Файловые системы

Файловая система NTFS

Файловая система NTFS

Общие сведения

Краткое описание NTFS

- Разработана для быстрого выполнения стандартных файловых операций типа чтения, записи и поиска.
- Поддерживает улучшенные операции восстановления файловой системы на очень больших жестких дисках.
- Включает возможности безопасности, требуемые для файловых серверов и высококачественных персональных компьютеров в корпоративной среде.

Вопрос

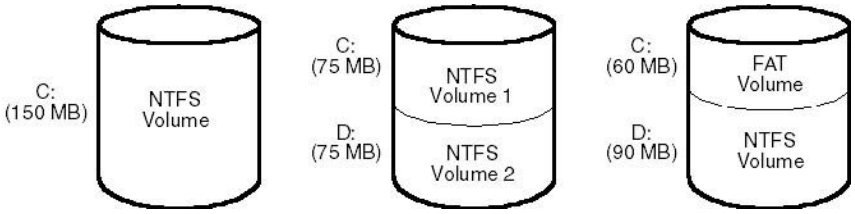
- Вспомните методы физической организации файловой системы.

Физическая организация NTFS

- NTFS использует физическую организацию близкую перечню номеров блоков (кластеров).
- Для увеличения эффективности кластеры выделяются файлам по возможности в виде совокупности последовательных кластеров (так называемых серий, экстентов, пробегов).
- Каждая последовательность кластеров описывается отдельной записью – (стартовый кластер, число кластеров).
- Подобный подход позволяет частично решить проблему фрагментации файлов, т.к. свободное место выделяется не отдельными кластерами, а группами смежных кластеров.

Тома NTFS

- Структура NTFS начинается с тома (volume), который соответствует логическому разделу на диске и создается, когда Вы форматируете диск или часть его для NTFS.
- NTFS обрабатывает каждый том независимо от других.
- NTFS поддерживает тома, состоящие из нескольких разделов.



- NTFS поддерживает размеры кластеров – от 512 байт до 64 Кбайт.

Типы томов NTFS (1)

- *Простой том (simple)*
- *Составной том* (*spanned*) – том, использующий более одного раздела для формирования одного протяженного. Можно использовать разделы с разных дисков для создания набора томов, большего по объему, чем любой имеющийся на компьютере физический диск.
- *Зеркальный том* (*mirrored, RAID 0*) содержит копии своих данных на двух разделах. В случае зеркала запись данных производится на оба раздела, а считывание происходит только с одного. Зеркальный том устойчив к сбою одного диска, в этом случае работает оставшаяся половина.

Типы томов NTFS (2)

- *Чередующийся набор томов* (*stripped, RAID 1*) – том, состоящий из нескольких разделов, по которым равномерными блоками распределены данные. Размер блока данных – 64 Кбайт.
- *Чередующийся набор томов с четностью* (*RAID 5*) – это чередующийся набор с дополнительным блоком данных размером в 64 Кбайт. Дополнительный блок содержит информацию о четности, которую система может использовать при восстановлении данных, расположенных на одном из разделов чередующегося набора, в случае выхода из строя диска, где был расположен раздел.

Внутреннее имя тома

- В разделе HKEY_LOCAL_MACHINE\SYSTEM\MountedDevices системного реестра хранится информация о базовых дисках.
- Внутреннее имя имеет форму \??\Volume{XX-XX-XX-XX}, где X – числа, образующие глобальный уникальный ID (GUID), присвоенный тому операционной системой.
- Для работы с томами существует системная утилита *mountvol* (будет рассмотрена позже).

Размер кластера тома NTFS

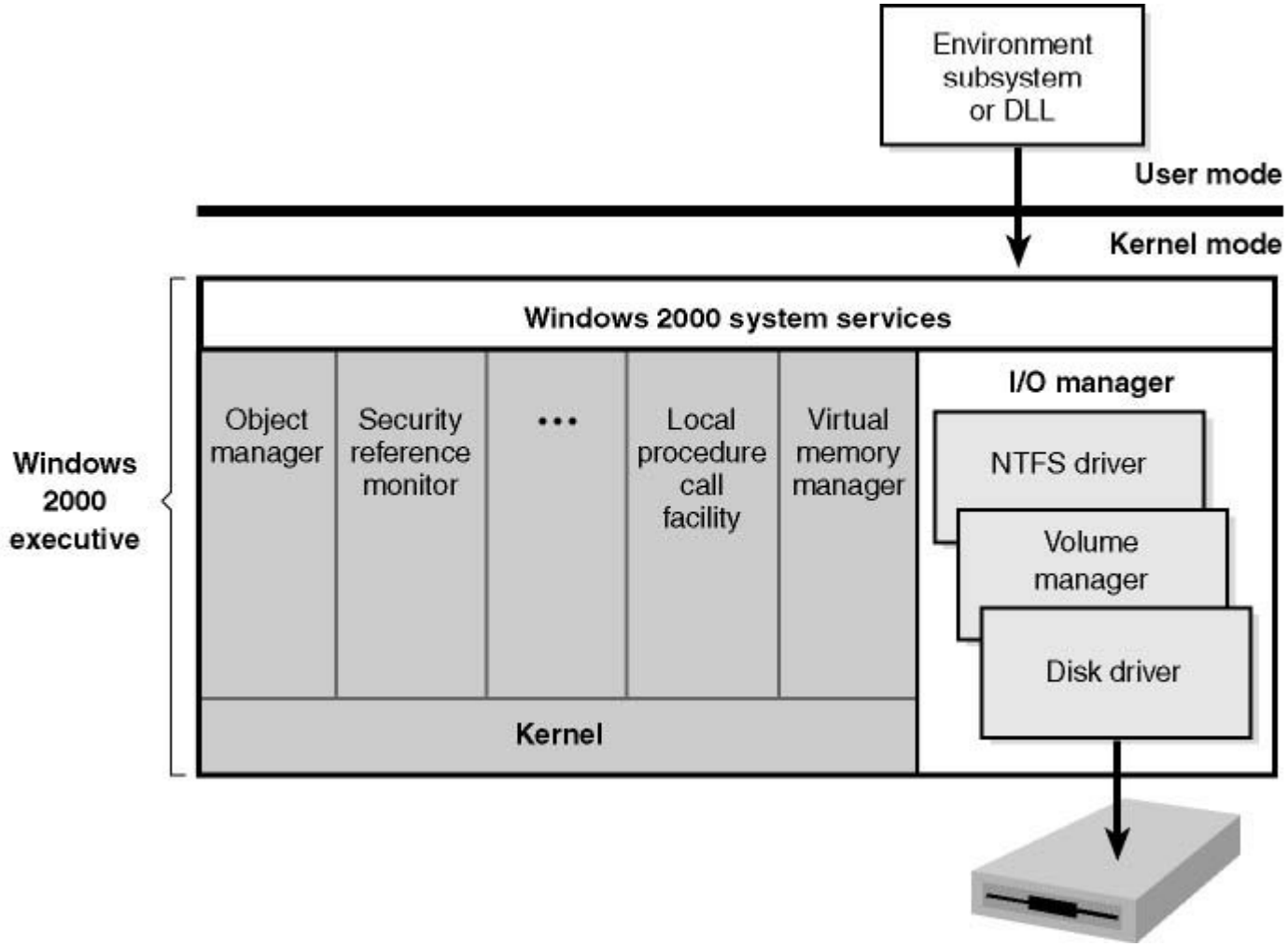
Размер тома	Рекомендуемый размер кластера
512 MB и менее	512 б
513 Мб – 1 Гб	1 Кб
1 Гб – 2 Гб	2 Кб
Более 2 Гб	4 Кб

- Пользователь может определить размер кластера при форматировании тома NTFS /a:<size>, т.е.

  format d: /a:1024 /fs:ntfs

- NTFS-сжатие не поддерживается для кластеров, размер которых больше 4 КБайт.

NTFS и архитектура Windows



Физическая структура NTFS

- Том NTFS условно делится на две части. Первые 12,5% тома отводятся под так называемую MFT зону. Запись данных в эту область невозможна. Остальные 87,5% тома представляют собой пространство для хранения файлов.

