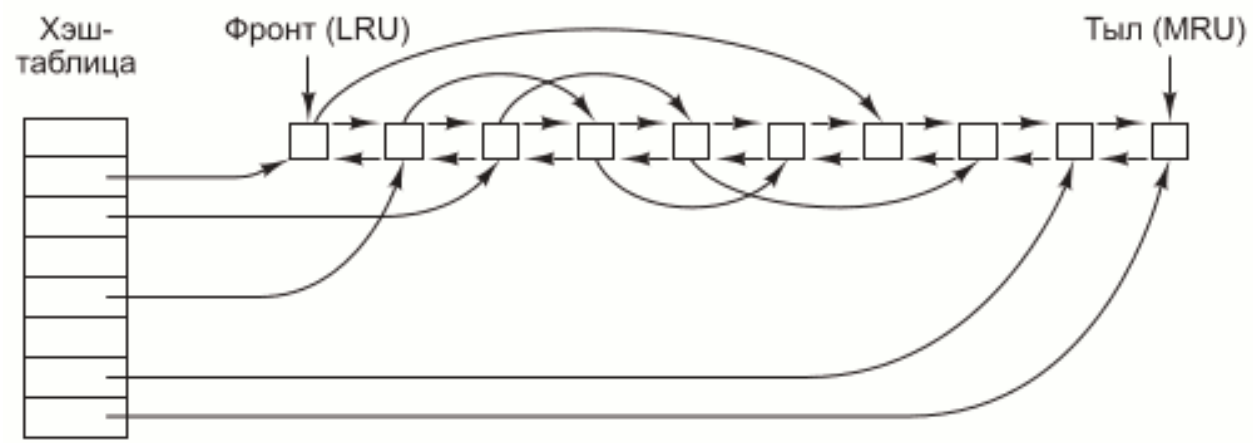
2. Проверка непротиворечивости системы каталогов.

Для каждого i-узла в каждом каталоге пересчитывается значение счетчика, соответствующее количеству использований файла.

В непротиворечивой ФС значения будут совпадать.

3. Прочие проверки (формат имени файла, номеров i-узлов, режима доступа и т. п.)

Кэширование



Для быстрого поиска в кэше используется хэш-таблица (показаны цепочки коллизий). Алгоритм замещения – любой из рассмотренных ранее, например, LRU с модификацией:

- Велика ли вероятность того, что данный блок вскоре снова понадобится?
- Важен ли данный блок с точки зрения непротиворечивости ФС?

Кэш с отложенной записью (write-back) (периодический sync) или *кэш со сквозной записью* (write-through).

Опережающее чтение (read-ahead)

Попытаться получить блоки в кэш ещё до того, как они понадобятся, чтобы повысить соотношение удачных обращений к кэшу.

Когда у ФС запрашивается блок k какого-нибудь файла, она выполняет запрос, но, завершив его выполнение, проверяет присутствие в кэше блока $k+1$. Если этот блок в нём отсутствует, она планирует чтение блока $k+1$.

Такая стратегия пригодна при последовательном доступе, но ухудшает производительность при произвольном доступе. ФС может попытаться угадывать режим доступа:

- при открытии – последовательный,
- после lseek – произвольный,
- ряд последовательных read – последовательный.

Сокращение кол-ва перемещений блока головок диска

Размещение блоков с высокой степенью вероятности обращений последовательно, предпочтительно на одном и том же цилиндре.
1. Выбирать новый блок под файл среди свободных ближе к остальным (для битового массива – легко, для связанного списка – непросто).
2. Объединение блоков в кластеры (кэширование и файловые операции – поблочно, а распределение места – покластерно).
3. Деление структуры ФС на группы цилиндров.

Для SSD скорости произвольного и последовательного доступа не отличаются.
Однако в каждый блок запись может производиться ограниченное количество раз.
Надо следить за равномерным распределением износа по диску.

