Проектирование операционных систем
Лекция 7
Файловые системы и организация ввода/вывода

Боровцов Е.Г., 2016 г.

Основные задачи ОС по управлению файлами и устройствами

Основными задачами, решаемыми ОС при организации обмена с периферийными устройствами, являются следующие:

- Обеспечение эффективного планирования операций ввода/вывода с минимальной загрузкой CPU и организацией параллельной работы подсистемы I/O и CPU;
- Согласование скоростей обмена между медленными периферийными устройствами и быстрой памятью, буферизация и кэширование данных;
- Поддержка синхронных и асинхронных операций ввода/вывода;
- Разделение периферийных устройств между процессами, конкурирующими за выполнение операций ввода/вывода;
- Организация высокоуровнего логического интерфейса между подсистемой ввода/вывода и остальными компонентами ОС, позволяющего абстрагироваться от деталей выполнения операций ввода/вывода;
- Поддержка широкого спектра периферийных устройств посредством системных драйверов с унифицированным интерфейсом;
- Обеспечение простого подключения в систему новых периферийных устройств и соответствующих драйверов для управления ими;
- Возможность динамической загрузки и выгрузки драйверов;
- Поддержка нескольких типов файловых систем;

Файловая система

Файловая система — высокоуровневая абстракция, позволяющая в удобной форме оперировать с элементами информации, располагаемыми на внешних носителях. Файловая система представляет собой компоненту операционной системы, обеспечивающую высокоуровневый доступ и структурированное хранение информации на внешних устройствах.

Основными функциями файловой системы являются:
1. Идентификация файлов. Связывание имени файла с выделенным ему пространством внешней памяти.
2. Распределение внешней памяти между файлами. Для работы с конкретным файлом пользователю не требуется иметь информацию о местоположении этого файла на внешнем носителе информации. Например, для того чтобы загрузить документ в редактор с жесткого диска, нам не нужно знать, на какой стороне какого магнитного диска, на каком цилиндре и в каком секторе находится данный документ.
3. Обеспечение надежности и отказоустойчивости. Стоимость информации может во много раз превышать стоимость компьютера.
4. Обеспечение защиты от несанкционированного доступа.
5. Обеспечение совместного доступа к файлам, так чтобы пользователю не приходилось прилагать специальных усилий по обеспечению синхронизации доступа.
6. Обеспечение высокой производительности.

Файлы, их типы и свойства

Файл — поименованная совокупность данных, хранимая на внешнем устройстве. По сути, файл является механизмом абстрагирования, позволяющим хранить данные на внешних устройствах без знания подробностей о месте и способе хранения информации, а так же особенностях функционирования периферийных устройств.

Правила формирования **имён файлов** определяются той или иной операционной системой (возможна привязка файлов к определённым приложениям на основе, например, расширения файлов - ассоциирование).

Файлы могут быть **неструктурированными** (поток байтов) и **структурированными** (последовательность записей постоянной или переменной длины, а так же иерархическая или последовательная структура с индексированием).

С точки зрения организации доступа различают **последовательный** доступ и **произвольный(прямой)** доступ(реализуется по номеру записи или смещению или по индексу).

Если рассматривать файл с точки зрения его содержимого, то можно выделить **текстовые** файлы и **бинарные(двоичные)** файлы.

Выделяются **обычные(регулярные) файлы**, **каталоги** и **специальные файлы** (файлы устройств, поименованные каналы, сокеты).

Атрибуты файлов

Как правило, в любой файловой системе с файлом связывается некоторый набор **атрибутов,** которыми могут быть, например, следующие:

Права доступа к файлу	кто и для каких операций имеет доступ к файлу
Создатель	пользователь — создатель файла
Владелец	пользователь — владелец файла
Группа	группа — владелец файла
Флаг «только чтение»	файл предназначен только для чтения
Флаг «скрытый»	файл является скрытым при сканировании каталога
Флаг «системный»	файл является компонентой ОС
Флаг «архивный»	файл должен архивироваться
Флаг «только исполнение»	для файла допустима только операция исполнения
Флаг «запрет удаления»	для файла запрещено удаление
Флаг «запрет переименования»	для файла запрещено переименование
Флаг «временный»	является временным-удаляется после использования
Флаг блокировки	доступ к файлу блокирован
Дата и время создания	
Дата и время последнего доступа	
Дата и время последнего изменения	
Размер	

Операции над файлами-1

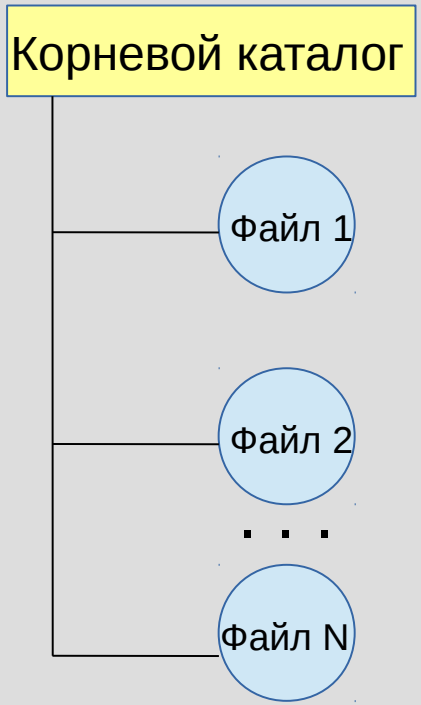
- **Create (Создать).** Создает файл без данных. Цель вызова состоит в объявлении о появлении нового файла и установке ряда атрибутов;
- **Delete (Удалить).** Когда файл больше не нужен, его нужно удалить, чтобы освободить дисковое пространство. Именно для этого и предназначен этот системный вызов;
- **Open (Открыть).** Перед использованием файла процесс должен его открыть. Цель системного вызова open — дать возможность системе извлечь и поместить в оперативную память атрибуты и перечень адресов на диске, чтобы ускорить доступ к ним при последующих вызовах;
- **Close (Закрыть).** После завершения всех обращений к файлу потребность в его атрибутах и адресах на диске уже отпадает, поэтому файл должен быть закрыт, чтобы освободить место во внутренней таблице. Многие системы устанавливают максимальное количество открытых процессами файлов, определяя смысл существования этого вызова. Информация на диск пишется блоками, и закрытие файла вынуждает к записи последнего блока файла, даже если этот блок и не заполнен;
- **Read (Произвести чтение).** Считывание данных из файла. Как правило, байты поступают с текущей позиции. Вызывающий процесс должен указать объем необходимых данных и предоставить буфер для их размещения;
- **Write (Произвести запись).** Запись данных в файл, как правило, с текущей позиции. Если эта позиция находится в конце файла, то его размер увеличивается. Если текущая позиция находится где-то в середине файла, то новые данные пишутся поверх существующих, которые утрачиваются навсегда;