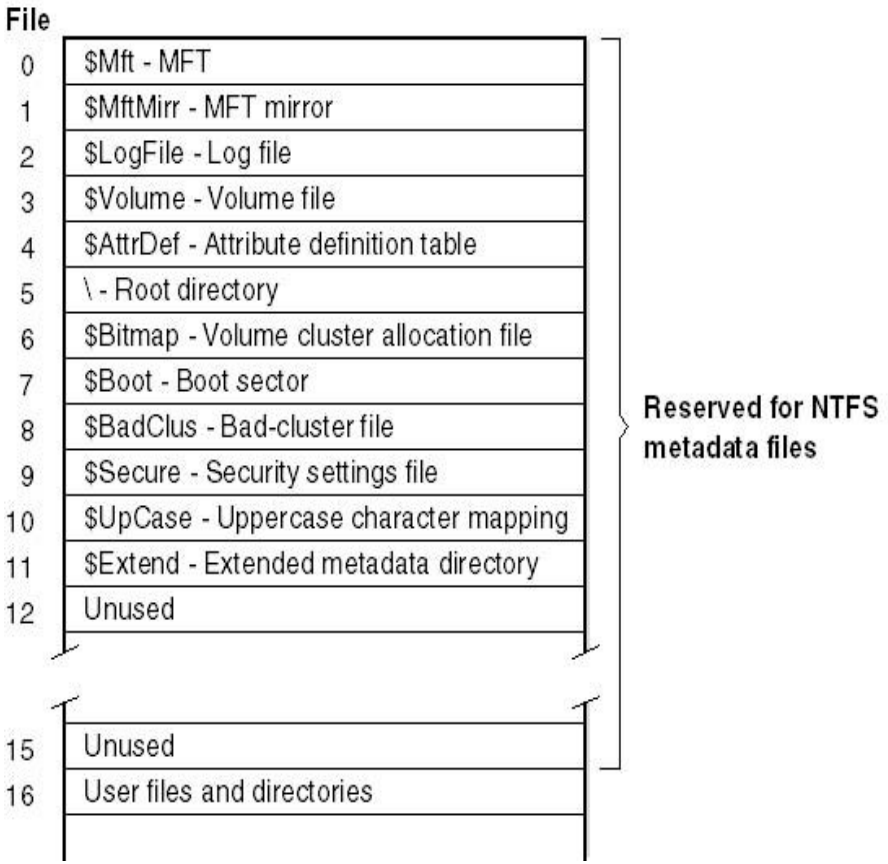
- Свободное место тома, однако, включает в себя всё физически свободное место – незаполненные фрагменты MFT-зоны туда тоже включаются.

Механизм использования MFT-зоны

- Когда файлы уже нельзя записывать в обычное пространство, MFT-зона просто сокращается, освобождая таким образом место для записи файлов.
- При освобождении места в обычной области MFT-зона может снова расширится. При этом не исключена ситуация, когда в этой зоне остались и обычные файлы.
- Метафайл MFT может фрагментироваться, хотя это и нежелательно.

MFT и ее структура

- Первые 16 файлов NTFS (метафайлы) носят служебный характер.
- Метафайлы находятся корневом каталоге NTFS диска – они начинаются с символа имени "$".
- Для метафайлов указан реальный размер – можно узнать, например, сколько ОС тратит на каталогизацию всего



Перечень метафайлов (1)

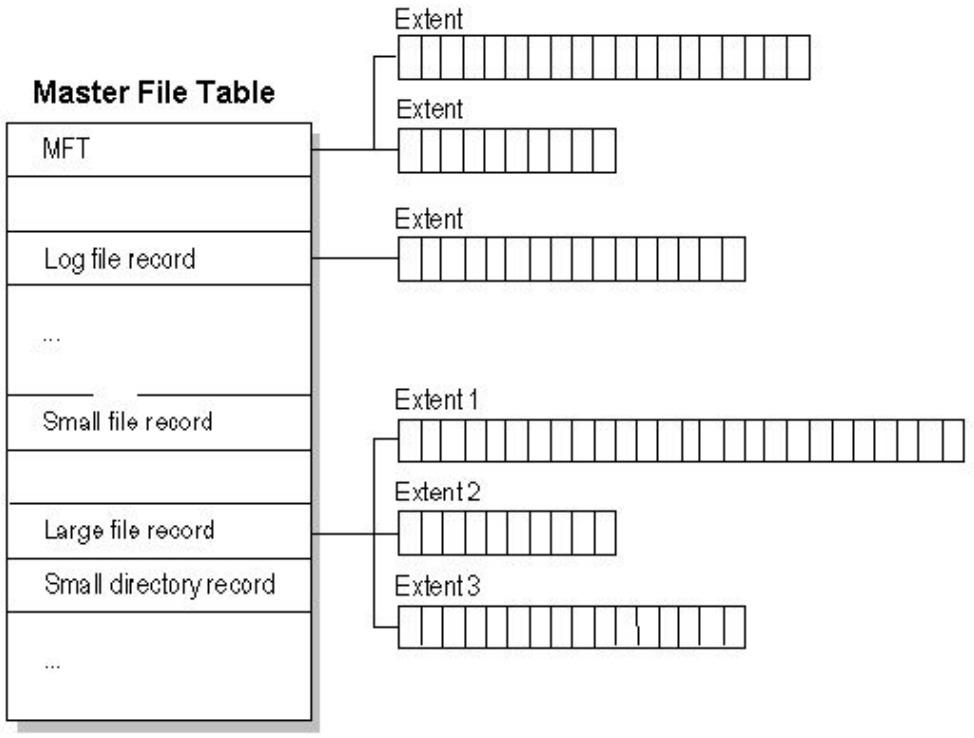
$MFT	список содержимого тома NTFS
$MFTmirr	копия первых 4 записей таблицы MFT
$LogFile	файл поддержки журналирования шагов транзакций
$Volume	служебная информация - метка тома, версия файловой системы, т.д.
$AttrDef	список стандартных атрибутов файлов на томе
$.	корневой каталог
$Bitmap	карта свободного места тома, каждый бит которой соответствует одному кластеру тома и указывает его состояние (свободен или занят)

Перечень метафайлов (2)

$Boot	Загрузочный сектор раздела NTFS
$BadClus	Список всех плохих кластеров тома. Кластер считается плохим, если в нем есть один плохой сектор
$Secure	База данных атрибутов безопасности. Применяется в NTFS начиная с версии 5.0
$Upcase	файл - таблица соответствия заглавных и прописных букв в имен файлов на текущем томе
$Extend	Файл хранит расширенную информацию файловой системы NTFS начиная с версии 5.0 (дисковые квоты $Quota, журнал изменений $UsrJrnl, точки монтирования $ReparsePoint и т.д.)

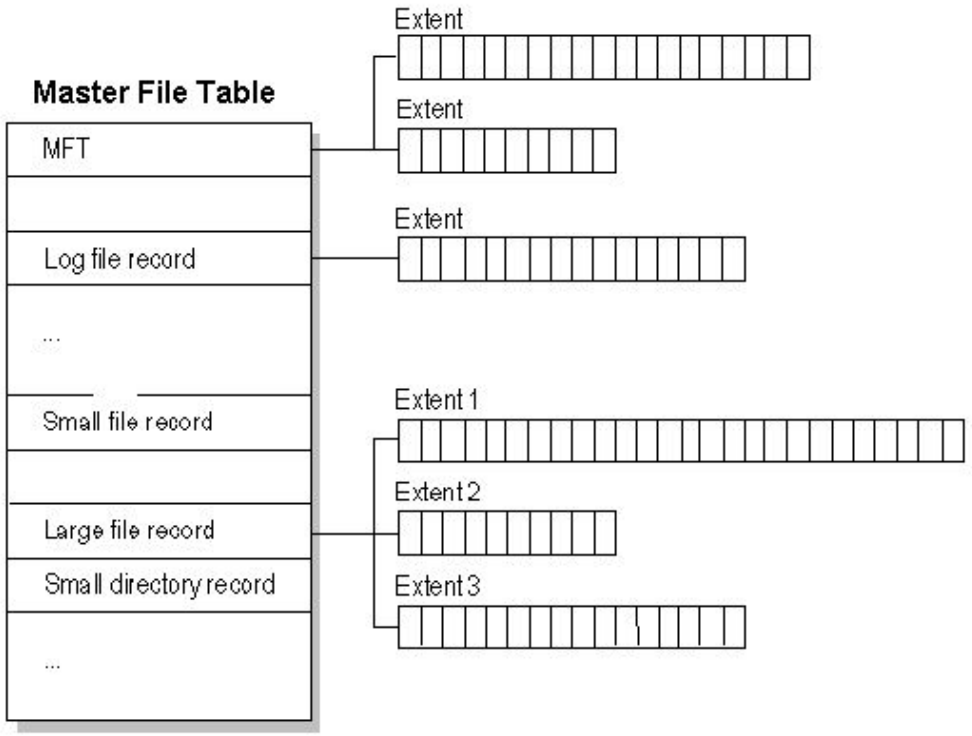
Файлы и их атрибуты (1)

- Каждый файл на томе NTFS представлен записью MFT, под которое отводится определенное количество пространства (4Кбайт).
- Запись MFT состоит из заголовка записи, за которым следует последовательность пар (заголовок атрибута, значение).