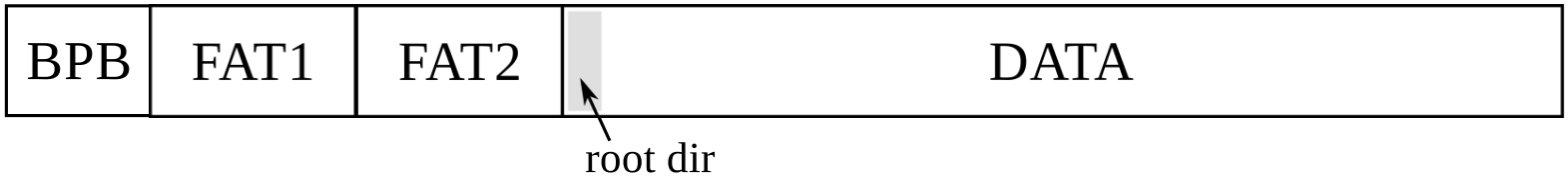
Дефрагментация дисков

По мере создания и удаления файлов диск обычно приобретает нежелательную фрагментацию, где повсеместно встречаются файлы и области свободного пространства. Вследствие этого при создании нового файла используемые им блоки могут быть разбросаны по всему диску, что ухудшает производительность.
С ОС может поставляться утилита дефрагментации (Windows – defrag): перемещение файлов с места на место, чтобы они размещались непрерывно, и объединения всего (или, по крайней мере, основной части) свободного дискового пространства в один или в несколько непрерывных участков на диске.
Некоторые файлы не могут быть перемещены; к ним относятся файлы, используемые в страничной организации памяти и реализации спящего режима, а также файлы системных журналов, поскольку выигрыш от этого не оправдывает затрат, необходимых на администрирование. В некоторых системах такие файлы все равно занимают непрерывные участки фиксированного размера и не нуждаются в дефрагментации.
Файловые системы Linux (ext2/ext3) обычно меньше страдают от дефрагментации благодаря способу выбора дисковых блоков, поэтому принудительная дефрагментация требуется довольно редко.

ФС на SSD-накопителях не следует дефрагментировать.

Примеры файловых систем

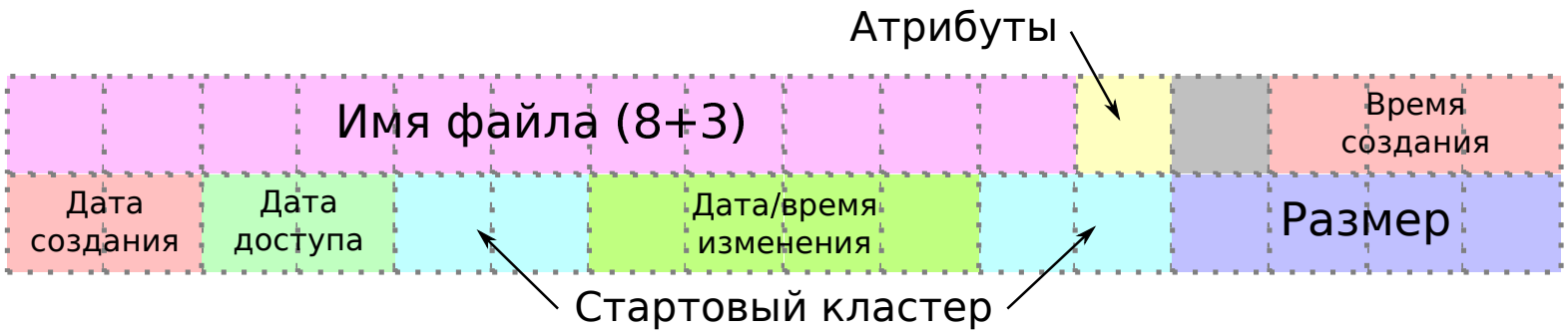
Файловая система MS-DOS (FAT) (1)



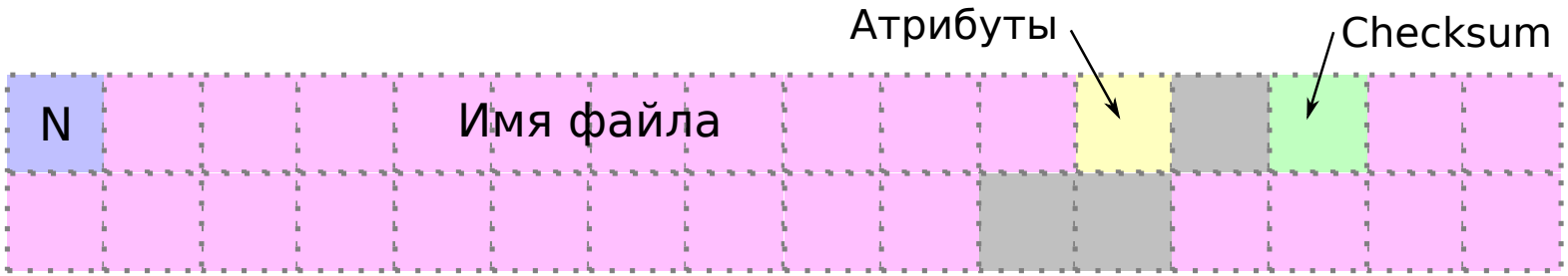
BPB = BIOS Parameter Block («загрузочный сектор»): размер сектора, размер кластера, кол-во копий FAT, размер корневого каталога, тип носителя, общее кол-во секторов и т. п.
Элемент FAT может иметь размер 12, 16 или 32 (используется 28) бит. Если кол-во кластеров $<$ 4085 – FAT-12, если от 4085 до 65524 – FAT-16, если $>$ 65524 – FAT-32.
Значения элементов FAT: 0 – свободный, xxxFF7 – сбойный, xxxFFF – EOF, остальные – занятый (следующий кластер в цепочке).
Кластеры 0 и 1 зарезервированы, отведены под корн.к. В FAT12/16 корн.к. фиксированного размера. В FAT-32 – может расти.
У корн.к. нет атрибутов (нет собственного имени, меток даты и времени), не содержит элементы «.» и «..». Может содержать файл метки тома.

