Файловые системы

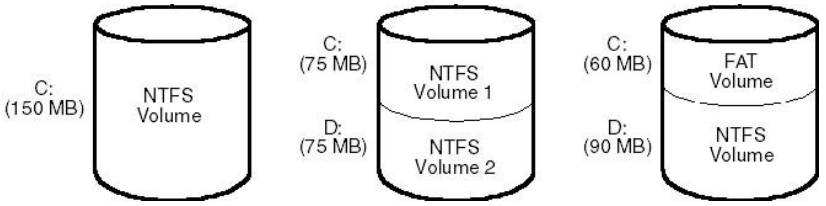
Файловая система NTFS

Краткое описание

- 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.

Понятия и термины NTFS

- Структура NTFS начинается с тома (volume). Том соответствует логическому разделу на диске и создается, когда Вы форматируете диск или часть его для NTFS.
- На одном диске может находиться один или несколько томов.



- NTFS обрабатывает каждый том независимо от других.

Типы томов

- *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.
- *Чередующийся набор томов* (*stripped , RAID 1*) – том, состоящий из нескольких разделов, по которым равномерными блоками распределены данные. Размер блока данных - 64 Кбайт. Первый блок данных размером в 64 Кбайт хранится на первом разделе, вторые 64 Кбайт на втором и т.д. Чередующиеся наборы томов могут повысить производительность системы, если использовать разделы, размещенные на разных дисках, поскольку операции чтения-записи могут выполняться параллельно.
- *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)

Развитие NTFS

- NTFS была разработана для семейства ОС MS Windows NT.
- В ОС Windows 2000-2003 реализовано много новых возможностей, например, динамические диски, которым будет посвящена дополнительная тема.