Файлы и их атрибуты (2)

- Обычно атрибуты располагаются непосредственно после заголовков, однако если длина значения слишком велика, чтобы поместиться в запись таблицы MFT, она может быть помещена в отдельный блок диска. Такой атрибут называется **нерезидентным**.



Заголовок атрибута

Смещение, байт	Размер, байт	Описание
0x00	4	Тип атрибута
0x04	4	Размер области памяти, занимаемой атрибутом
0x08	1	Флаг нерезидентного атрибута
0x09	1	Длина имени атрибута
0x0A	2	Смещение области данных атрибута
0x0C	2	Флаг упакованного атрибута
0x0E	2	Идентификатор атрибута

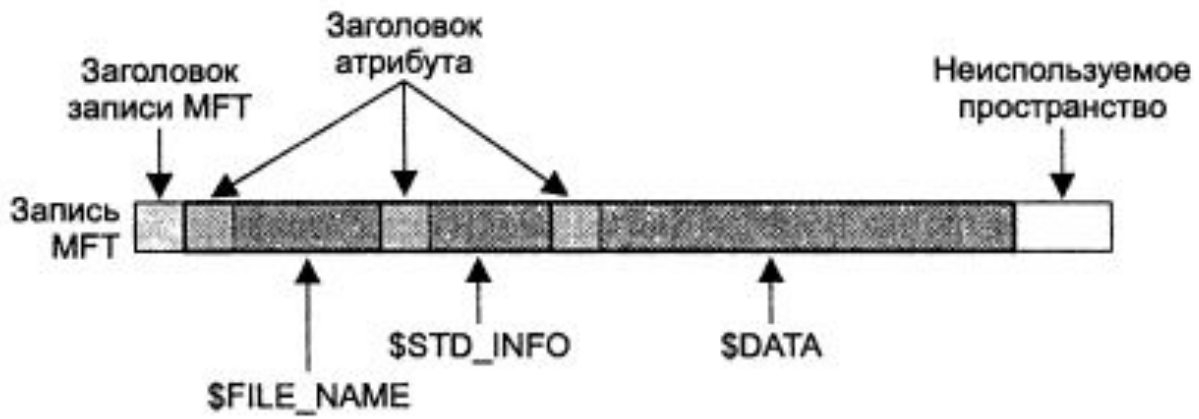
Атрибуты файлов NTFS (1)

$STANDARD_INFORMATION (стандартная информация)	общая информация: дата и время создания и последнего изменения файла, дата и время последнего доступа к файлу, флаги доступа к файлу, а также дата и время изменения записи MFT, соответствующей данному файлу
$ATTRIBUTE_LIST (список атрибутов)	используется, если для хранения атрибутов файла (каталога) требуется больше одной записи MFT
$FILE_NAME (имя файла)	хранится имя файла или каталога, набор флагов доступа, размер файла, а также ссылка на запись MFT каталога, в котором хранится данный файл или каталог
$SECURITY_DESCRIPTOR (дескриптор безопасности)	фиксирует информацию о том, кто может обращаться к файлу, кто является его владельцем и так далее (ACL)
$DATA (данные)	данные файла

Атрибуты файлов NTFS(2)

$VOLUME_VERSION	версия тома, используется только в системных файлах тома
$VOLUME_INFORMATION (информация о томе)	используется только в системном файле тома и включает в частности версию и имя тома
$VOLUME_NAME	отметка тома
$INDEX_ROOT (корневой индекс)	корневая вершина дерева типа B+ (корень индекса), используемого для поиска файлов в каталоге (резидентный)
$INDEX_ALLOCATION (размещение индекса)	нерезидентные части индексного списка B-дерева
$BITMAP (битовый массив)	предоставляет информацию об использовании записей в MFT или каталоге

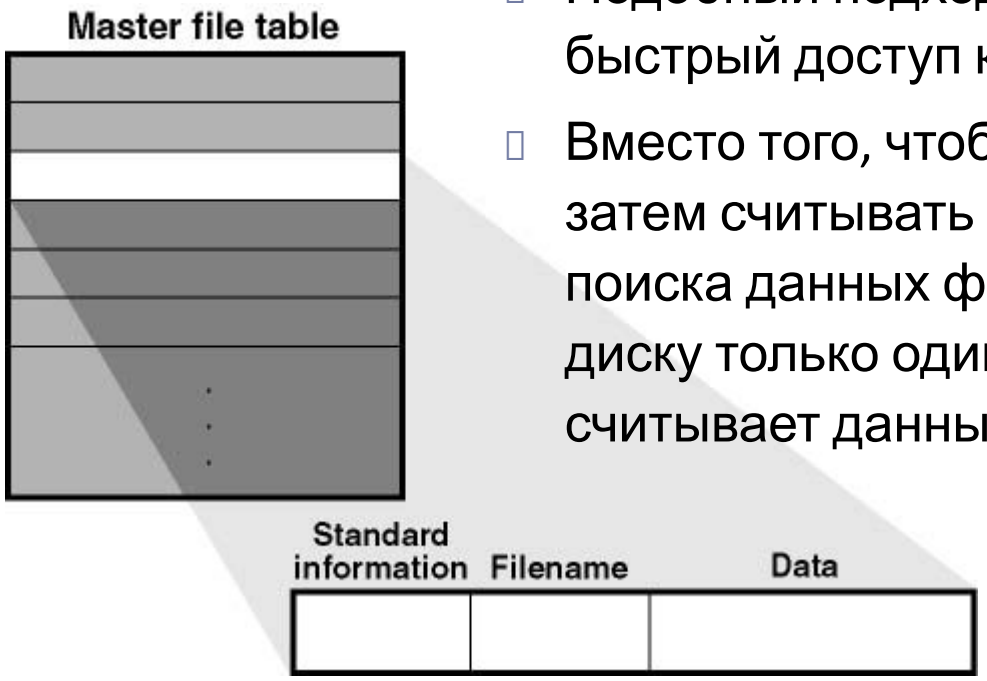
Запись MFT



- заголовок записи MFT
- стандартная информация
- имя файла
- данные
- дескриптор безопасности*

* в последних версиях NTFS все дескрипторы безопасности хранятся в отдельном файле, потому что дескрипторы многих файлов одинаковы.

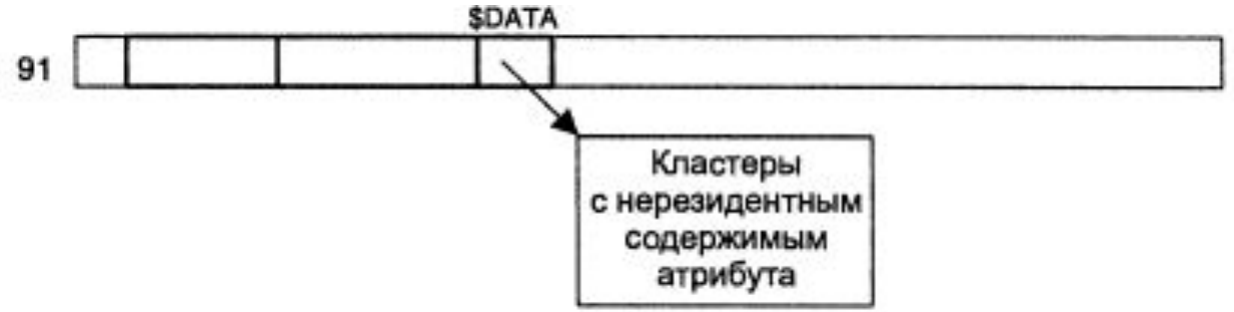
Резидентное хранение файлов и каталогов



- Файлы и каталоги с размером менее размера записи MFT могут полностью содержаться внутри записи MFT.
- Подобный подход обеспечивает очень быстрый доступ к файлам.
- Вместо того, чтобы искать файл в таблице и затем считывать цепочку кластеров для поиска данных файла, NTFS обращается к диску только один раз и немедленно считывает данные.