Файловая система MS-DOS (FAT) (2)

Формат элемента каталога:



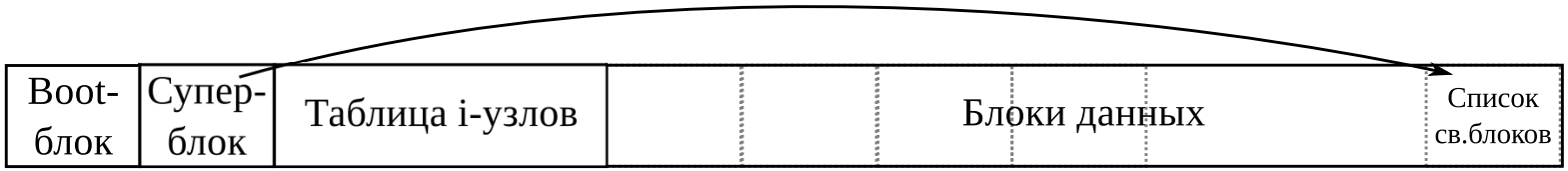
LFN-элемент.



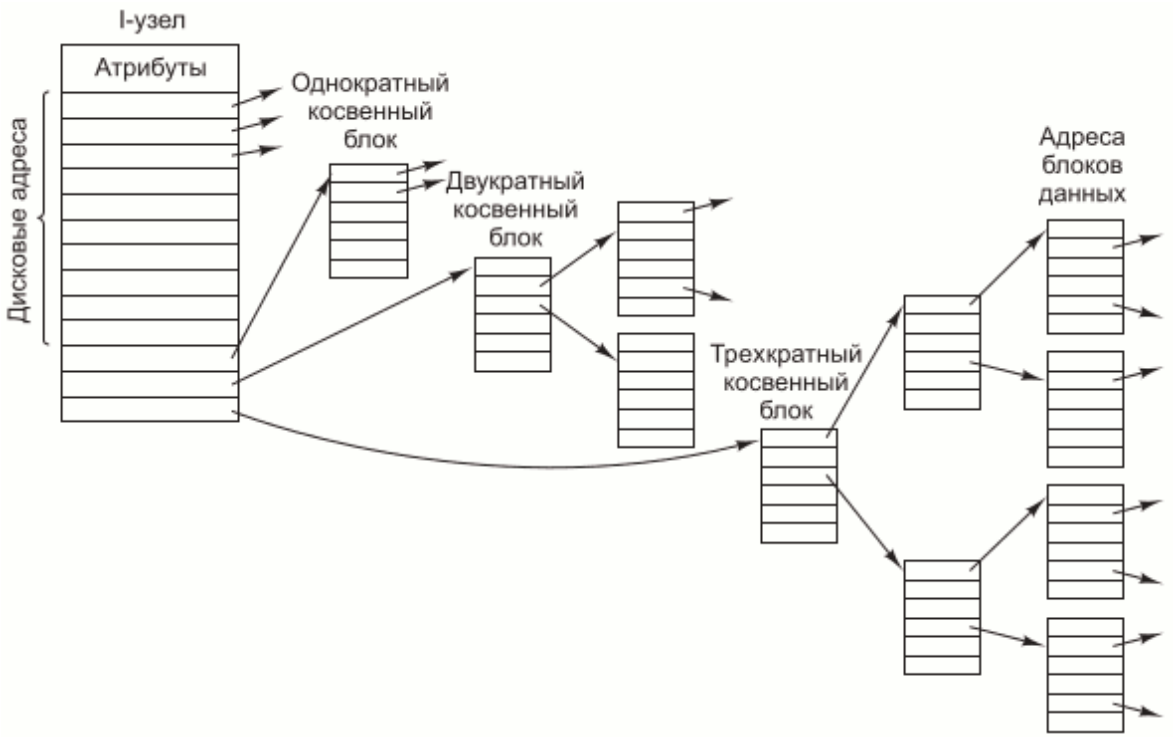
Для LFN длина имени до 255 симв.

Макс. размер ФС на основе FAT-32 – до 2 Тбайт.

Файловая система UNIX V7 (1)



Суперблок содержит параметры раздела: размер блока, количество блоков, количество i-узлов, адрес списка свободных блоков и проч.