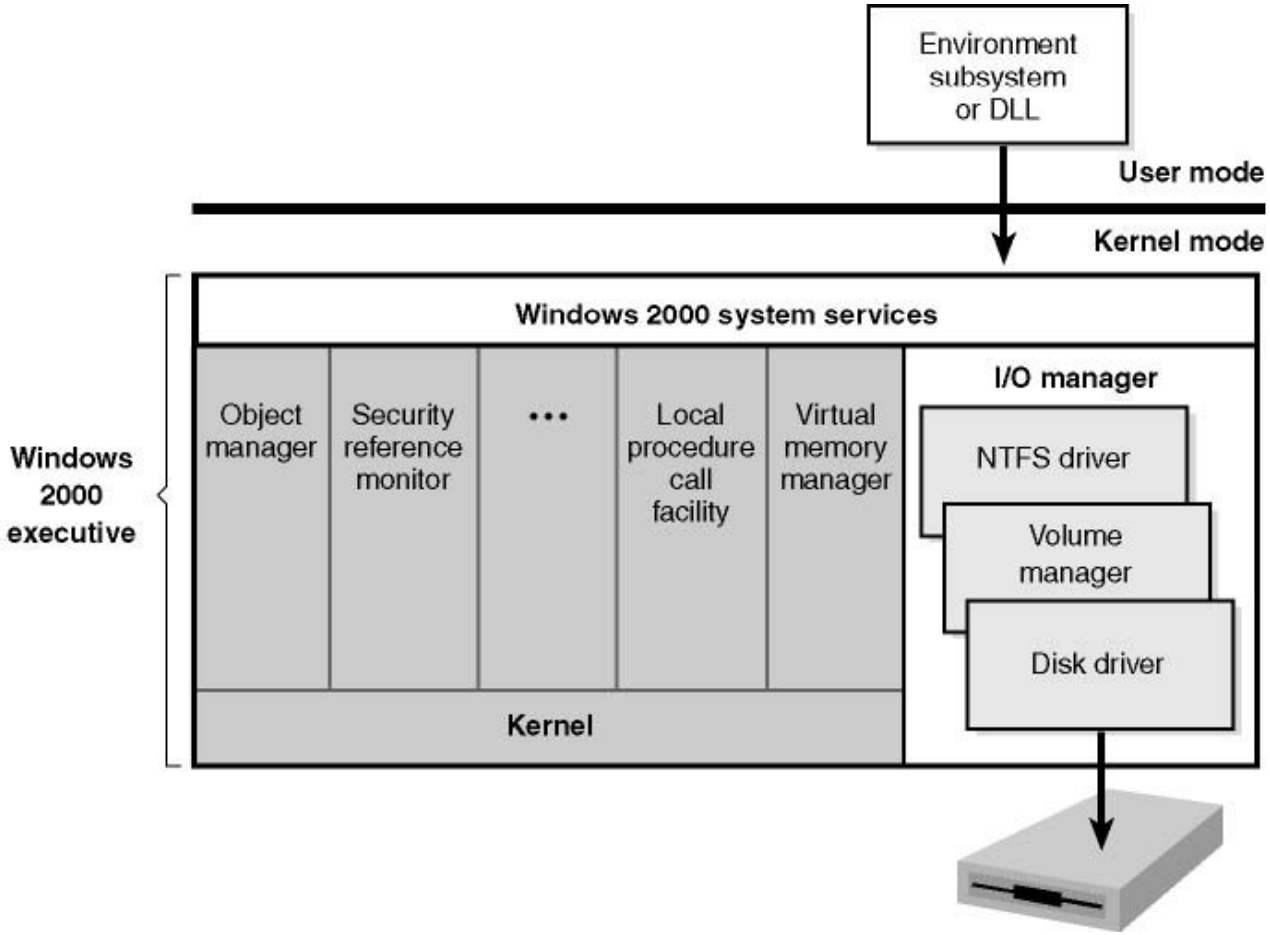
Размер кластера для NTFS

Размер логического диска	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

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

NTFS и архитектура Windows 2000



Физическая структура NTFS

NTFS поддерживает размеры кластеров – от 512 байт до 64 Кбайт

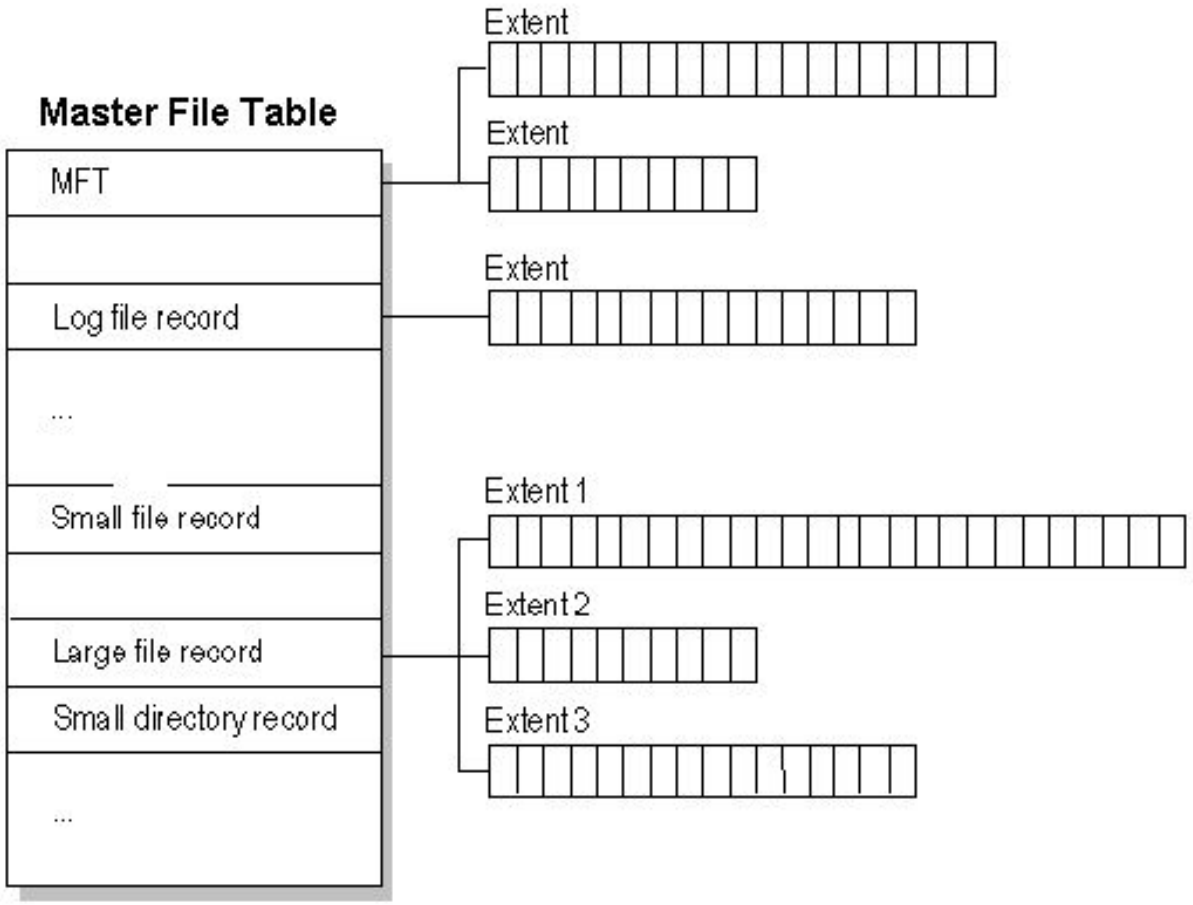
Физическая структура NTFS

- Том NTFS условно делится на две части. Первые 12% тома отводятся под так называемую MFT зону – пространство, в которое растет метафайл MFT (об этом ниже). Запись каких-либо данных в эту область невозможна. MFT-зона всегда держится пустой – это делается для того, чтобы самый главный, служебный файл (MFT) не фрагментировался при своем росте. Остальные 88% тома представляют собой обычное пространство для хранения файлов.
- Свободное место тома, однако, включает в себя всё физически свободное место – незаполненные куски MFT-зоны туда тоже включаются. Механизм использования MFT-зоны таков: когда файлы уже нельзя записывать в обычное пространство, MFT-зона просто сокращается (в текущих версиях ОС ровно в два раза), 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: никакой аномалии тут нет. Метафайл MFT все-таки может фрагментироваться, хоть это и было бы нежелательно.

MFT и ее структура

- Каждый файл на томе NTFS представлен записью в специальном файле, называемом главной файловой таблицей (MFT – master file table).
- NTFS резервирует первые 16 записей таблицы для специальной информации. Первая запись этой таблицы описывает непосредственно главную файловую таблицу;
  - за ней следует зеркальная запись (mirror record) MFT. Если первая запись MFT разрушена, то NTFS читает вторую запись для отыскания зеркального файла MFT, первая запись которого идентична первой записи MFT. Местоположения сегментов данных MFT и зеркального файла MFT записаны в секторе начальной загрузки. Дубликат сектора начальной загрузки находится в логическом центре диска.
  - Третья запись MFT — файл регистрации (log file); используется для восстановления файлов. Файл регистрации подробно описан ниже. Семнадцатая и последующие записи главной файловой таблицы используются собственно файлами и каталогами (также рассматриваются как файлы NTFS) на томе. На слайде показана упрощенная структура MFT.