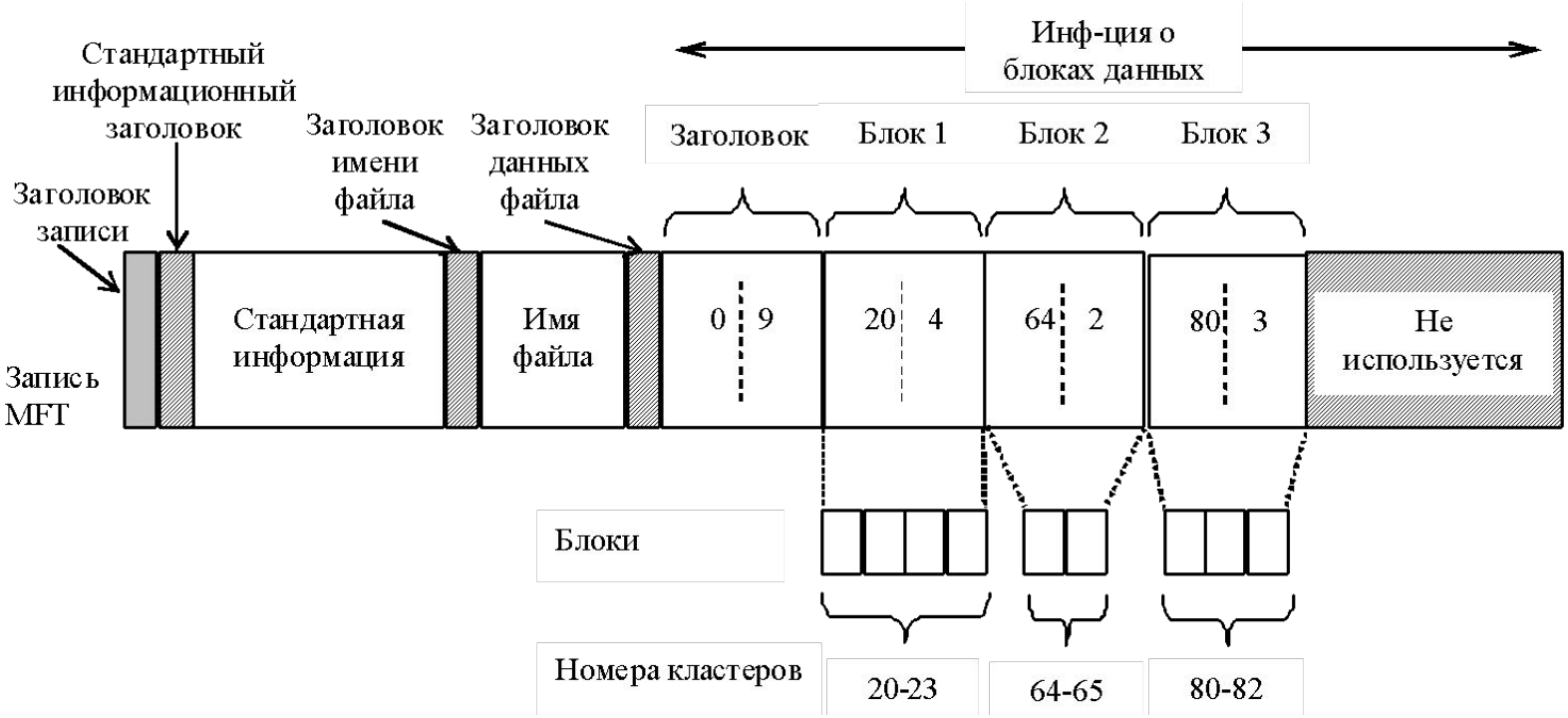
Нерезидентное хранение файлов среднего размера

- В большинстве случаев все данные файла не помещаются в запись MFT, поэтому этот атрибут, как правило, является нерезидентным.



- Файл без фрагментации описывается всего одной записью (стартовый кластер, число кластеров).

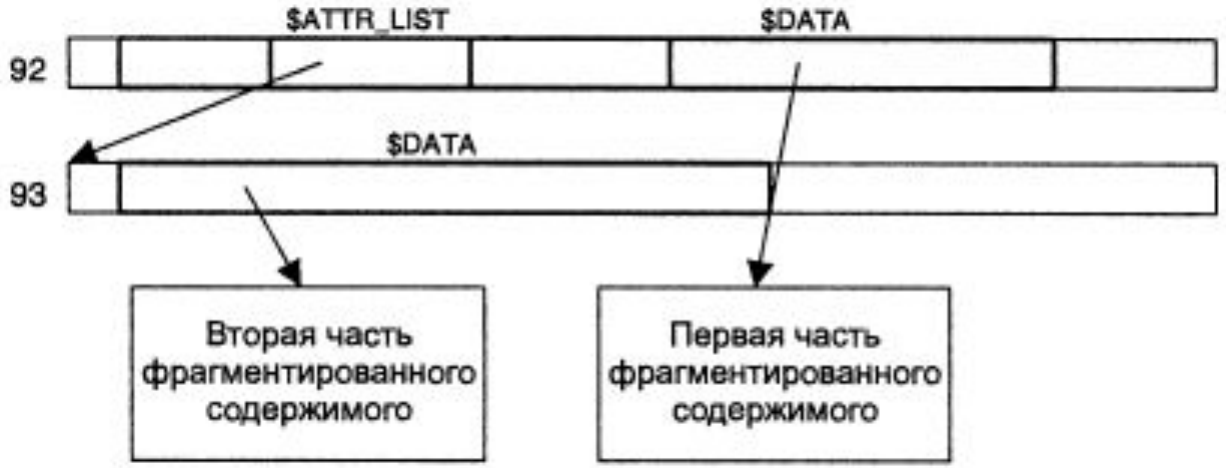
Нерезидентное хранение файлов среднего размера



- На рисунке файл состоит из 3-х серий кластеров: 20 – 23, 64 – 65, 80 – 82. Число таких серий зависит от того насколько удачно ФС сумела найти место для хранения файла.

Нерезидентное хранение больших файлов

- Если файл настолько велик (или сильно фрагментирован), что его атрибут данных не помещается в одной записи MFT, то этот атрибут становится нерезидентным, то есть он находится в другой записи таблицы MFT, ссылка на которую помещена в исходной записи о файле



Вопрос

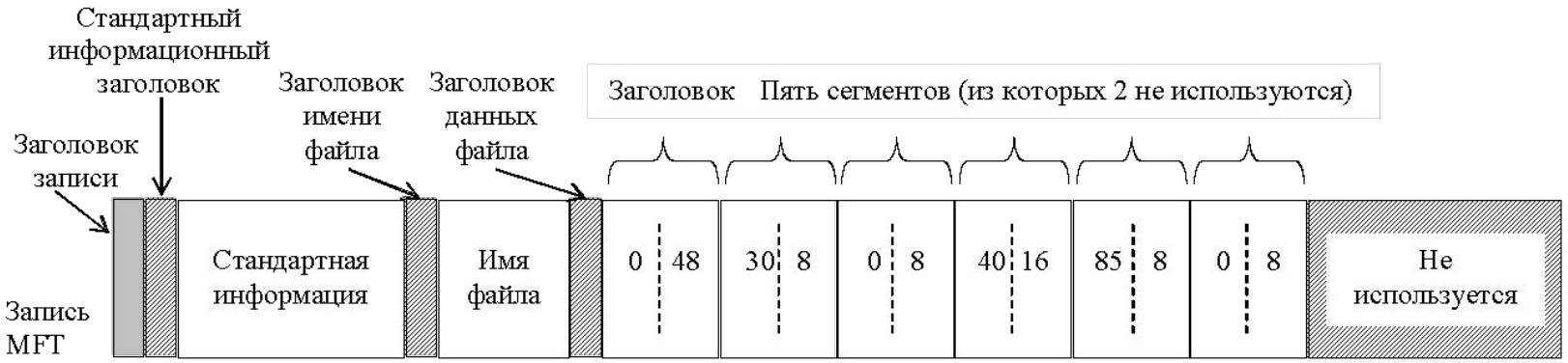
- Сравните организацию хранения больших файлов в NTFS и файловых системах ОС UNIX.