I-узлы по 64 байта содержат:

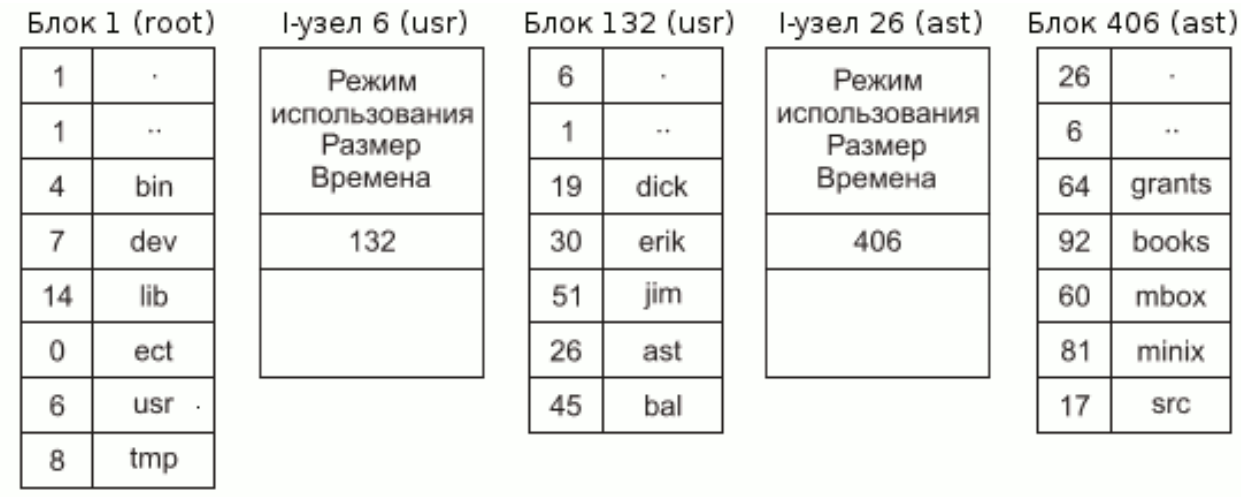
- mode (2б),
- кол-во связей (2б),
- UID (2б) и GID (2б),
- размер (4б),
- адреса 13 блоков (39б),
- generation (1б),
- три тайм-штампа (12б).

Файловая система UNIX V7 (2)

Элементы каталога имеют фиксированный размер – 16 байт:

Номер i-узла	Имя файла (связь) (14 байт)

Пример поиска файла /usr/ast/mbox



Файловая система компакт-дисков (ISO 9660) (1)

CD – протяженная спираль с записью битов в линейной последовательности. Биты на протяжении этой спирали разбиты на логические блоки по 2352 байта. За вычетом преамбул, коррекции ошибок и др. служебных данных полезная часть занимает 2048 байт.
Отдельный компакт-диск также может быть разбит на несколько логических томов (разделов).

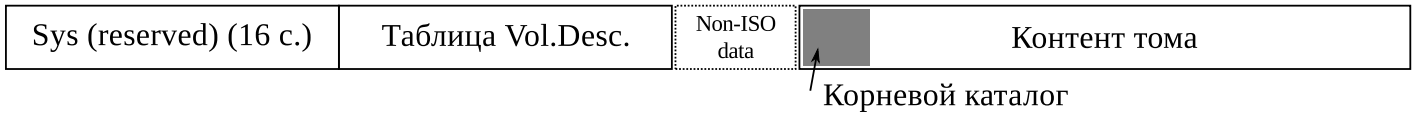
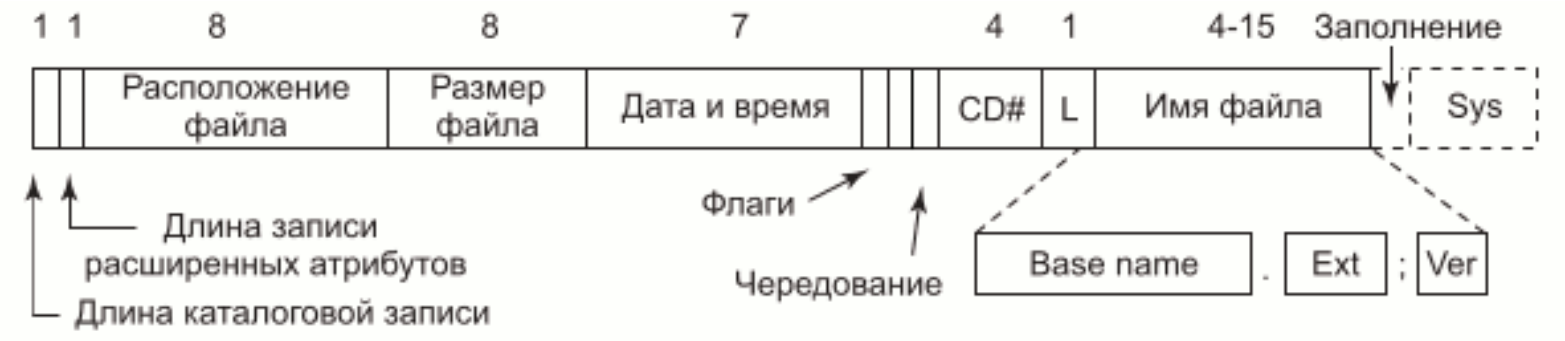