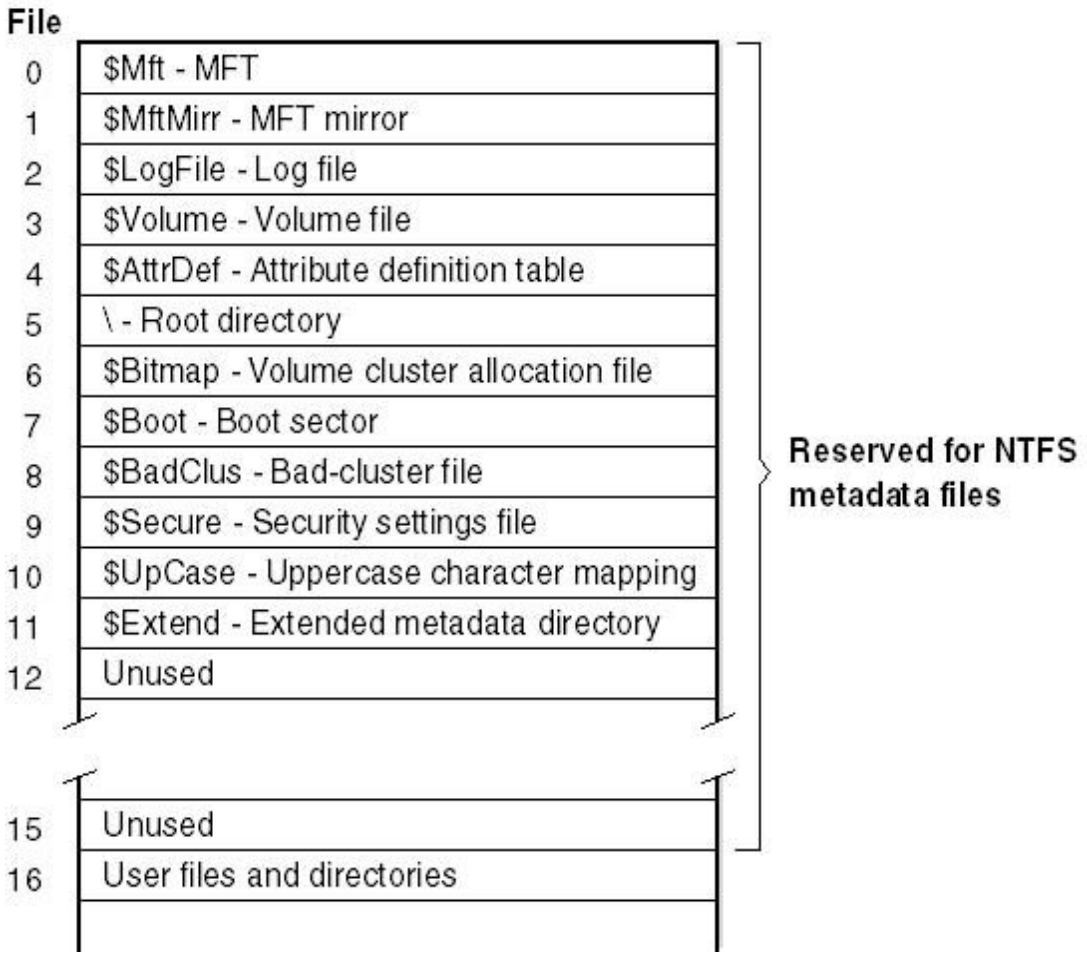
MFT и ее структура



Метафайлы

- 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 диска.

Метафайлы


File
0
$Mft - MFT
1
$MftMirr - MFT mirror
2
$LogFile - Log file
3
$Volume - Volume file
4
$AttrDef - Attribute definition table
5
\ - Root directory
6
$Bitmap - Volume cluster allocation file
7
$Boot - Boot sector
8
$BadClus - Bad-cluster file
9
$Secure - Security settings file
10
$UpCase - Uppercase character mapping
11
$Extend - Extended metadata directory
12
Unused
15
Unused
16
User files and directories
Reserved for NTFS metadata files

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, точки монтирования и т.д.)

Атрибуты файлов

- Главная файловая таблица отводит определенное количество пространства для каждой записи файла (4Кбайт). Атрибуты файла записываются в распределенное пространство MFT.
- Каждая запись MFT состоит из заголовка записи, за которым следует последовательность пар (заголовок атрибута, значение).
- Как правило значения атрибутов 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**. Например, таким атрибутом может являться атрибут **Data**.

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)	Стандартный атрибут. 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)	Перечисляет все другие атрибуты.
Filename (имя файла)	Имя файла или каталога. В этом атрибуте хранится имя файла или каталога, набор флагов доступа, размер файла, а также ссылка на запись MFT каталога, в котором хранится данный файл или каталог.
MS-DOS Name	Имя файла в формате 8.3
Version	Номер последней версии файла
Security Descriptor (дескриптор безопасности)	Фиксирует информацию о том, кто может обращаться к файлу, кто является его владельцем и так далее (ACL)
Data (данные)	Содержит данные файла

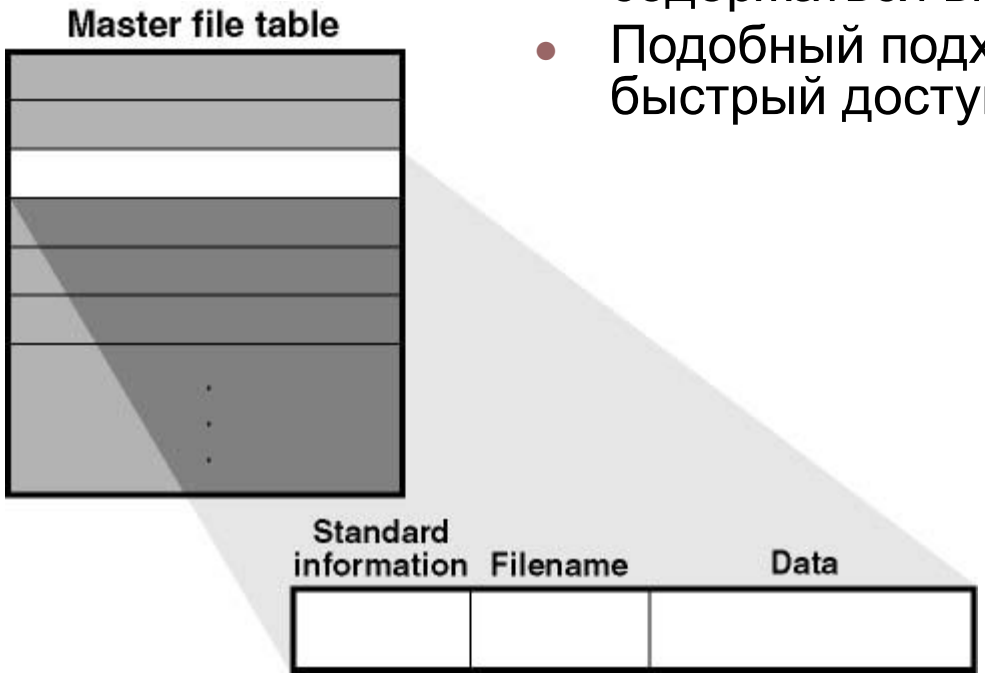
Атрибуты файлов NTFS(2)