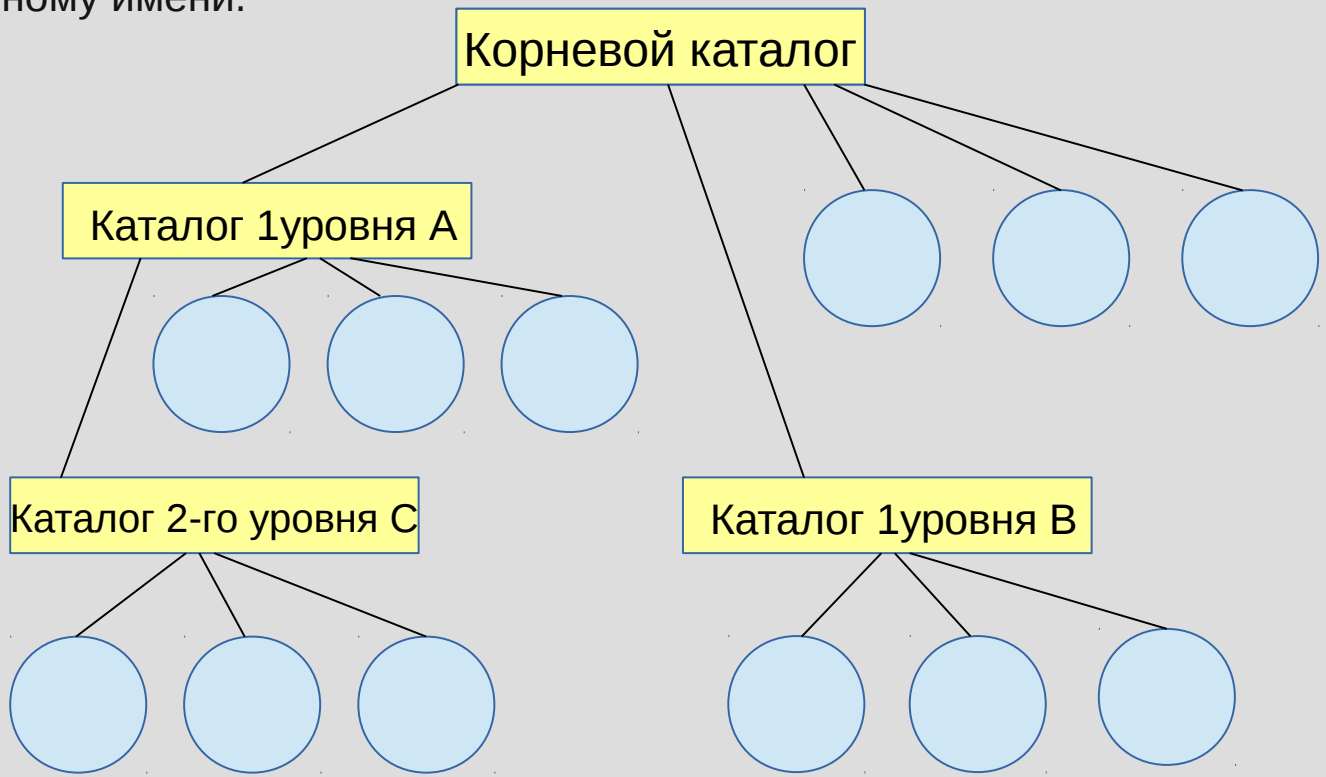
Операции над файлами-2

- **Append (Добавить).** Этот вызов является усеченной формой системного вызова write. Он может лишь добавить данные в конец файла. Как правило, у систем, предоставляющих минимальный набор системных вызовов, вызов append отсутствует, но многие системы предоставляют множество способов получения того же результата, и иногда в этих системах присутствует вызов append;
- **Seek (Найти).** При работе с файлами произвольного доступа нужен способ указания места, с которого берутся данные. Одним из общепринятых подходов является применение системного вызова seek, который перемещает указатель файла к определенной позиции в файле. После завершения этого вызова данные могут считываться или записываться с этой позиции;
- **Get attributes (Получить атрибуты).** Процессу для работы зачастую необходимо считать атрибуты файла. К примеру, имеющаяся в UNIX программа make обычно используется для управления проектами разработки программного обеспечения, состоящими из множества сходных файлов. При вызове программа make проверяет время внесения последних изменений всех исходных и объектных файлов и для обновления проекта обходится компиляцией лишь минимально необходимого количества файлов. Для этого ей необходимо просмотреть атрибуты файлов, а именно время внесения последних изменений;
- **Set attributes (Установить атрибуты).** Значения некоторых атрибутов могут устанавливаться пользователем и изменяться после того, как файл был создан. Такую возможность дает именно этот системный вызов. Характерным примером может послужить информация о режиме защиты. Под эту же категорию подпадает большинство флагов;
- **Rename (Переименовать).** Нередко пользователю требуется изменить имя существующего файла. Этот системный вызов помогает решить эту задачу. Необходимость в нем возникает не всегда, поскольку файл может быть просто скопирован в новый файл с новым именем, а старый файл затем может быть удален.

Каталоги

Все современные файловые системы поддерживают многоуровневое именование файлов за счет наличия во внешней памяти дополнительных файлов со специальной структурой - каталогов (или директорий). Каждый каталог, по сути, является файлом определённой структуры, где каждая запись хранит информацию об одном файле или другом каталоге, входящем в данный. Возможен вариант единственного корневого каталога, хранящего записи обо всех файлах (CDC6600, CP/M) — имеем «плоскую файловую систему». В случае вложенных каталогов имеем древовидную структуру. Обращение-по полному имени.





Операции над каталогами-1

- **Create (Создать каталог).** Каталог создается пустым, за исключением точки и двойной точки, которые система помещает в него автоматически (или в некоторых случаях при помощи программы mkdir).
- **Delete (Удалить каталог).** Удалить можно только пустой каталог. Каталог, содержащий только точку и двойную точку, рассматривается как пустой, поскольку они не могут быть удалены.
- **Opendir (Открыть каталог).** Каталоги могут быть прочитаны. К примеру, для вывода имен всех файлов, содержащихся в каталоге, программа ls открывает каталог для чтения имен всех содержащихся в нем файлов. Перед тем как каталог может быть прочитан, он должен быть открыт по аналогии с открытием и чтением файла.
- **Closedir (Закрыть каталог).** Когда каталог прочитан, он должен быть закрыт, чтобы освободить пространство во внутренних таблицах системы.

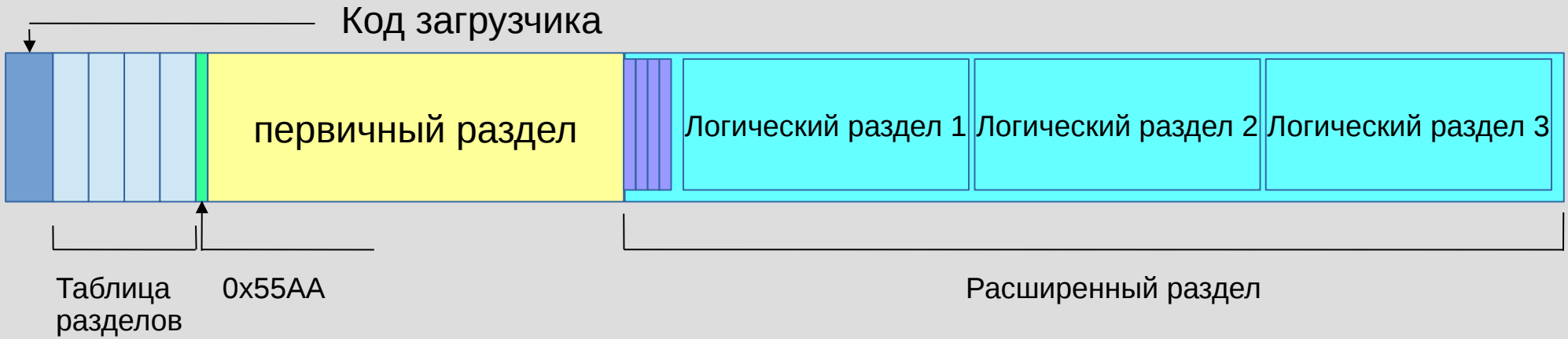
Операции над каталогами-2

- **Readdir (Прочитать каталог).** Этот вызов возвращает следующую запись из открытого каталога. Раньше каталоги можно было читать с помощью обычного системного вызова read, но недостаток такого подхода заключался в том, что программист вынужден был работать с внутренней структурой каталогов, о которой он должен был знать заранее. В отличие от этого, readdir всегда возвращает одну запись в стандартном формате независимо от того, какая из возможных структур каталогов используется.
- **Rename (Переименовать каталог).** Во многих отношениях каталоги подобны файлам и могут быть переименованы точно так же, как и файлы.
- **Link (Привязать).** Привязка представляет собой технологию, позволяющую файлу появляться более чем в одном каталоге. В этом системном вызове указываются существующий файл и новое имя файла в некотором существующем каталоге и создается привязка существующего файла к указанному каталогу с указанным новым именем. Таким образом, один и тот же файл может появиться в нескольких каталогах, возможно, под разными именами. Подобная привязка, увеличивающая показания файлового счетчика i-узла (предназначенного для отслеживания количества записей каталогов, в которых фигурирует файл), иногда называется жесткой связью, или жесткой ссылкой (hard link).
- **Unlink (Отвязать).** Удалить запись каталога. Если отвязываемый файл присутствует только в одном каталоге (что чаще всего и бывает), то этот вызов удалит его из файловой системы. Если он фигурирует в нескольких каталогах, то он будет удален из каталога, который указан в имени файла. Все остальные записи останутся. Фактически системным вызовом для удаления файлов в UNIX (как ранее уже было рассмотрено) является unlink.

Еще одним вариантом связи файлов является символическая ссылка (soft link, symbolic link)

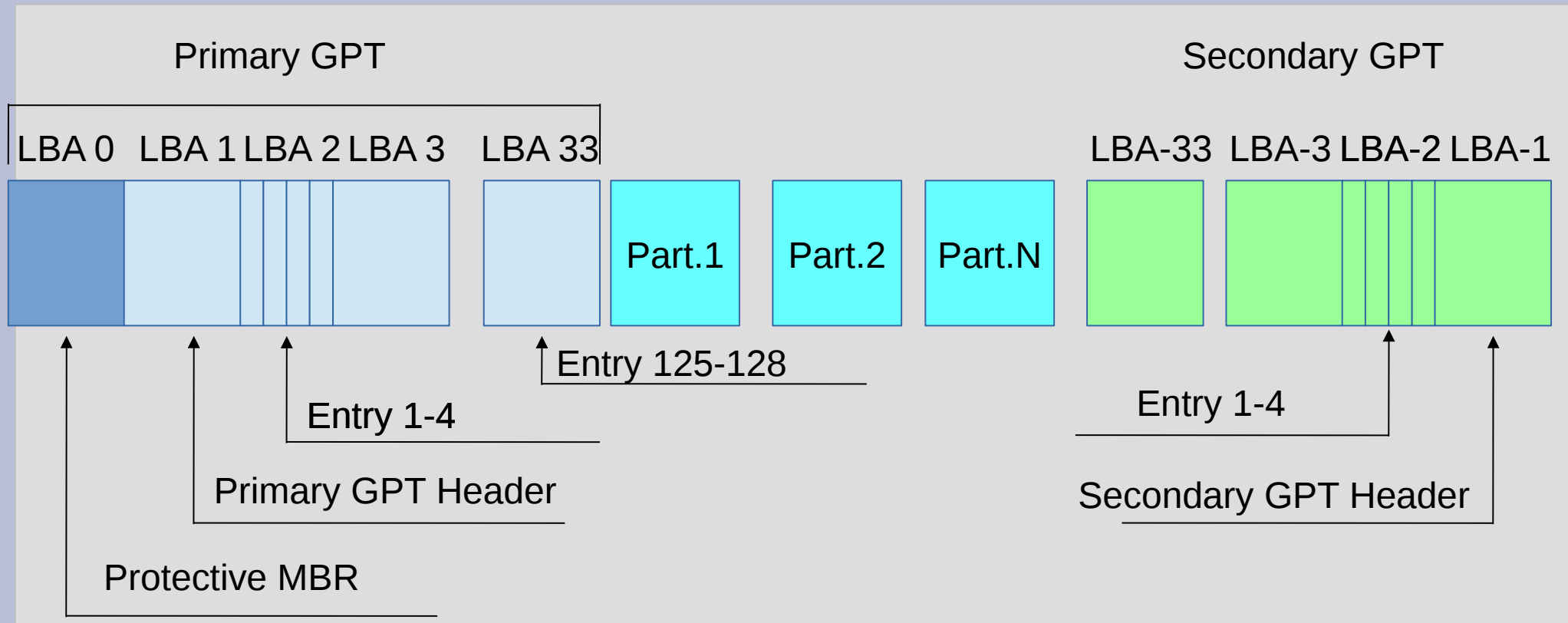
Реализация файловых систем: таблицы разделов- MBR

Диск делится на разделы, разделы образуют тома. Как именно диск делится на разделы определяется таблицей разделов. Таблицы разделов бывают двух типов: Master Boot Record (MBR) и GUID Partition Table (таблица разделов на основе глобального уникального идентификатора) .