Таблица дескрипторов томов содержит как минимум Primary Volume Descriptior (0x01) и Ending Volume Descriptor (0xFF). Каждый дескриптор занимает 1 сектор (2048 байт).
Дескриптор тома содержит: тип, стд.идентификатор (CD001), версию, метку системы (32б), метку тома (32б), размер тома, размер блока, указатель на корневой каталог, информацию об издателе, временны́е метки и др. инф.

Файловая система компакт-дисков (ISO 9660) (2)

Хранение многобайтных целых чисел («формат 733»): 0x11223344 →

Элементы каталога переменного размера (max 255 байт):



Первые два элемента описывают текущий и родительский каталог.
Уровни ISO 9660 (1988):

1. Имена файлов «8.3», только 0-9_A-Z, без фрагментации, каталоги без расширения, max глубина – 8.
2. Имена до 31 симв.
3. Файлы могут быть фрагментированы.

Файловая система компакт-дисков (ISO 9660) (3)

Расширения стандарта:

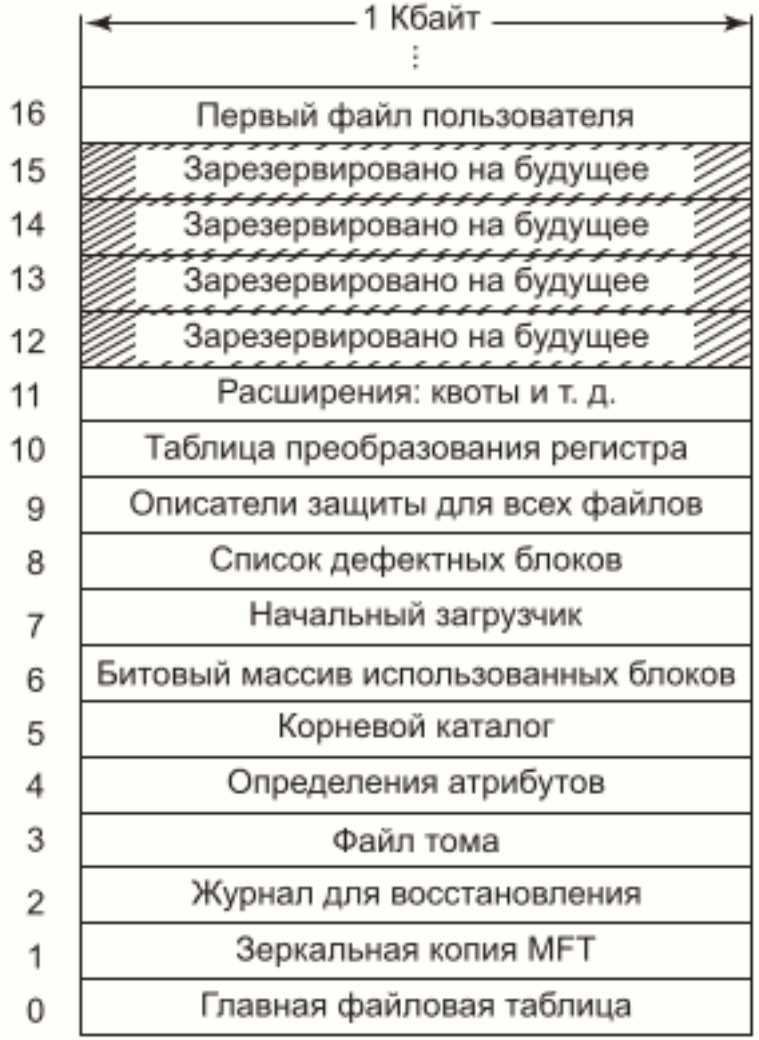
- **Joliet**: (предложено Microsoft) дополнительный набор имён файлов (до 64 симв. в кодировке UCS-2) хранится в дополнительном томе, определяемом Supplementary Volume Descriptor (SVD).
- **El Torito**: (предложено Phoenix и IBM) расширение формата загрузочного CD.
- **Rock Ridge** (IEEE P1282): в элементах каталога заполняется поле Sys, туда помещаются доп. атрибуты, такие как: PX – атрибуты POSIX, PN – старший и младший номера устройств, SL – символическая ссылка, NM – альтернативное имя (длинное, до 255 симв.), CL – расположение дочернего каталога; PL – расположение родительского каталога; RE – перемещение каталога; TF – отметки времени.