Сжатие файлов

- Файловая система NTFS поддерживает прозрачное сжатие файлов следующим образом:
  - Когда файловая система NTFS записывает на диск файл, помеченный для сжатия, она изучает первые 16 кластеров. Затем к этим кластерам применяется алгоритм сжатия.
  - Если полученные на выходе данные могут поместиться в 15 или менее кластеров, то сжатые данные записываются на диск, предпочтительно в виде одной серии.
  - Если получить выигрыш не получается, то группа из16 кластеров записывается без сжатия.
  - Затем алгоритм повторяется для группы следующих 16 кластеров и т.д.
- Сжатие файла частями по 16 кластеров явилось компромиссом, если бы порции были меньше, то эффективность бы сжатия снизилась. Если размер порции был бы больше, то это замедлило бы произвольный доступ.

Пример сжатия файла (1)

Пример 48-блочного файла сжатого до 32 блоков



Запись MFT после сжатия файла

Пример сжатия файла (2)

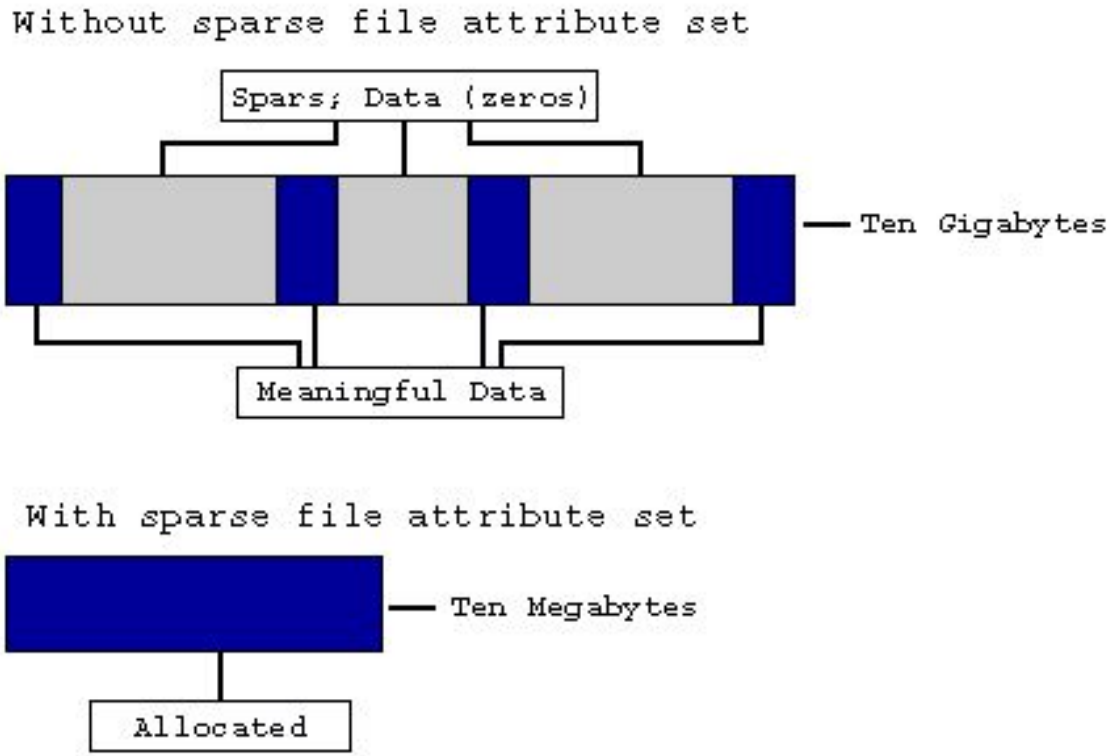
- На предыдущем слайде показан файл, в котором первые 16 блоков успешно сжаты в 8 блоков, следующие 16 не могут быть сжаты, наконец, последние 16 блоков также успешно сжаты на 50%.
- Эти три части файла записаны в виде трех сегментов, информация о которых хранится в записи MFT. «Пропущенные» блоки обозначаются в записи MFT как сегменты с нулевым дисковым адресом. На слайде за заголовком (0,48) следует 5 пар, две для первого (сжатого) сегмента, одна для несжатого и две для последнего (сжатого) сегмента.
- При чтении этого файла система NTFS должна знать, какие из сегментов файла сжаты, а какие нет. Она видит это по дисковым адресам. Дисковый адрес 0 указывает на то, что предыдущий сегмент сжат. Дисковый блок 0 не может использоваться для хранения данных во избежание неоднозначности (это загрузочный сектор).
- Произвольный доступ к сжатому файлу возможен, например, для чтения блока 35 необходимо определить где находится этот блок и

Разреженные файлы (sparse files)

- Если у вас есть файлы, которые содержат множество нулей, т.е. «пустые области», то NTFS позволяет сохранять пространство диска, предоставляя возможность использовать **sparse** (разреженные) файлы.
- В случае разреженных файлов система просто не выделяет место для «пустых областей» – в результате чего и достигается уменьшение размера файла. При обращении системы к частям, отмеченным как пустые, NTFS возвращает нулевые значения. При просмотре свойств файла система сообщит о зарезервированном для него размере, хотя фактический объем может быть меньше.
- Разреженные файлы применяются, в частности, в журнале NTFS ($LogFile).

Создание разреженных файлов

- Разреженные файлы конвертируются с помощью следующей команды: **fsutil sparse**.



Многопоточные файлы

- При необходимости в одном файле, записанном на диске NTFS, можно хранить несколько потоков информации. Это позволяет, в частности, снабжать файлы документов дополнительной информацией, хранить в одном файле несколько версий документов (например, на разных языках), хранить в отдельных потоках одного файла программный код и данные и т. п.
- При создании файла основные данные следует записать в неименованный поток. Затем необходимо создать внутри того же файла именованный поток, предназначенный для данных образа. Теперь один файл будет содержать два потока.

Пример создания многопоточного файла

- В интерфейсе командной строки перейдем в раздел NTFS (например, в папку, содержащую системные шрифты) и введем следующую команду (не делайте лишних пробелов!):

```
C:\WINDOWS\Fonts>dir > New_Stream.TXT:New_Stream
```

- В результате выполнения этой команды система создаст файл *New_Stream.TXT*. Он будет содержать два потока: неименованный, в котором находится 0 байт, и именованный (с именем New_Stream), где будет находиться результат выполнения команды *dir*. Доступ к именованному потоку можно получить, обратившись к нему по имени через двоеточие после имени файла.
- Для вывода содержимого именованного потока воспользуемся:

```
C:\WINDOWS\Fonts>more < New_Stream.TXT:New_Stream
```

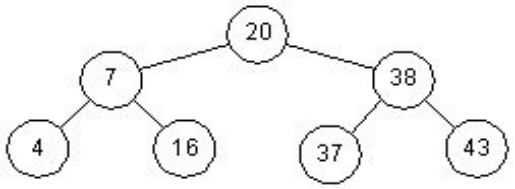
Каталоги NTFS

- Каталог на NTFS представляет собой специфический файл, хранящий ссылки на другие файлы и каталоги, создавая иерархическое строение данных на диске. Файл каталога поделен на блоки, каждый из которых содержит имя файла, базовые атрибуты и ссылку на элемент MFT, который уже предоставляет полную информацию об элементе каталога.
- Внутренняя структура каталога представляет собой бинарное B+ дерево (форма двоичного дерева, каждый узел которого может иметь более двух дочерних узлов), элементы которого сортируются по имени.