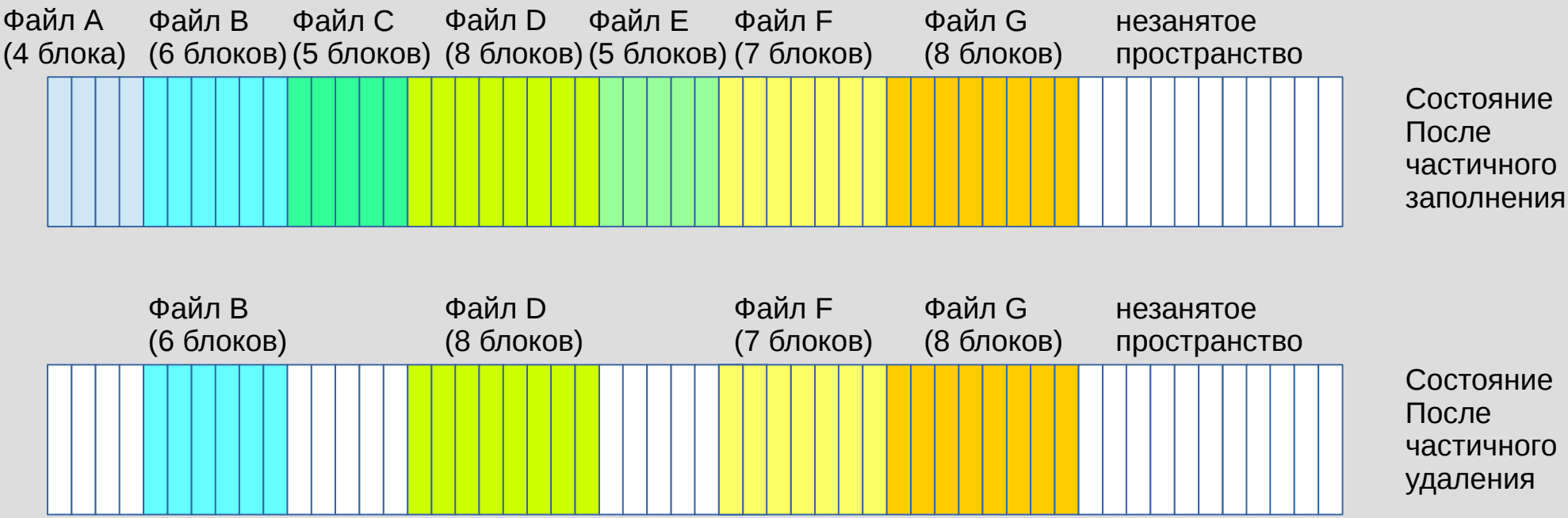


Достоинство MBR — простота организации, широкая распространённость, наличие большого количества инструментов для работы с MBR;
Недостатки: Размеры разделов ограничены 2 ТВ;
количество разделов ограничивается четырьмя; (обходится)

Реализация файловых систем: таблицы разделов- GPT



Организация файлов: непрерывное размещение блоков



Достоинствами подхода являются: простота реализации — надо фиксировать начальный адрес первого блока и число блоков; высокая скорость доступа — не надо позиционироваться на следующий блок накопителя.
Существенный недостаток: по истечение некоторого времени (выполнения большого количества операций записи и удаления) — имеет место сильная фрагментация дискового пространства.

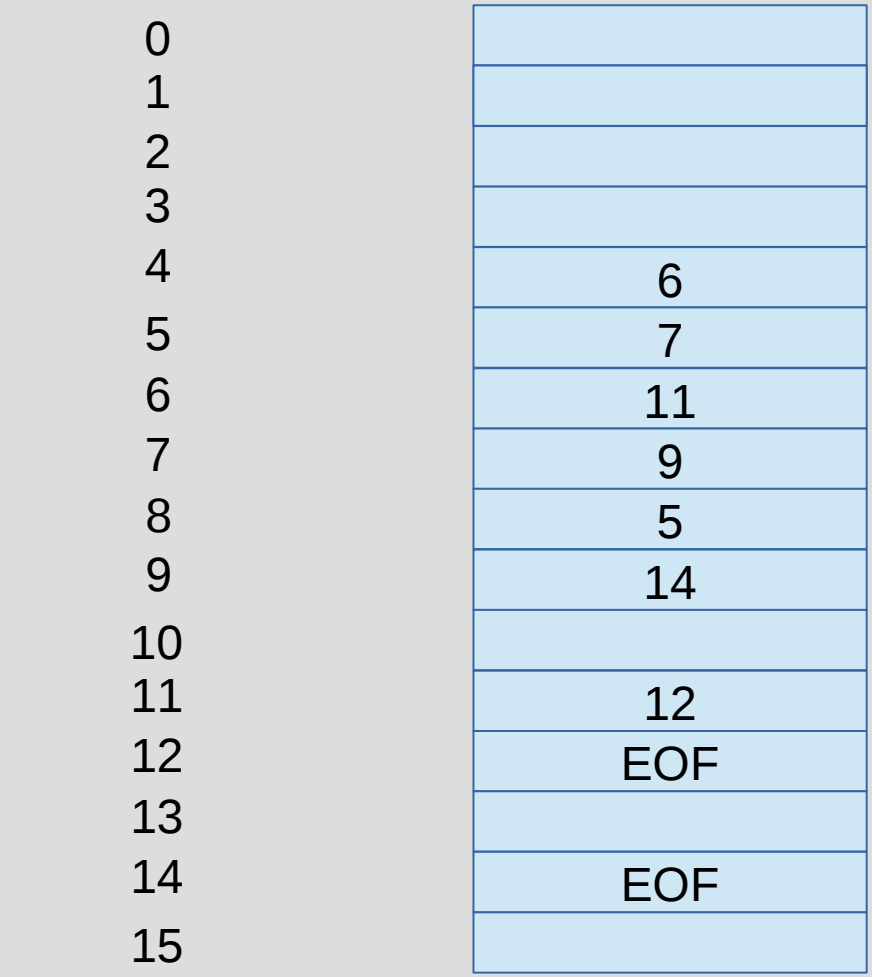
Организация файлов:размещение с использованием связных списков

Файл A занимает блоки 0,1,3;
Файл B занимает блоки 2,4,7,11;
Файл C занимает блоки 5,6,10

Достоинство: отсутствует проблема записи файла размером больше имеющегося непрерывного блока — даже при сильной фрагментации дискового пространства.

Недостатки: замедление скорости доступа вследствие дополнительных операций позиционирования; снижается надежность хранения — сбойный блок приводит к разрыву цепочки указателей; дополнительные затраты дискового пространства в каждом блоке для хранения указателей, кроме того, размер каждого блока теперь не кратен степени 2

Организация файлов: размещение с использованием связных списков и таблицы, хранимой в памяти