Файловая система NTFS: MFT

- 64-битная адресация
- Журналирование операций
- Имена файлов: unicode, до 255 симв., путь – до 32K.
- Файл = атрибуты + потоки
- Жёсткие ссылки, симв. ссылки (reparse points)
- Разряжённые (sparse) файлы
- Сжатие, шифрование
- СКД (ACL), квоты

Расположение MFT указывается в boot-секторе. Может содержать до 2^{48} записей по 1K.

Master File Table (MFT)



Файловая система NTFS: Атрибуты элемента MFT

Каждая запись MFT содержит заголовок (магическое число, номер использования, счётчик ссылок, размер элемента, ID основной записи и т.п.) за которым следует набор атрибутов. Атрибут состоит из ID (определяются в файле $AttrDef) и значения. Атрибут м.б. *резидентным* (значение тут же в MFT) или *нерезидентным* (значение в выделенном кластере).

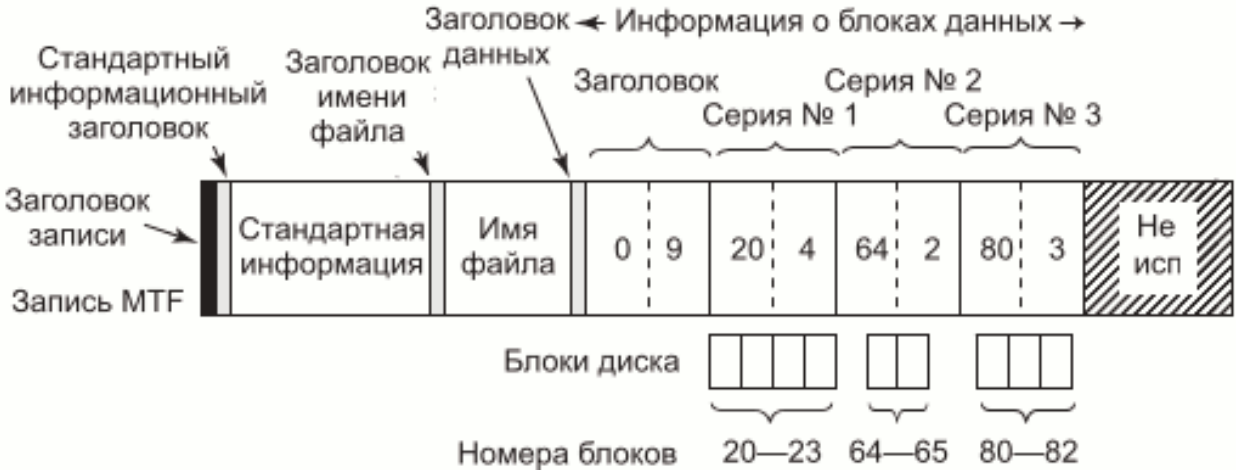
Стандартные атрибуты:

Standard information	Биты флагов, временные метки и т. д.
File name	Имя файла в Unicode
Attribute list	Местоположение доп. записей MFT
Object ID	Уникальный для данного тома 64-битный ID файла
Reparse point	Символическая ссылка или точка монтирования
Index root	Используется для каталогов
Index allocation	Используется для очень больших каталогов
Bitmap	Используется для очень больших каталогов
Data	Данные потока, могут повторяться

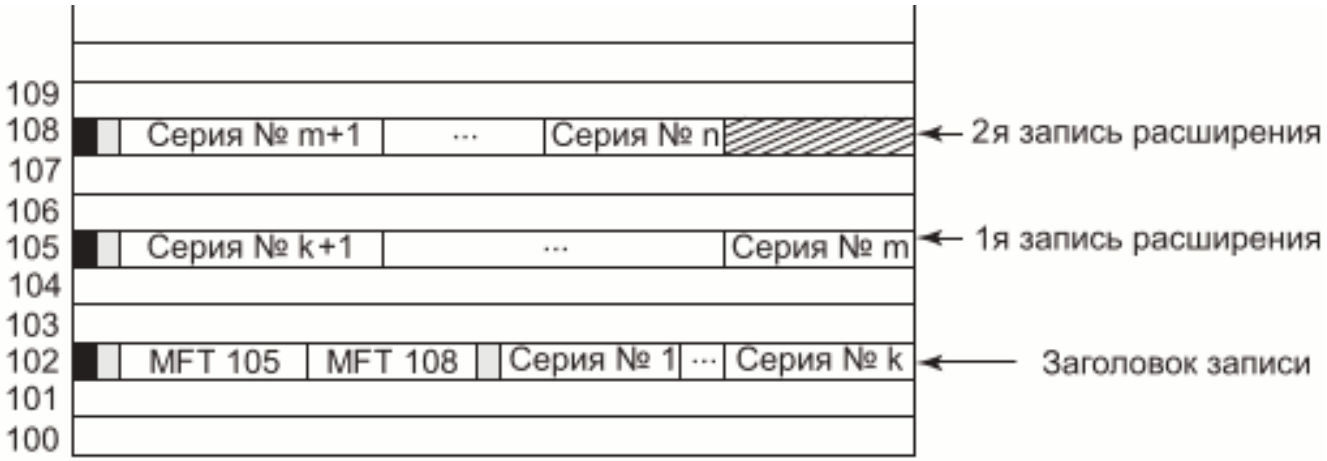
Доп. потоки имеют имя (`dirpath\filename::$DATA` – дефолтовый, `dirpath\filename:streamname:$DATA` – дополнительный).