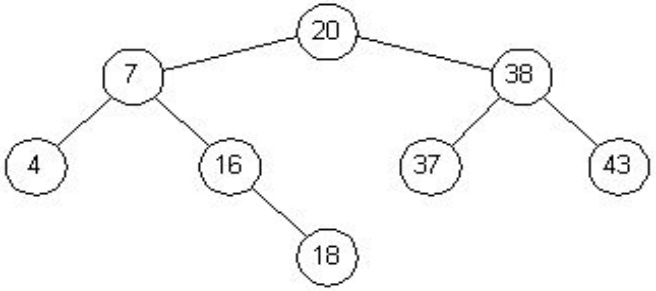
Поиск в каталогах NTFS

- Для поиска файла с данным именем в линейном каталоге, таком, например, как у FAT-а, системе приходится просматривать все элементы каталога, пока она не найдет нужный. Бинарное же дерево располагает имена файлов таким образом, чтобы поиск файла осуществлялся более быстрым способом – с помощью получения двухзначных ответов на вопросы о положении файла.
- Вывод – для поиска одного файла среди 1000, например, FAT придется осуществить в среднем 500 сравнений (наиболее вероятно, что файл будет найден на середине поиска), а системе на основе дерева – всего около $Log_2(N)$, т.е. 10-ти ($2^{10} = 1024$).

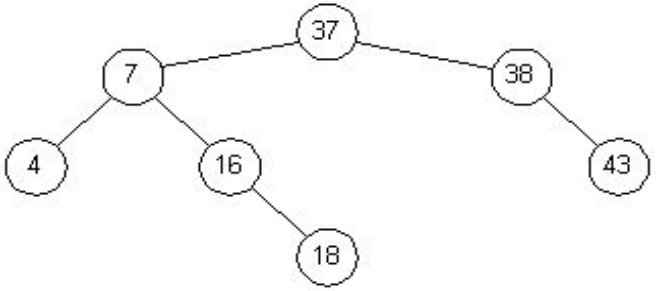
Бинарное дерево



Бинарное дерево



Бинарное дерево после добавления узла 18

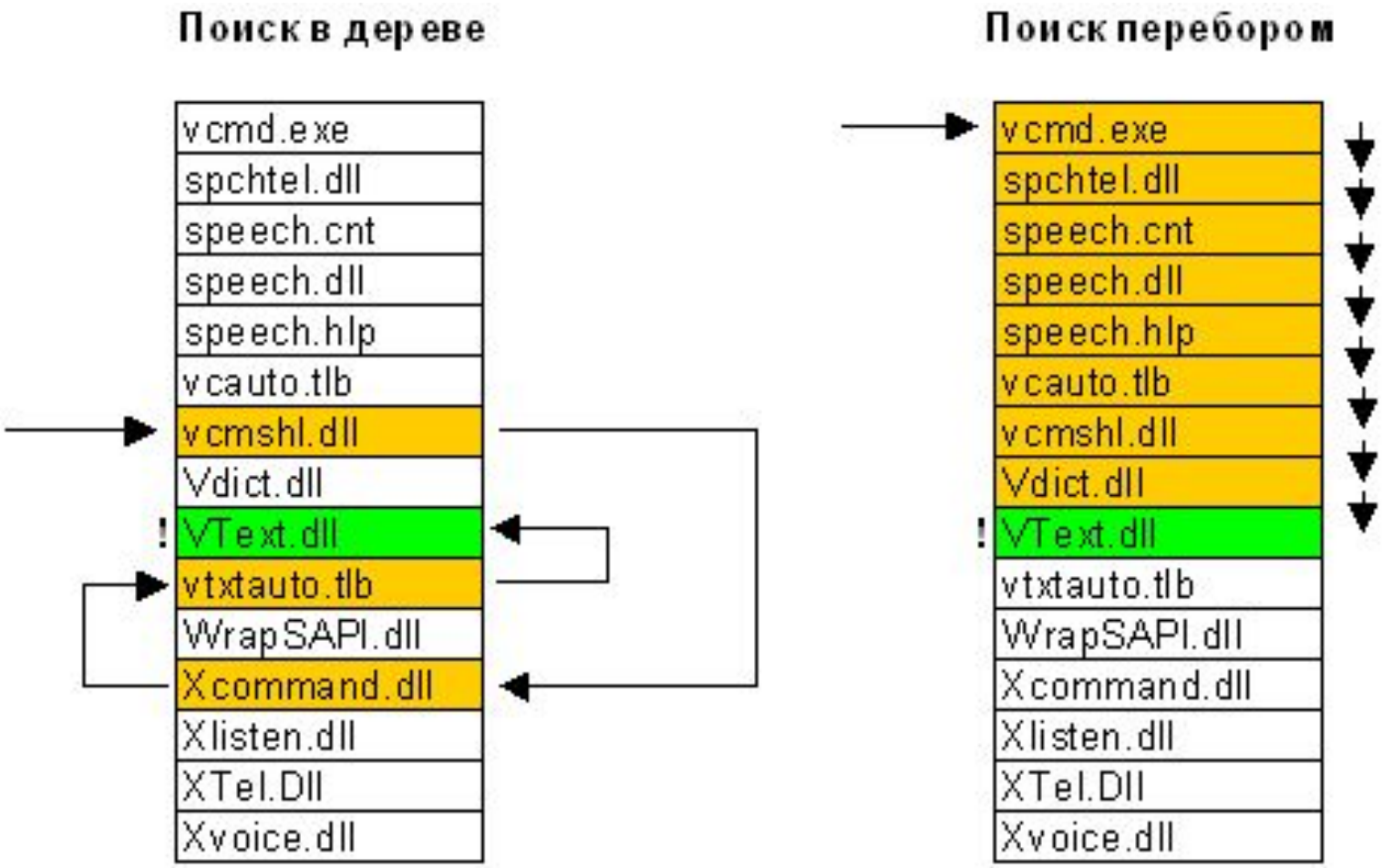


Бинарное дерево после удаления узла 20

Вопрос

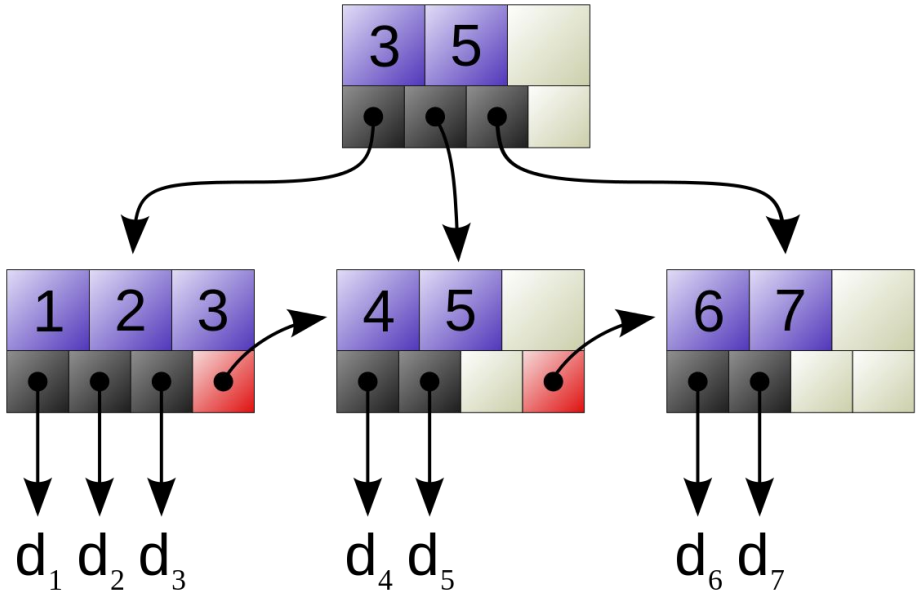
- Нарисуйте бинарное дерево.

Простой и бинарный поиск



B+ дерево

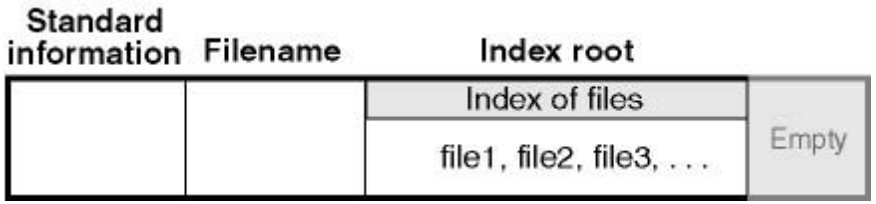
- B+ дерево – структура данных, представляет собой сбалансированное дерево поиска.
- B+ дерево является модификацией бинарного дерева, истинные значения ключей которого содержатся только в листьях, а во внутренних узлах – ключи-разделители, содержащие диапазон изменения ключей для поддеревьев.