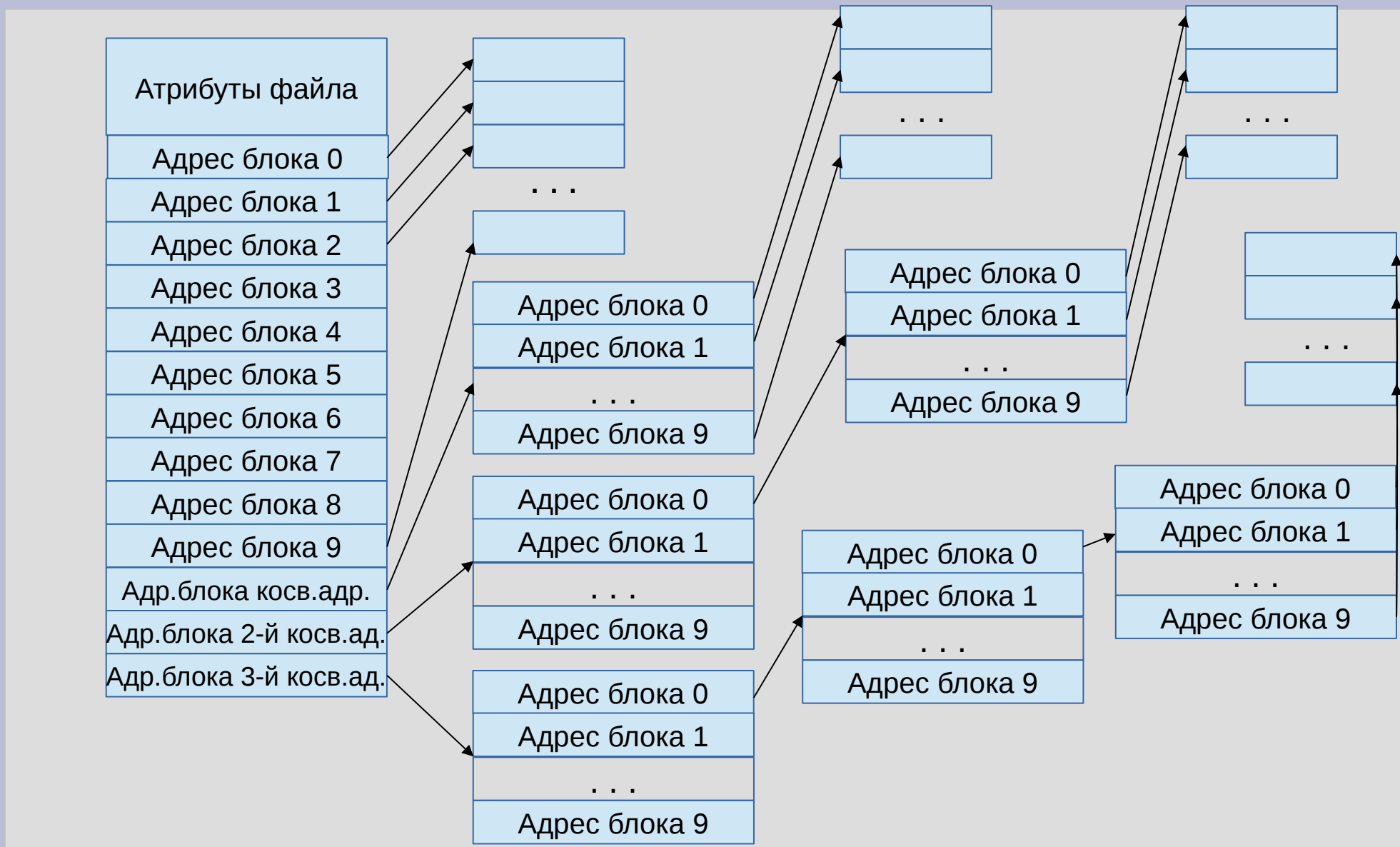
Файл A занимает блоки: 4,6,11,12

Файл B занимает блоки: 8,5,7,9,14

Записи в каталоге содержат ссылки на первый блок. Таблица хранится в памяти.

Достоинства: для данных, в отличие от предыдущего случая, доступен весь блок; реализуется прямой доступ к блокам файла; доступ осуществляется быстро(таблица в памяти).
Основной недостаток: таблица полностью хранится в памяти и занимает много места

Организация файлов: размещение с использованием index-узлов (i-узлов)



Реализация каталогов

При открытии файла используется имя пути, чтобы найти запись в каталоге. Запись в каталоге указывает на адреса блоков диска.

В зависимости от системы это может быть:

- дисковый адрес всего файла (для непрерывных файлов)
- номер первого блока (связные списки)
- номер i-узла

Одна из основных задач каталоговой системы - преобразование ASCII-имени в информацию, необходимую для нахождения данных. Также она хранит атрибуты файлов.

Варианты хранения атрибутов:

- В каталоговой записи
- В i-узлах

Организация дискового пространства

При организации дискового пространства решается несколько вопросов, такие, например, как выбор размера дискового блока и учет свободных блоков.

При выборе размера блока возможны две крайности:

- большие блоки - например, 1Мбайт - файл размером в 1 байт займет целый блок в 1Мбайт — неэффективное использование дискового пространства;
- маленькие блоки - чтение файла состоящего из большого числа блоков будет медленным.

На сегодняшний день размер дискового блока составляет от 1 до 4 Кбайт (*nix);
от 512 байт до 64 Кбайт (обычно в районе 2 Кбайт) (Windows).

- Учет свободных блоков обычно осуществляется двумя способами:

- связный список свободных блоков диска;
- битовая карта, позволяющая определить свободные блоки диска.

Надежность файловой системы: резервное копирование

Надежность файловой системы реализуется следующими способами:

1. Резервное копирование

Случаи, для которых необходимо резервное копирование:

- Аварийные ситуации, приводящие к потере данных на диске;
- Случайное удаление или программная порча файлов.

Основные принципы создания резервных копий:

- Создавать несколько копий - ежедневные, еженедельные, ежемесячные, ежеквартальные;
- Как правило, необходимо сохранять не весь диск, а только выборочные каталоги;
- Применять инкрементные резервные копии - сохраняются только измененные файлы;
- Сжимать резервные копии для экономии места;
- Фиксировать систему при создании резервной копии, чтобы во время резервирования система не менялась;
- Хранить резервные копии в защищенном месте, не доступном для посторонних.

Существует две стратегии:

Физическая архивация - поблочное копирование диска (копируются блоки, а не файлы)

Недостатки:

- копирование пустых блоков;
- проблемы с дефектными блоками;
- невозможно применять инкрементное копирование;
- невозможно копировать отдельные каталоги и файлы.

Преимущества:

- высокая скорость копирования;
- простота реализации.

Логическая архивация - работает с файлами и каталогами. Применяется чаще физической.

Надежность файловой системы: обеспечение непротиворечивости

Если в системе произойдет сбой, прежде чем модифицированный блок будет записан, файловая система может попасть в противоречивое состояние. Особенно, если это блок i-узла, каталога или списка свободных блоков.

В большинстве файловых систем есть специальная программа, проверяющая непротиворечивость системы. В UNIX это утилита fsck, в Windows - scandisk.

Если произошел сбой, то во время загрузки ОС они проверяют файловую систему (если файловая система журналируемая, такая проверка не требуется).

Журналируемая файловая система - операции выполняются в виде транзакций, если транзакция не завершена, то во время загрузки происходит откат.

Существует два типа проверки на непротиворечивость системы:
проверка блоков - проверяется дублирование блоков в файле или в списке свободных блоков. Далее проверяется, нет ли блока файла, который еще присутствует в списке свободных блоков. Если блока нет в занятых и в свободных, то блок считается потерянным (фактически, уменьшается место на диске), такие блоки добавляются к свободным. Также блок может оказаться в двух файлах. Приемлемым действием программы проверки файловой системы в этом случае станет выделение свободного блока, копирование в него содержимого дублированного блока и вставка копии в один из файлов. По крайней мере сохраняется структура ФС.

Проверка файлов - в первую очередь проверяется каталоговая структура. Файл может оказаться либо в нескольких каталогах, либо не в одном каталоге (уменьшается место на диске).

Производительность файловой системы

Кэширование

Блочный кэш (буферный кэш) - набор блоков хранящиеся в памяти, но логически принадлежащих диску. Перехватываются все запросы чтения к диску, и проверяется наличие требуемых блоков в кэше.Ситуация схожа со страничной организацией памяти, следовательно,можно применять те же алгоритмы. Кроме того, необходимо, чтобы измененные блоки периодически записывались на диск. В UNIX это выполняет демон update (вызывая системный вызов sync). В MS-DOS и Windows модифицированные блоки сразу записываются на диск (сквозное кэширование).

Опережающее чтение блока

Если файл считывается последовательно, то после получения к-блока, можно считать блок к+1 (если его нет в памяти). Подобный приём существенно увеличивает быстродействие системы. При произвольном обращении к файлу опережающее чтение бесполезно, более того, за счет лишних операций чтения производительность системы может даже деградировать.

Снижение времени перемещения блока головок

Если при записи группировать блоки файлов, так, чтобы они размещались рядом (соседние сектора или дорожки), то перемещение головок будет меньше.

Выполнение процедур дефрагментации дискового пространства

Организация файловых систем:файловая система ISO9660

Стандарт принят в 1988 г.

По стандарту диски могут быть разбиты на логические разделы.

Блоки записываются последовательно, по спирали; сектора по 2352 байта.

Порядок записи информации:

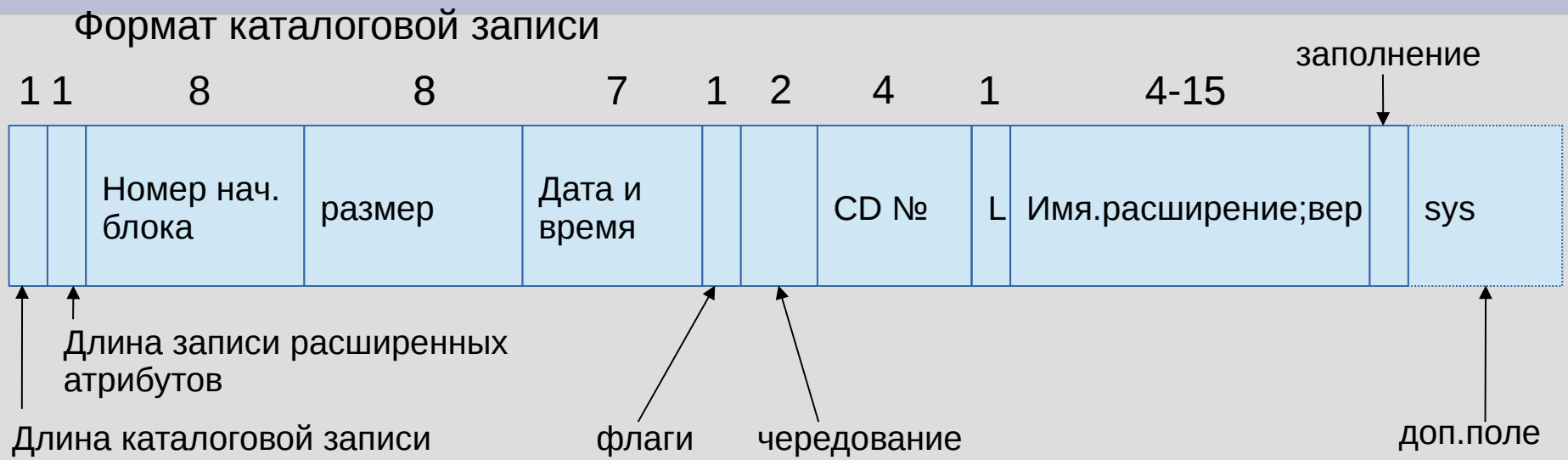
1. Каждый CD-ROM начинается с 16 блоков (неопределенных ISO 9660), эта область может быть использована для размещения загрузчика ОС или для других целей.