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+, используемого для поиска файлов в каталоге. Всегда резидентный.
Index Allocation (размещение индекса)	Узлы ветвей дерева типа B+. нерезидентные части индексного списка B-дерева
External Attribute Information	номер первого кластера и количество кластеров нерезидентного атрибута
Bitmap (битовый массив)	Предоставляет информацию об использовании записей в MFT или каталоге

Хранение файлов

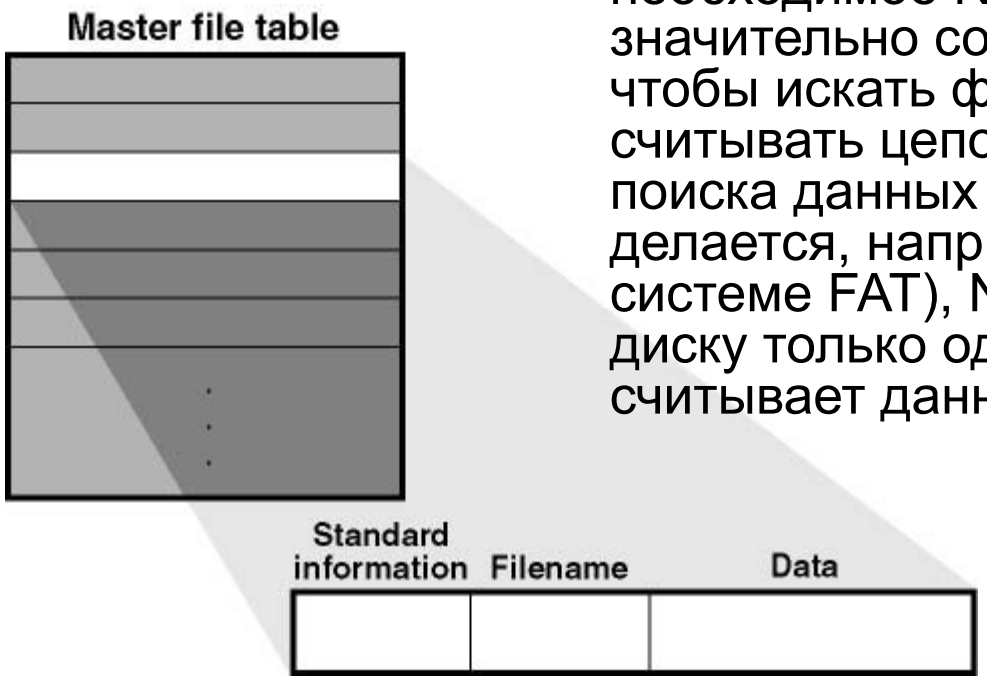
- Файлы NTFS в общем случае состоят из следующих атрибутов:
  - заголовок записи MFT (H - header)
  - стандартная информация (SI - standard information);
  - имя файла (FN - file name);
  - данные (data);
  - дескриптор безопасности (SD - security descriptor).

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.

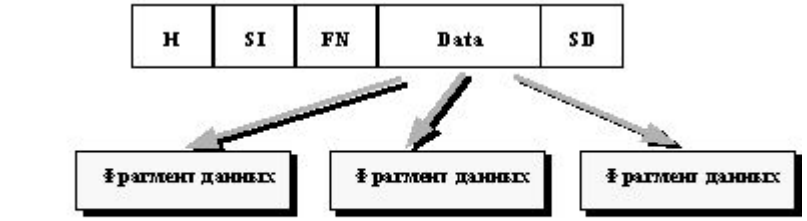
Резидентное хранение файлов и каталогов



- Если значение атрибута хранится непосредственно в MFT, время, необходимое NTFS для доступа к нему, значительно сокращается. Вместо того, чтобы искать файл в таблице и затем считывать цепочку кластеров для поиска данных файла (как это делается, например, в файловой системе FAT), NTFS обращается к диску только один раз и немедленно считывает данные.