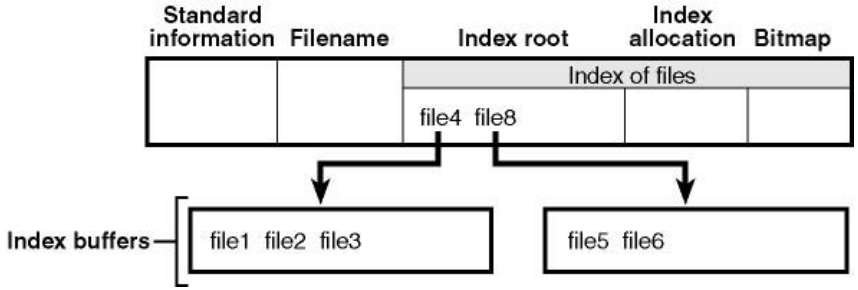
Хранение каталогов

- Небольшие записи каталогов находятся полностью внутри структуры MFT, для хранения используется атрибут $Index_Root (корневой индекс), который всегда резидентный.
- В том случае, когда каталог не умещается целиком в MFT, требуется выделение дополнительного пространства:
  - корневой индекс хранит корень дерева B+ со ссылками на нерезидентные индексные буферы;
  - для хранения описания файлов выделяются нерезидентные индексные буферы (4 Кбайт), каждый из которых адресуется с использованием виртуального номера кластера (virtual clusters numbers, VCN) для сквозной нумерации кластеров в рамках одной записи MFT;
  - соответствие между VCN и LCN (logical clusters numbers) хранится в атрибуте каталога $Index_Allocation.

Пример хранения каталогов

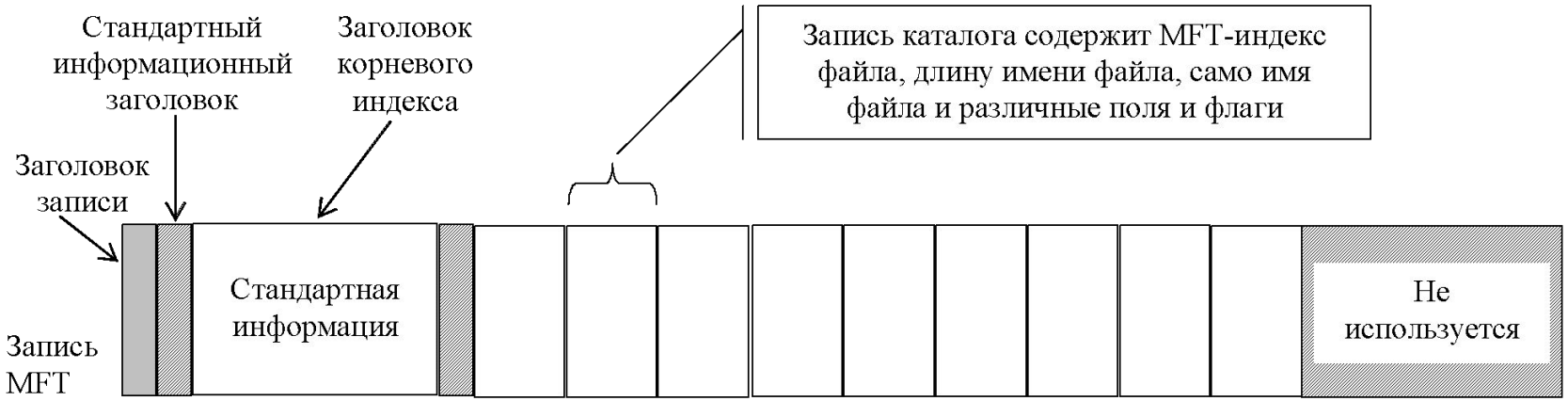


а) запись MFT для небольшого каталога (резидентное хранение)



б) запись MFT для “большого” каталога (нерезидентное хранение)

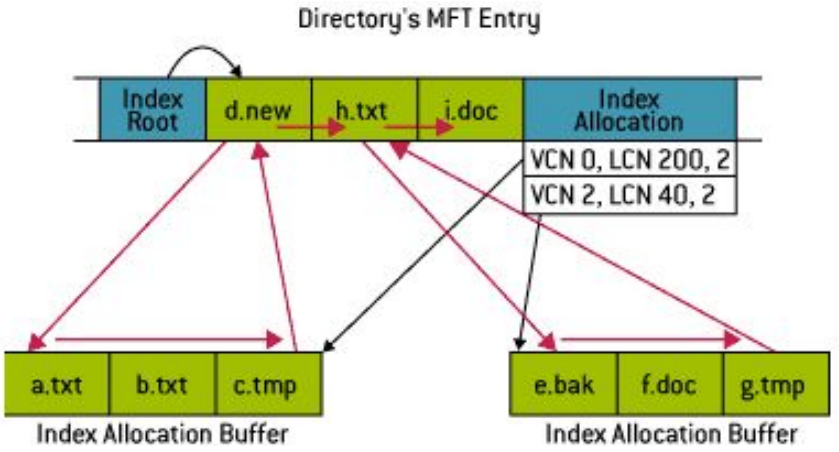
Запись MFT для небольшого каталога



- Запись MFT для небольшого каталога содержит несколько записей, каждая из которых описывает файл или каталог.
- Фиксированная запись содержит индекс записи MFT файла, длину имени файла, а также другие разнообразные поля и флаги.
- Поиск файла в каталоге по имени состоит в последовательном переборе всех имен файлов.

Пример нерезидентного хранения каталогов (1)

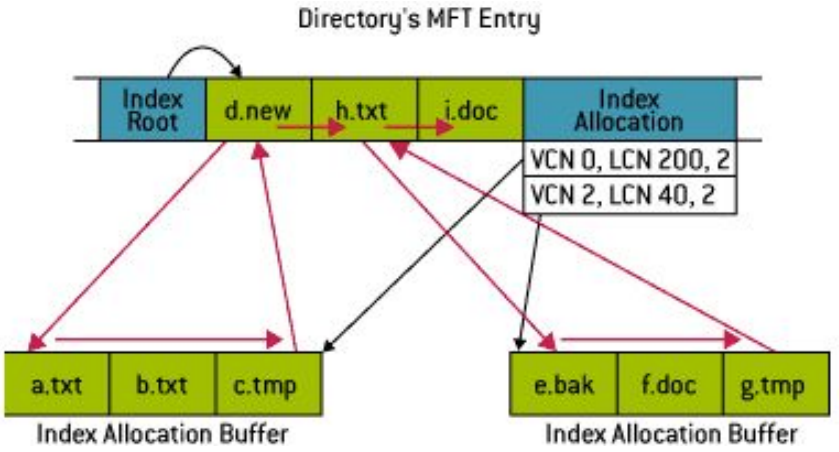
- На рисунке показана запись MFT каталога, в трех узлах которой содержится девять элементов.
- Корень B+ дерева находится в атрибуте $INDEX_ROOT.



- В записи MFT каталога девять элементов не помещается, поэтому для их хранения NTFS выделяет два буфера размещения индексов (index allocation). Обычно, корень индекса и буферы размещения индексов могут хранить элементы для более чем трех файлов, в зависимости от длины имен. VCN представляет собой порядковый номер кластера в файле или каталоге.
- Красные стрелки указывают, что элементы NTFS хранятся в алфавитном порядке.

Пример нерезидентного хранения каталогов (2)

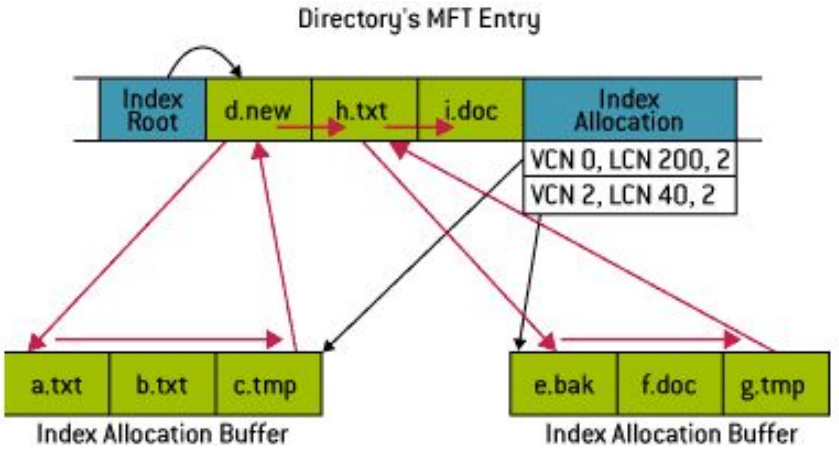
- Рассмотрим ситуацию открытия файла *e.bak*.
- Во-первых, NTFS считывает атрибут индексного корня $INDEX_ROOT и сравнивает строку *e.bak* с именем первого элемента *d.new*.
- Т.к. алфавитный номер *e.bak* больше, чем *d.new*, то NTFS переходит к следующему элементу – *h.txt*.
- Повторив операцию сравнения, NTFS выясняет, что алфавитный номер *e.bak* меньше, чем *h.txt*.