2. Далее следует один блок основного описателя тома - хранит общую информацию о CD-ROM, в нее входит:

- идентификатор системы (32байта);
- идентификатор тома (32байта);
- идентификатор издателя (128байт);
- идентификатор лица, подготовившего данные (128байт);
- имена трех файлов, которые могут содержать краткий обзор, авторские права и библиографическая информация;
- ключевые поля: размер логического блока (как правило 2048, но могут быть 4096, 8192 и т.д.); количество блоков; дата создания; дата окончания срока службы диска.
- описатель корневого каталога (номер блока, содержащего каталог).

3. Могут быть дополнительные описатели тома, подобные основному.

Организация файловых систем:файловая система ISO9660-1



L - длина имени файла в байтах;
Имя файла - 8 символов, 3 символа расширения (из-за совместимости с MS-DOS).
Имя файла может встречаться несколько раз, но с разными номерами версий.
Sys - поле System use (используется различными ОС для своих расширений).
Порядок каталоговых записей:

- Описатель самого каталога (аналог ".");
- Ссылка на родительский каталог (аналог "..");
- Остальные записи (записи файлов) в алфавитном порядке;

Количество каталоговых записей не ограничено, но ограничено количество вложенности каталогов - 8.
В стандарте ISO 9660 определены три уровня ограничений:

- имена файлов = 8-3;
- имена каталогов 8 символов, имена каталогов без расширений;
- глубина вложенности каталогов ограничена восемью;
- файлы должны быть непрерывными.

2. имена файлов и каталогов до 31 символа
3. файлы могут быть не непрерывными, состоять из разделов

Организация файловых систем:файловая система ISO9660-расширения

Rock Ridge расширения для UNIX

Это расширение было создано, чтобы файловая система UNIX была представлена на CD-ROM.
Расширения используют поле System use и содержат следующие поля:
PX - атрибуты POSIX (стандартные биты rwxrwxrwx, (чтение, запись, запуск) (владелец, группа, все));
PN - старший и младший номер устройств (чтобы можно было записать каталог /dev, который содержит устройства);
SL - символьная связь;
NM - альтернативное имя, позволяет использовать произвольные имена, без ограничений ISO9660;
CL - расположение дочернего каталога (чтобы обойти ограничение на вложенность каталогов);
PL — расположение родительского каталога (чтобы обойти ограничение на вложенность каталогов);
RE - перераспределение (чтобы обойти ограничение на вложенность каталогов);
TF - временные штампы (время создания, последнее изменение , последний доступ).

Joliet расширения для Windows

Это расширение было создано, чтобы файловая система ОС Windows 95 была представлена на CD-ROM.
Расширения используют поле System use и содержат следующие поля:

- Длинные имена файлов (до 64 символов);
- Набор символов Unicode (поддержка различных языков);
- Преодоление ограничений на вложенность каталогов;
- Имена каталогов с расширениями;

Romeo расширения для Windows

Стандарт Romeo предоставляет другую возможность записи файлов с длинными именами на компакт-диск. Длина имени может составлять 128 символов, использование кодировки Unicode не предусмотрено. Альтернативные имена в этом стандарте не создаются, поэтому программы MS-DOS не смогут прочитать файлы с такого диска. Стандарт Romeo выбирается только в том случае, если диск предназначен для чтения приложениями Windows 95 и Windows NT.

HFS(Hierarchical File System) расширения для Macintosh

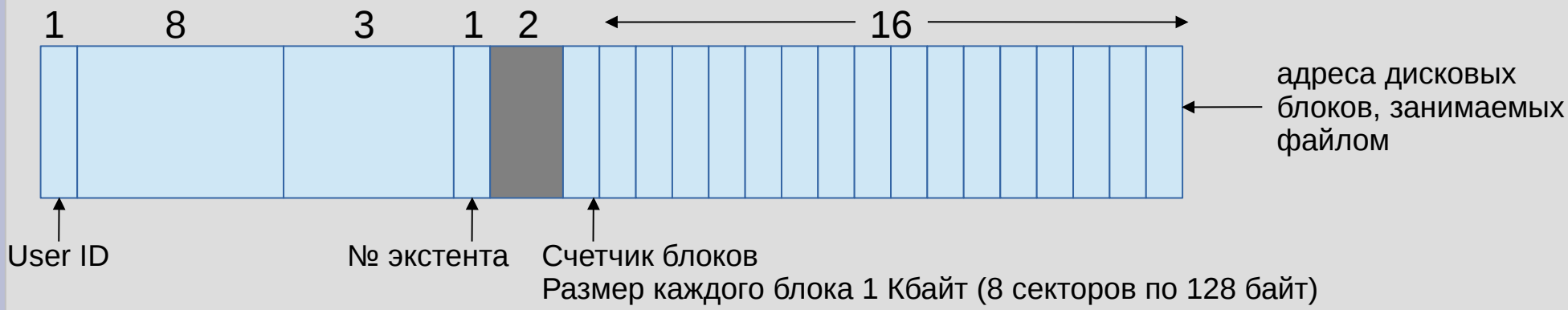
Иерархическая файловая система компьютеров Macintosh, не совместима ни с какими другими файловыми системами.

Файловая система UDF (Universal Disk Format)

Изначально созданная для DVD, с версии 1.50 добавили поддержку CD-RW и CD-R.
Эта ФС позволяет дописывать диски, а также поддерживает большие размеры файлов и длинные имена файлов.

Организация файловых систем:файловая CP/M

В файловой системе только один каталог, с фиксированными записями по 32 байта. Имена файлов - 8+3 символов верхнего регистра. После каждой перезагрузки рассчитывается битовый массив занятых и свободных блоков. Массив находится постоянно в памяти (для 180-Кбайтного диска - 23 байта массива). После завершения работы он не записывается на диск.



Максимальный размер файла 16Кбайт (16*1Кбайт).
Для файлов размером от 16 до 32 Кбайт можно использовать две записи, до 48 Кбайт - три записи и т.д.
Порядковый номер записи хранится в поле № экстента.
User ID - каждый пользователь мог работать только со своими файлами.

Организация файловых систем:файловая система FAT(-12, -16,-32)

Каталоговые записи фиксированной длины по 32 байта. Имена файлов - 8+3 символов верхнего регистра.

8	3	1	10	2	2	2	4
имя файла	Тип файла	атр	Используется с win98 Для реализации long names	time	date	Адр. 1-го блока	размер

Атрибуты - только для чтения, архивный, системный, скрытый.
Поле время (16 разрядов) разбивается на три подполя:

- секунды - 5бит (2^5=32 поэтому хранятся с точностью до 2-х секунд);
- минуты — 6бит;
- часы — 5бит;

Поле даты (16 разрядов) разбивается на три подполя:

- день — 5бит;
- месяц — 4бита;
- год - 7бит (начинается с 1980г, т.е. максимальный 2107г.);

Теоретически размер файлов может быть до 4Гбайт (32 разряда).
Адреса всех блоки файла в записи не хранятся, а только адрес первого блока. Этот адрес используется в качестве индекса для 64К (для FAT-16) элементов FAT-таблицы, хранящейся в оперативной памяти.
В зависимости от количества блоков на диске в системе MS-DOS применяется три версии файловой системы FAT:

- FAT-12;
- FAT-16;
- FAT-32 - для адреса используются только 28 бит, поэтому правильнее назвать FAT-28;

Размер блока (кластера) должен быть кратным 512 байт.

Организация файловых систем:файловая система FAT-2

FAT-12

В первой версии MS-DOS использовалась FAT-12 с 512 байтовыми кластерами, поэтому максимальный размер раздела мог достигать 2Мбайта (2^12*512байта).

С увеличением ёмкости дисков этого объёма стало не хватать, поэтому был увеличен максимальный размер кластера 1,2 и 4 Кбайта (2^12) (при этом эффективность использования диска падает).
FAT-12 до сих пор применяется для гибких дисков(там, где они сохранились).

FAT-16

16-разрядные указатели адресов кластеров файла. Размеры кластеров 512, 1, 2, 4, 8, 16 и 32Кбайт (2^15). Таблица постоянно занимает в памяти 128 Кбайт.Максимальный размер раздела диска достигает 2Гбайта (2^16*32Кбайта), причём кластер в 32 Кбайта, для файлов со средним размером в 1Кбайт, не эффективен.