Файловая система NTFS: Размещение файла

Запись MFT для потока из 3 участков (data run) и 9 блоков:



Файлы с пропусками называются *разрежёнными* (sparse) файлами. Например: 80 блоков, определены блоки 0–49 и 60–79 – в элементе MFT будут два атрибута, определяющих размещение потока данных.

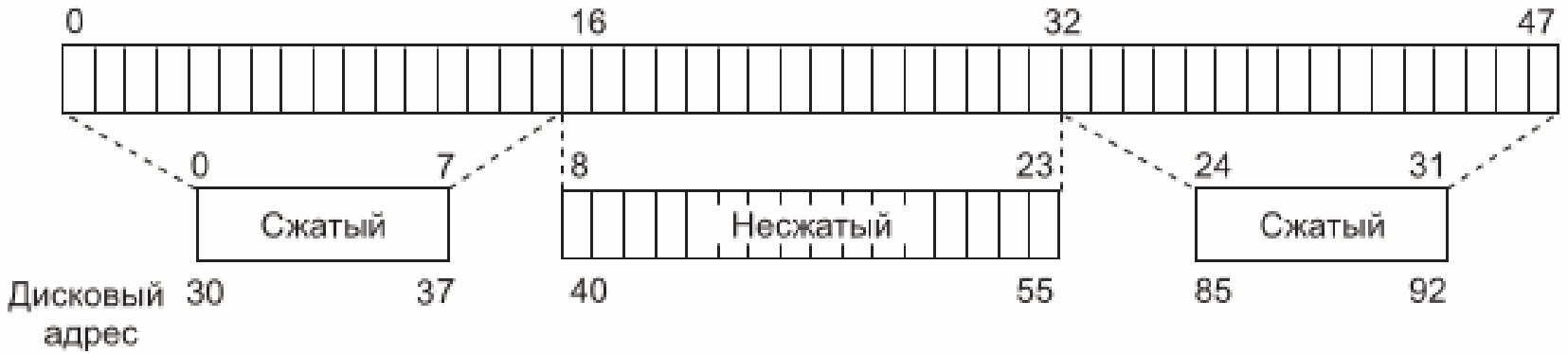


Если требуется три записи MFT для хранения всех участков...

Список расширений MFT может быть нерезидентным.

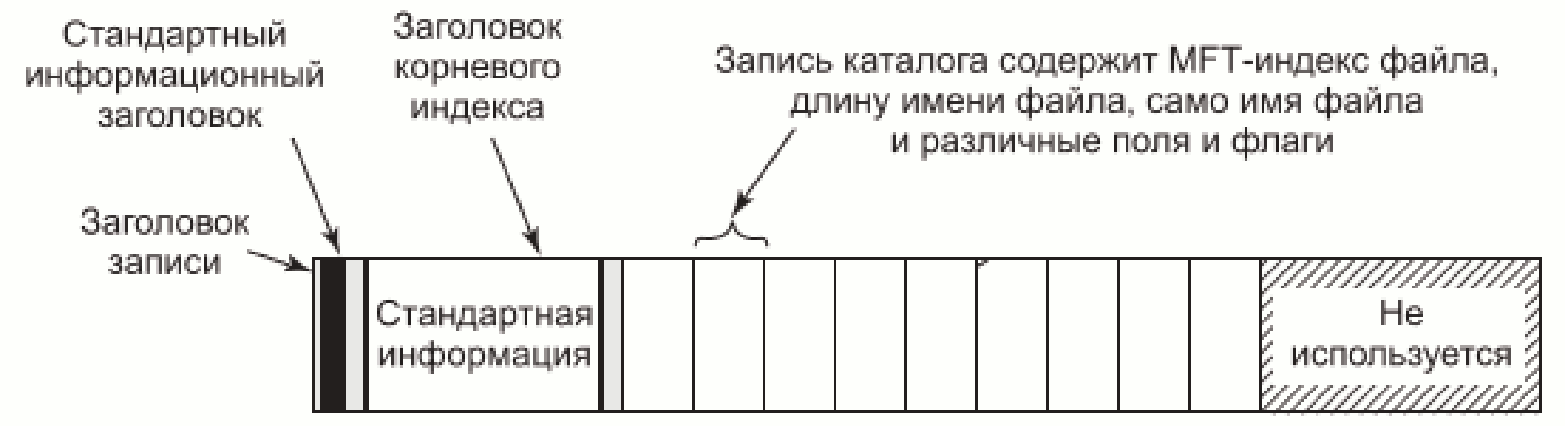
Файловая система NTFS: Сжатие файлов

Файл разбивается на фрагменты по 16 блоков. Каждый фрагмент сжимается независимо. Если сжатый фрагмент $\leqslant$ 15 блоков — два участка (сжатый и «пустой»).