- Затем NTFS отыскивает в $INDEX_ROOT номер виртуального кластера VCN индексного буфера, содержащего элементы каталога, алфавитные номера которых меньше, чем *h.txt*, но больше, чем *d.new*.

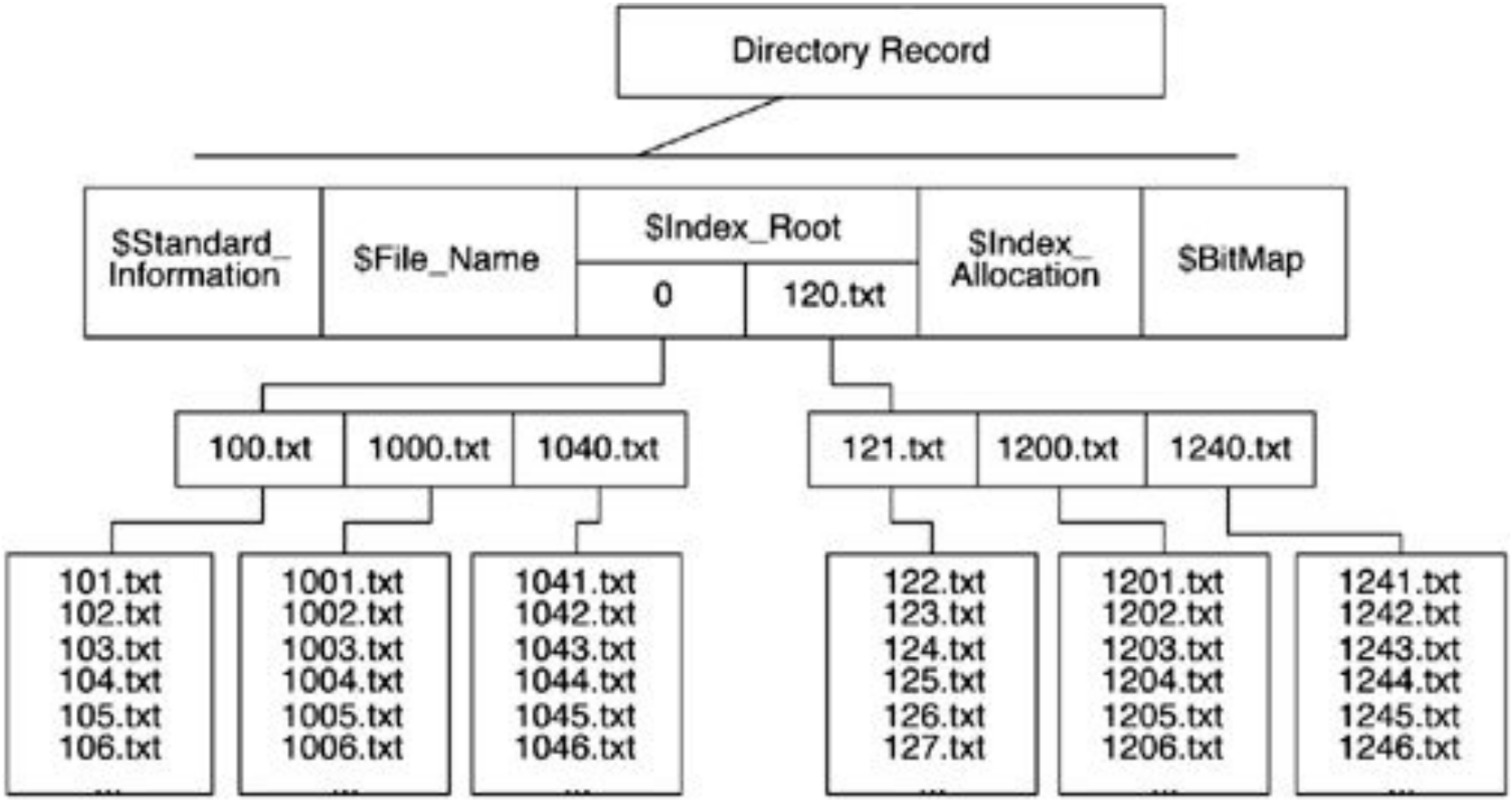
Пример нерезидентного хранения каталогов (3)

- На основании информации из $INDEX_ALLOCATION, файловая система преобразует VCN в логический номер кластера LCN, т. е. номер кластера относительно начала тома.
- Далее NTFS читает буфер размещения индексов и просматривает его в поисках совпадений.
- Если элемент индексного буфера для *h.txt* не содержит файла *e.bak*, то NTFS сообщает о неудачном завершении

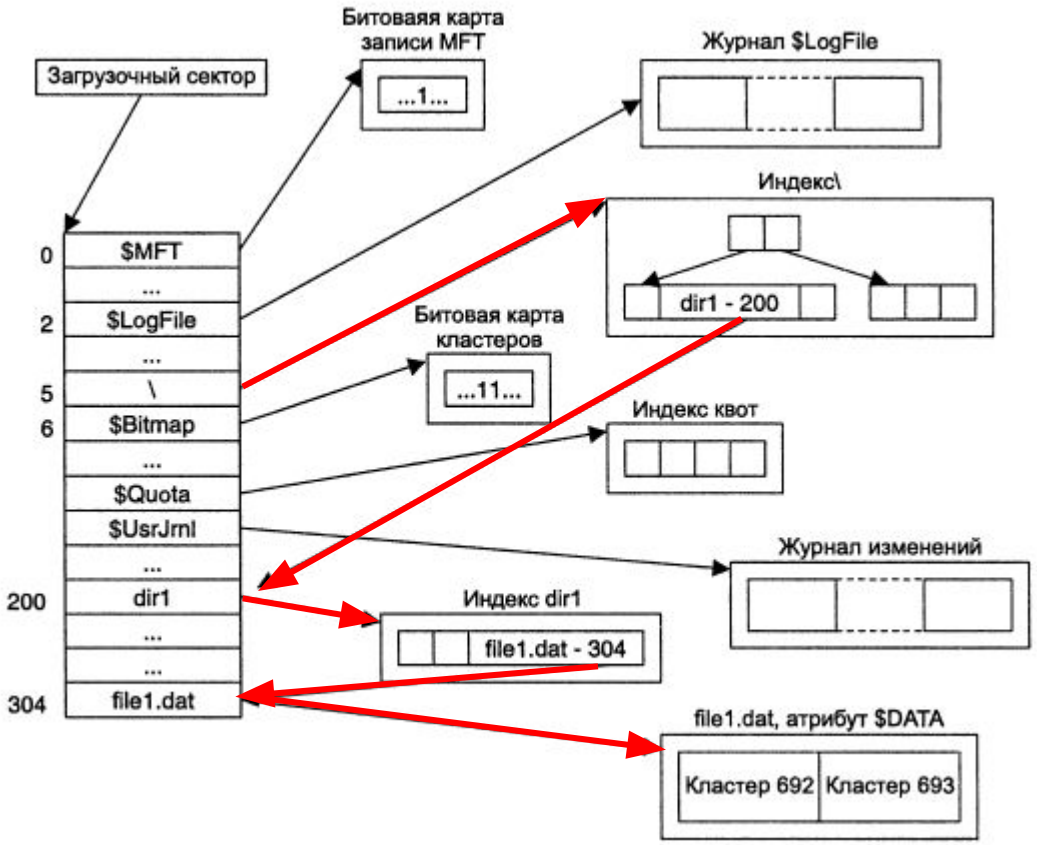


- На рисунке первый же элемент индексного буфера совпадает с критерием поиска, и NTFS читает номер записи MFT *e.bak*.

Хранение корневого каталога



Пример чтения каталогов и файла NTFS



- файл \dir1\file1.dat