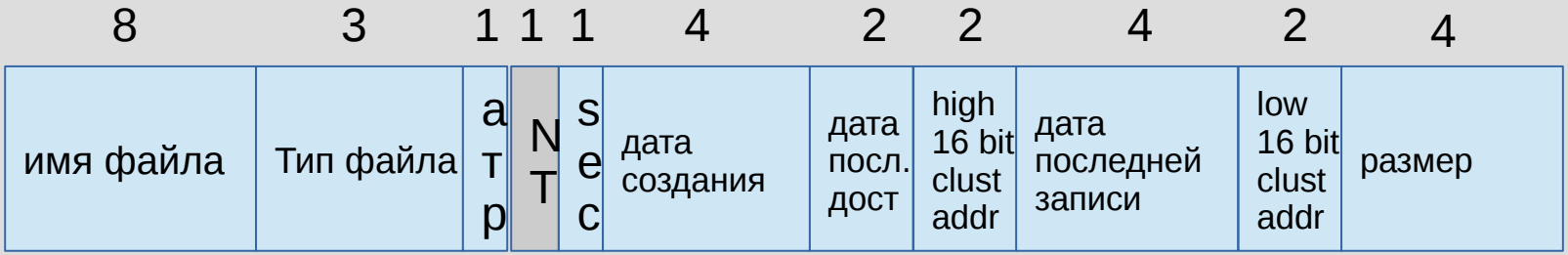
FAT-32

28-разрядные адреса, размеры кластеров 512, 1, 2, 4, 8, 16 и 32Кбайт.
Максимальный размер раздела диска мог бы достигать 2^28*2^15, но здесь уже вступает другое ограничение - 512 байтные сектора адресуются 32-разрядным числом, а это 2^32*2^9, т.е. 2 Тбайта. (см.табл.ниже)

Размер кластера, Кбайт	Fat-12, Мбайт	Fat-16, Мбайт	Fat-32, Тбайт
0,5	2	32	0,13
1	4	64	0,27
2	8	128	0,54
4	16	256	1
8		512	2
16		1024	2
32		2048	2

Организация файловых систем:файловая система FAT(расширение для длинных имён)

Каталоговые записи реорганизованы и используют дополнительные 10 байтов.



Добавлено пять полей:
1. NT - предназначено для совместимости с Windows.
2. Sec - дополнение к старому полю «время», позволяет хранить время с точностью до секунды(было 2 секунды).
3. Дата и время создания файла (Creation time).
4. Дата (но не время) последнего доступа (Last access).
5. Для хранения номера блока выделено еще 2 байта (16 бит), т.к. номера блоков стали 32-разрядные.

Основная надстройка над FAT-32 - это длинные имена файлов.

Для каждого файла присваиваются два имени:
1. Короткое 8+3 для совместимости с MS-DOS.
2. Длинное имя файла, в формате Unicode.

Доступ к файлу может быть получен по любому имени.

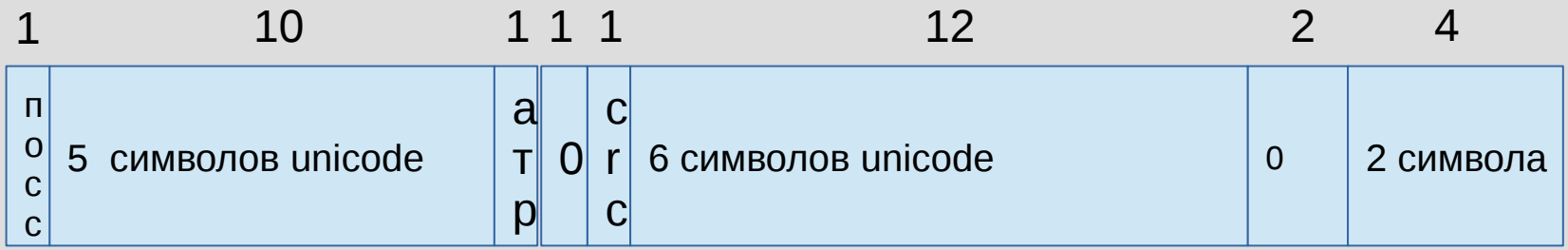
Если файлу дано длинное имя (или используются пробелы), то система выполняет следующие шаги:

- берет первые шесть символов;
- преобразуются в верхний регистр ASCII, удаляются пробелы, лишние точки, некоторые символы преобразуются в "_";
- добавляется суффикс ~1;
- если такое имя есть, то используется суффикс ~2 и т. д.;

Короткие имена хранятся в в обычном дескрипторе файла.

Организация файловых систем: файловая система FAT(расширение для длинных имён-2)

Длинные имена хранятся в дополнительных каталоговых записях, идущих перед основным описателем файла. Каждая такая запись содержит 13 символов формата Unicode (для символа Unicode нужно два байта).



Поле "Атрибуты" позволяет отличить фрагмент длинного имени (значение 0x0F) от дескриптора файла. Старые программы MS-DOS каталоговые записи со значением поля атрибутов 0x0F просто игнорируют.

Последовательность - порядковый номер в последовательности фрагментов.

Длина имени файла ограничена 255 символами.

Контрольная сумма нужна для выявления ошибок, т.к. файл с длинным именем может быть удалён и в соответствующем месте создана новая каталоговая запись, и тогда останутся неудалённые записи, которые "прилипнут" к новому файлу.

Организация файловых систем:файловая система NTFS

Файловая система NTFS была разработана для Windows NT.

Особенности:

- 64-разрядные адреса, т.е. теоретически может поддерживать 2^64*2^16 байт;
- Размеры блока (кластера) от 512байт до 64 Кбайт, для большинства используется 4 Кбайта;
- Поддержка больших файлов;
- Имена файлов ограничены 255 символами Unicode;
- Длина пути ограничивается 32 767 (2^15) символами Unicode;
- Имена чувствительны к регистру, my.txt и MY.TXT это разные файлы (но из-за Win32 API использовать нельзя), это заложено на будущее;
- Журналируемая файловая система, т.е. не попадет в противоречивое состояние после сбоев;
- Контроль доступа к файлам и каталогам;
- Поддержка жестких и символических ссылок;
- Поддержка сжатия и шифрования файлов;
- Поддержка дисковых квот.

Организация файловых систем:файловая система NTFS-MFT

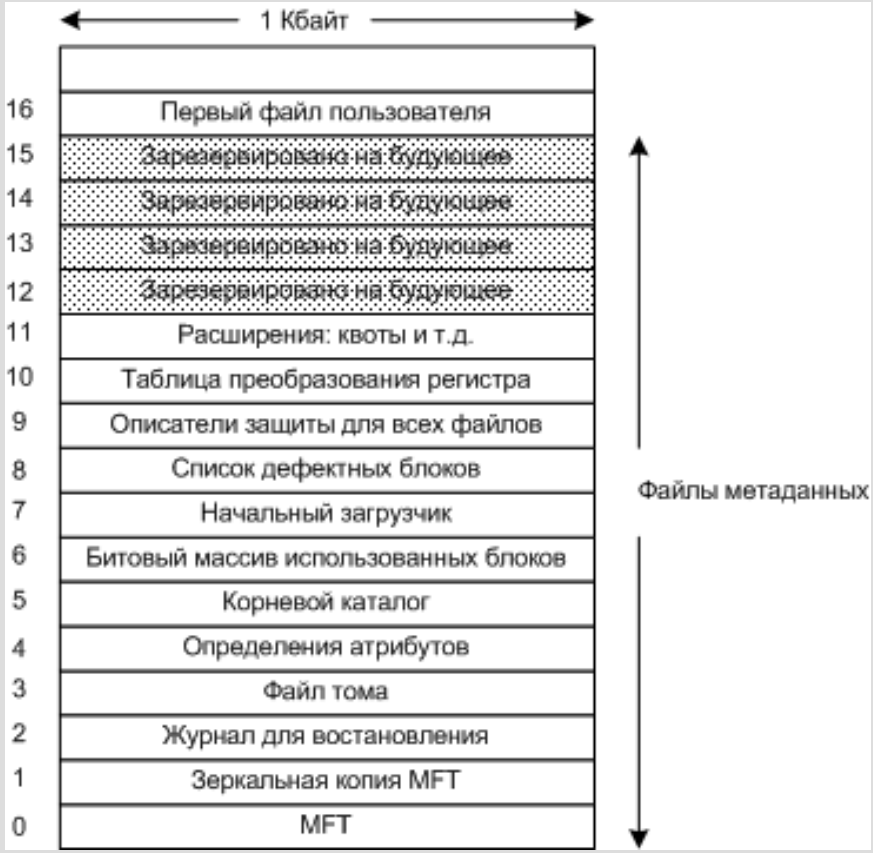
Главная файловая таблица MFT (Master File Table) - главная структура данных в каждом томе, записи фиксированные по 1Кбайт. Каждая запись описывает один каталог или файл. Для больших файлов могут использоваться несколько записей, первая запись называется базовой записью. MFT представляет собой обычный файл (размером до 2^48 записей), который может располагаться в любом месте на диске.

Первые 16 записей MFT зарезервированы для файлов метаданных. Каждая запись описывает нормальный файл, имена этих файлов начинаются с символа "$".

Каждая запись представляет собой последовательность пар (заголовок атрибута, значение).