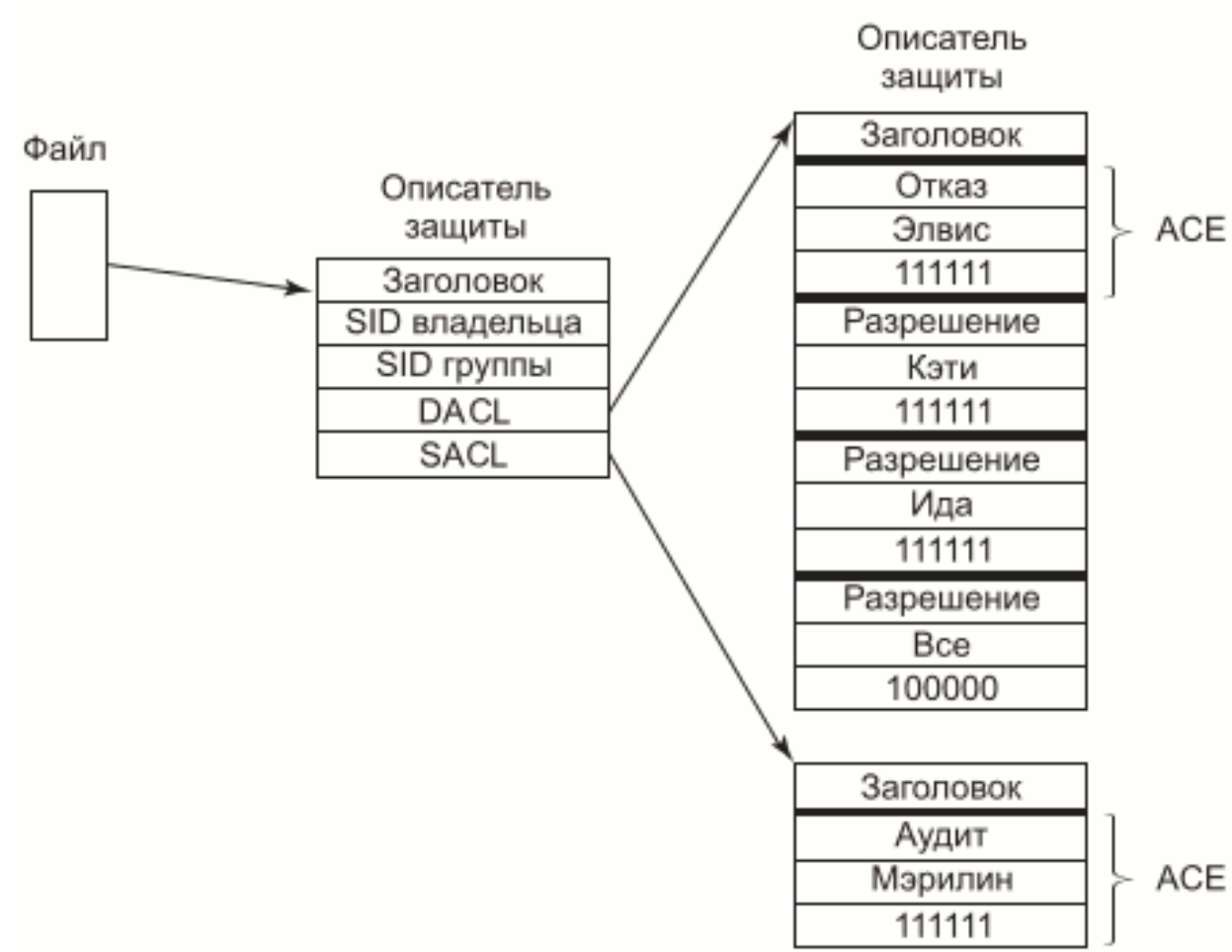
Файловая система NTFS: Каталоги

MFT-элемент для небольшого каталога:



Большие каталоги используют другой формат. Вместо линейного перечисления файлов используется дерево B+ (чтобы сделать возможным алфавитный поиск и облегчить вставку новых имен в нужное место каталога).

Файловая система NTFS: Контроль доступа (ACL)



DACL = Discretionary ACL (что разрешено? чтение/запись/и т. п.)
SACL = System ACL (что логгировать? чтение/запись/и т. п.)