Некоторые записи метаданных в MFT:

0. Первая запись описывает сам файл MFT, и содержит все блоки файла MFT. Номер первого блока файла MFT содержится в загрузочном блоке.
1. Дубликат файла MFT, резервная копия.
2. Журнал для восстановления, например, перед созданием, удалением каталога делается запись в журнал. Система не попадет в противоречивое состояние после сбоев.
3. Информация о томе (размер, метка и версия).
4. Определяются атрибуты для MFT записей.
6. Битовый массив использованных блоков - для учета свободного места на диске.
7. Указывает на файл начальной загрузки



Организация файловых систем:файловая система NTFS-атрибуты

Примеры атрибутов, используемых в записях MFT:

- стандартная информация - флаги(только чтение, архивный), временные штампы и т.д.;
- имя файла - имя файла в кодировке Unicode;
- имя файла в формате MS-DOS;
- список атрибутов - расположение дополнительных записей MFT(если атрибуты файла не умещаются в основной записи);
- дескриптор безопасности — информация о защите файла: список прав доступа и поле аудита(какие операции должны регистрироваться);
- имя тома;
- версия тома;
- корневой индекс - используется для каталогов;
- размещение индекса - используется для очень больших каталогов(нерезидентные части индексного списка);
- битовая карта MFT — карта использования блоков на томе;
- поток данных утилиты регистрации - используется для шифрования;
- данные - используется для хранения самого файла. За заголовком следует список адресов блоков, определяющий положение файла на диске; если файл очень маленький (несколько сотен байт), то следует сам файл (такой файл называется - непосредственный файл).

Организация файловых систем:файловая система NTFS-нерезидентные файлы

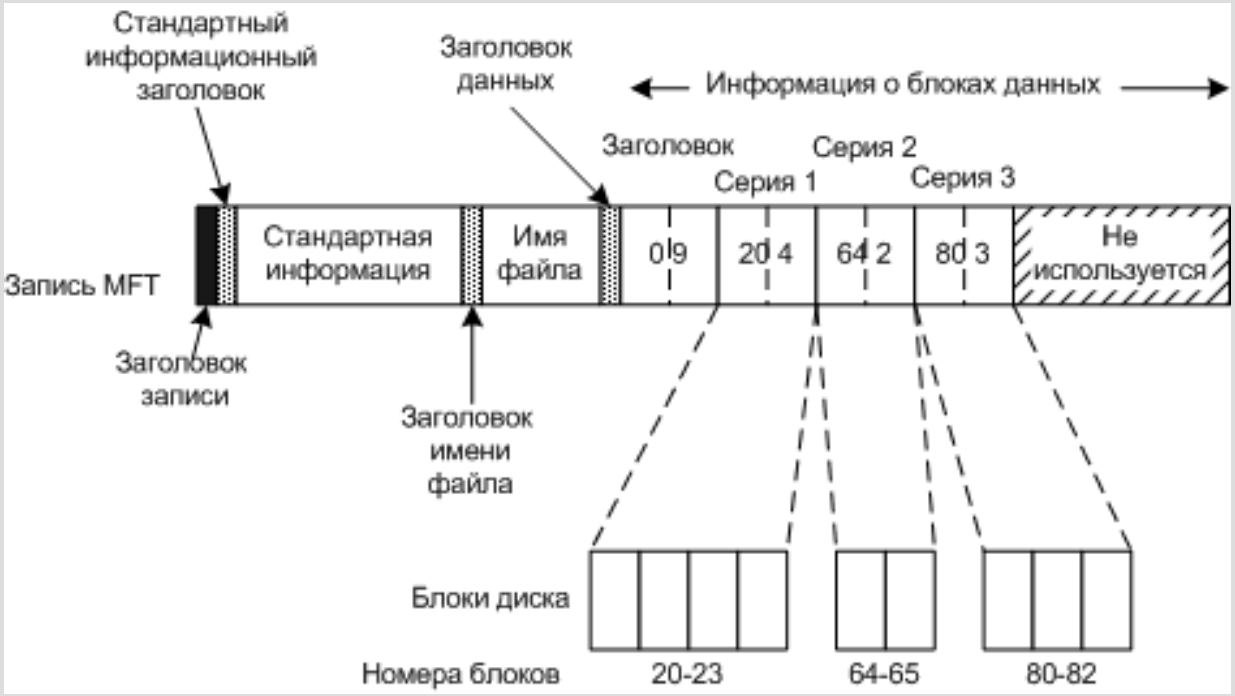
Как привило, все данные файла не помещаются в запись MFT. Блоки файлам назначаются, по возможности, в виде серий последовательных блоков (сегментов файлов). В идеале файл должен быть записан в одну серию (нефрагментированный файл); файл, состоящий из n блоков, может иметь от 1 до n серий.

Заголовок содержит количество блоков (9 блоков).

Каждая серия записывается в виде пары адрес блока - количество блоков (LCN,N) (20-4, 64-2, 80-3).

Каждая пара, при отсутствие сжатия, это два 64-разрядные числа (16 байт на пару).

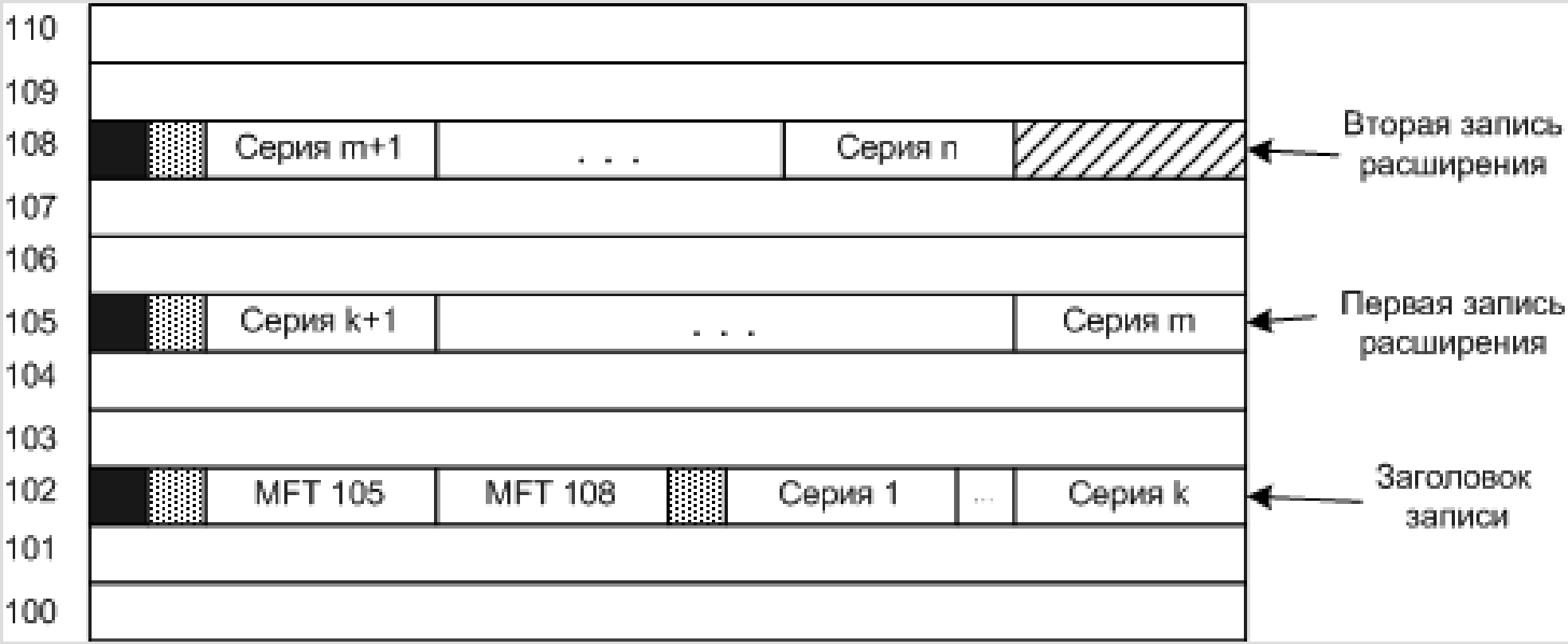
Многие адреса содержат большое количество незначащих нулей; сжатие делается за счет убирания нулей в старших байтах. В результате для пары требуется, чаще всего, 4байта.



Запись MFT для 9-блочного файла, состоящего из трех сегментов (серий). Вся запись помещается в одну запись MFT (файл не сильно фрагментирован).

Организация файловых систем:файловая система NTFS-большие файлы

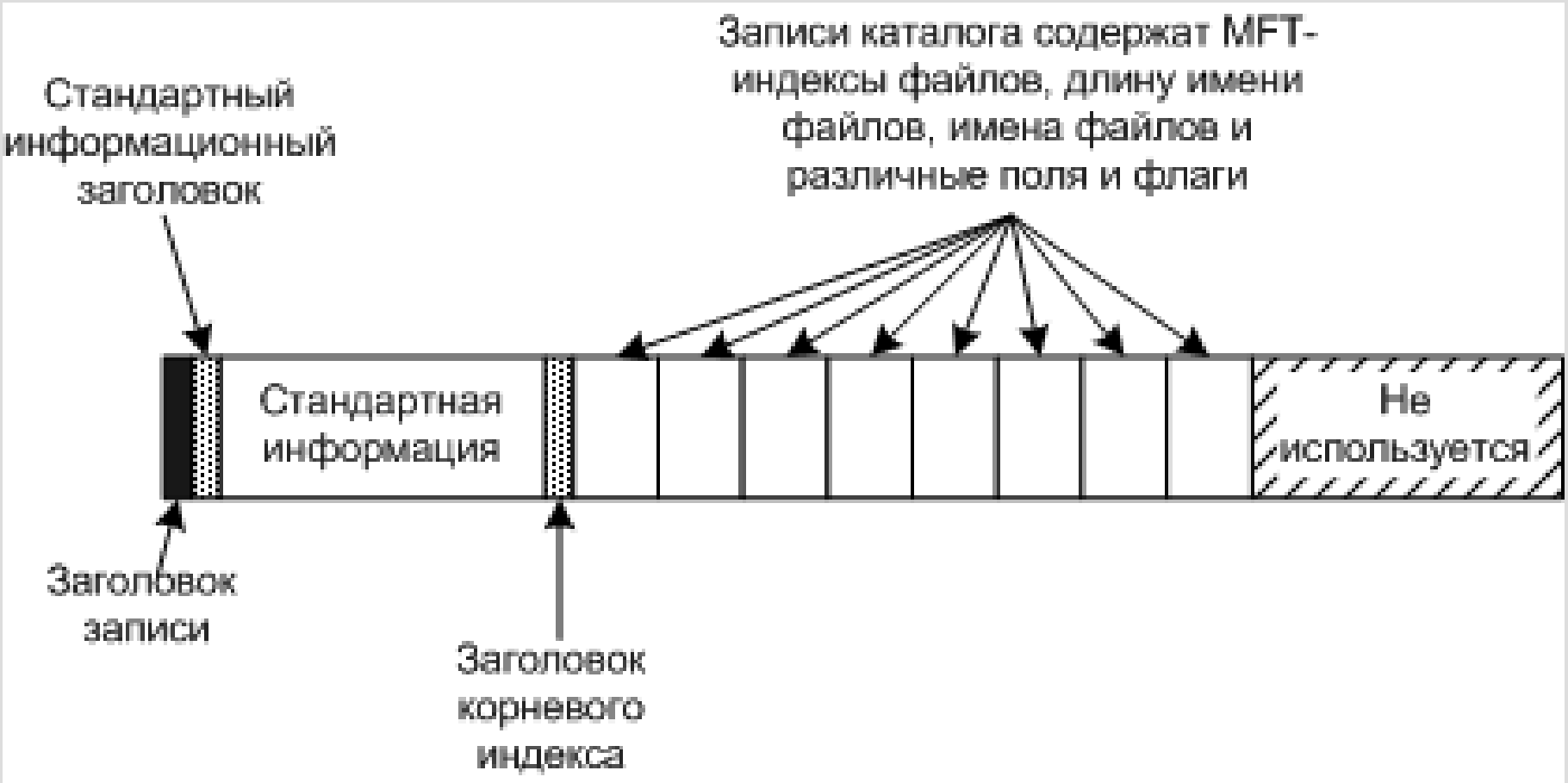
Если файл сильно фрагментирован, требуется несколько записей MFT.



Три записи MFT для сильно фрагментированного файла.
В первой записи указывается индексы дополнительных записей

Может потребоваться очень много индексов MFT, так что индексы не поместятся в запись.
В этом случае список хранится не в MFT, а в файле.

Организация файловых систем:файловая система NTFS-каталоги



Запись MFT для небольшого каталога

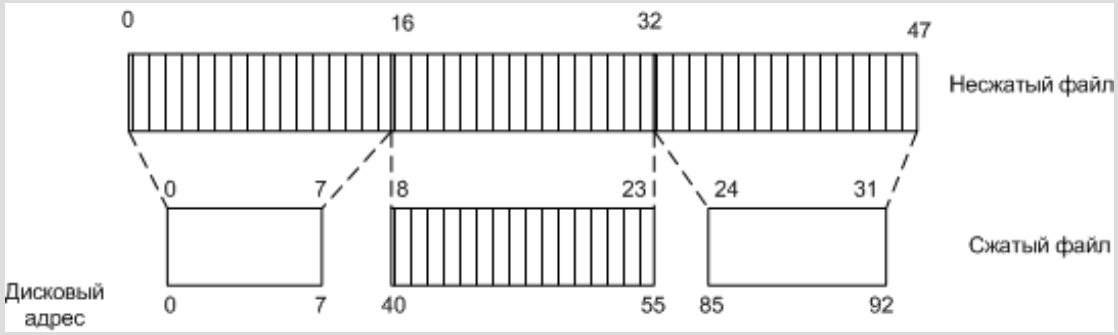
Организация файловых систем:файловая система NTFS-сжатие файлов

Если файл помечен как сжатый, то система автоматически сжимает его при записи, а при чтении происходит его декомпрессия.
Алгоритм работы механизма сжатия следующий:

- рассматриваются первые 16 блоков файла (независимо от сегментов файла);
- к ним применяется алгоритм сжатия;
- если полученные данные можно записать хотя бы в 15 блоков, они записываются в сжатом виде;
- если их можно записать только в 16 блоков, то они записываются в несжатом виде
- алгоритм повторяется для следующих 16 блоков.

Недостатки сжатия:

- сжатие приводит к сильной фрагментации;
- Чтобы прочитать сжатый блок системе придется распаковать весь сегмент. Поэтому сжатие применяют к 16 блокам; если увеличить количество блоков, уменьшится производительность (но возрастет эффективность сжатия).



Пример 48-блочного файла, сжатого до 32 блоков

Запись MFT для файла

Организация файловых систем:файловая система NTFS-шифрование файлов

Если файл помечен как шифрованный, то система автоматически шифрует его при записи, а при чтении происходит дешифровка. Шифрование и дешифрование выполняет не сама NTFS, а специальный драйвер EFS (Encrypting File System). Каждый блок шифруется отдельно.

Организация файловых систем:файловая система Linux - ext2

По организации похожа на UFS, только вместо единого суперблока используется группа блоков

Особенности:

- размер блока 1 Кбайт;
- Размер каждого i-узла 128 байт;
- i-узел содержит 12 прямых и 3 косвенных адресов, длина адреса в i-узле стала 4 байта,

что обеспечивает поддержку размера файла чуть более 16Гбайт.

Особенности работы файловой системы:

- создание новых каталогов распределяется равномерно по группам блоков,

чтобы в каждой группе было одинаковое количество каталогов;

- новые файлы старается создавать в группе, где и находится каталог;
- при увеличении файла система старается новые блоки записывать ближе к старым;
- благодаря этому файловую систему не нужно дефрагментировать, она не способствует фрагментации файлов (в отличии от NTFS), что проверено многолетним использованием.

Организация файловых систем:файловая система Linux — ext3

В отличие от EXT2, EXT3 является журналируемой файловой системой, т.е. не попадет в противоречивое состояние после сбоев. Но она полностью совместима с EXT2.

Драйвер Ext3 хранит полные точные копии модифицируемых блоков (1КБ, 2КБ или 4КБ) в памяти до завершения операции. Это может показаться расточительным. Полные блоки содержат не только изменившиеся данные, но и не модифицированные.

Такой подход называется "физическим журналированием", что отражает использование "физических блоков" как основную единицу ведения журнала. Подход, когда хранятся только изменяемые байты, а не целые блоки, называется "логическим журналированием" (используется XFS). Поскольку ext3 использует "физическое журналирование", журнал в ext3 имеет размер больший, чем в XFS. За счет использования в ext3 полных блоков, как драйвером, так и подсистемой журналирования, нет сложностей, которые возникают при "логическом журналировании".

Типы журналирования поддерживаемые Ext3, которые могут быть активированы из файла /etc/fstab:
data=journal (full data journaling mode) - все новые данные сначала пишутся в журнал и только после этого переносятся на свое постоянное место. В случае аварийного отказа журнал можно повторно перечитать, приведя данные и метаданные в непротиворечивое состояние. Самый медленный, но самый надежный.
data=ordered - записываются изменения только мета-данных файловой системы, но логически metadata и data-блоки группируются в единый модуль, называемый transaction. Перед записью новых метаданных на диск связанные data-блоки записываются первыми. Этот режим журналирования ext3 установлен по умолчанию. При добавлении данных в конец файла режим data=ordered гарантированно обеспечивает целостность (как при full data journaling mode). Однако если данные в файл пишутся поверх существующих, то есть вероятность перемешивания "оригинальных" блоков с модифицированными. Это результат того, что data=ordered не отслеживает записи, при которых новый блок ложится поверх существующего и не вызывает модификации метаданных.
data=writeback (metadata only) - записываются только изменения мета-данных файловой системы. Самый быстрый метод журналирования. С подобным видом журналирования мы имеем дело в файловых системах XFS, JFS и ReiserFS.

Организация файловых систем:файловая система Linux — ext4

Продолжательница ext3, файловая система ext4, также является журналируемой, но в отличие от ext3 она изменяет схему адресации блоков, используемую своими предшественницами, поддерживая за счет этого как более объемные файлы, так и в целом более объемную файловую систему. Основным изменением в ext4 по сравнению с ее предшественниками является использование экстентов. Экстенты представляют собой непрерывные блоки хранилища: например, 128 Мбайт непрерывных 4-килобайтовых блоков в противовес индивидуальным блокам хранения, указываемым в ext2. В отличие от предшественников, в ext4 для каждого блока хранилища операции с метаданными не требуются. Эта схема также сокращает фрагментацию длинных файлов. В результате ext4 может обеспечить более быстрые операции файловой системы и поддержку более объемных файлов и более крупных размеров файловой системы. Например, для размера блока в 1 Кбайт ext4 увеличивает максимальный размер файла с 16 Гбайт до 16 Тбайт, а максимальный размер файловой системы — до 1 Эбайт (экзабайт).

Файловые системы и организация ввода/вывода: литература

1.Таненбаум Э., Бос Х**.** Современные операционные системы. 4-е изд. — СПб.: Питер, 2015. — 1120 с.: ил. — (Серия «Классика computer science»).
2. Основы операционных систем.Курс лекций. Учебное пособие / В.Е.Карпов, К.А.Коньков / Под редакцией В.П. Иванникова. - М.:ИНТУИТ.РУ «Интернет-Университет Информационных Технологий» - 2005, 536 с.
3. В.Г. Олифер, Н.А. Олифер. Сетевые операционные системы.- СПб.:Питер.-2002. - 544 с.