Нерезидентное хранение файлов

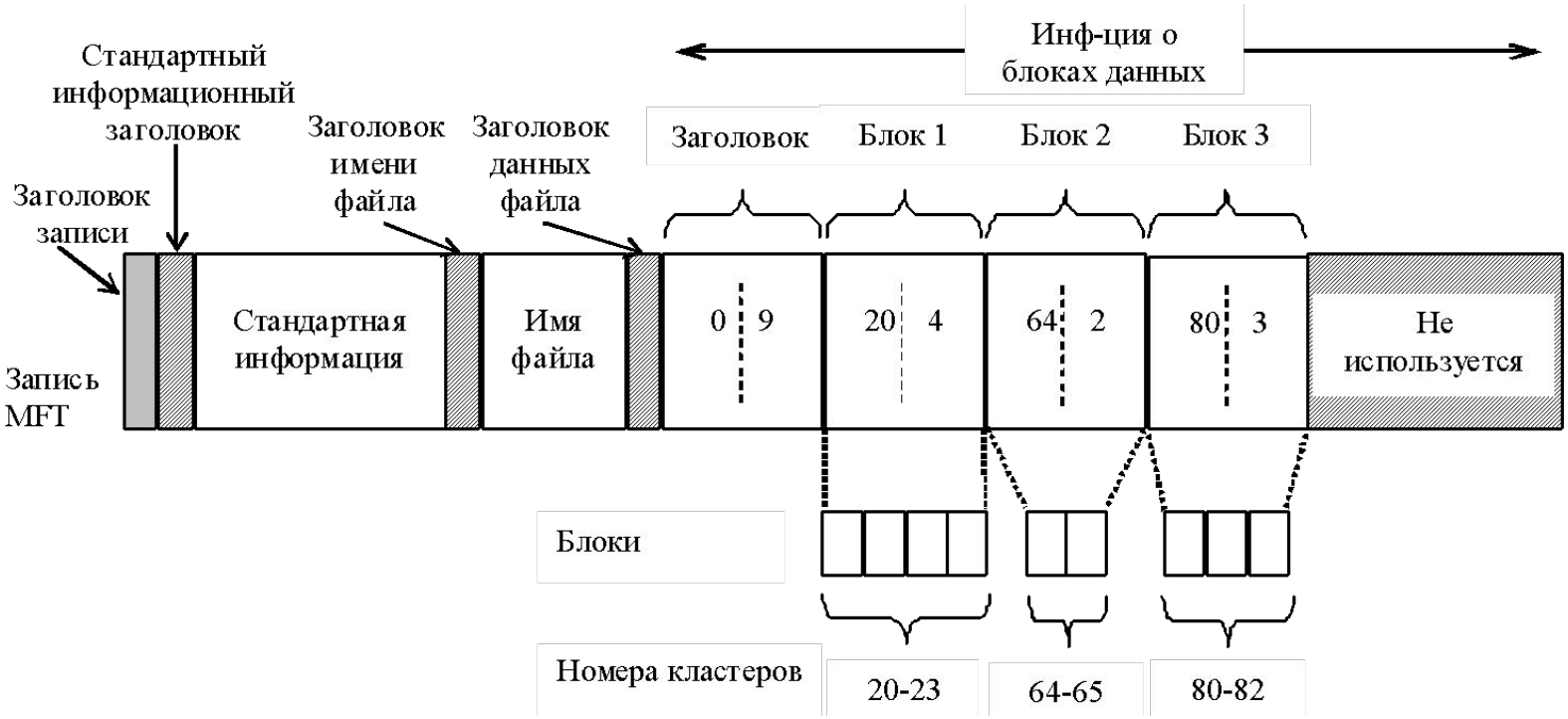
- Конечно, в большинстве случаев все данные файла не помещаются в запись MFT, поэтому этот атрибут, как правило, является нерезидентным.



Data- данные в нерезидентной

- Для увеличения эффективности кластеры выделяются файлам по возможности в виде блоков (серий, пробегов) последовательных кластеров. Каждый блок описывается отдельной записью – (стартовый кластер, число кластеров). Файл без фрагментации описывается всего одной записью.

Нерезидентное хранение файлов среднего размера

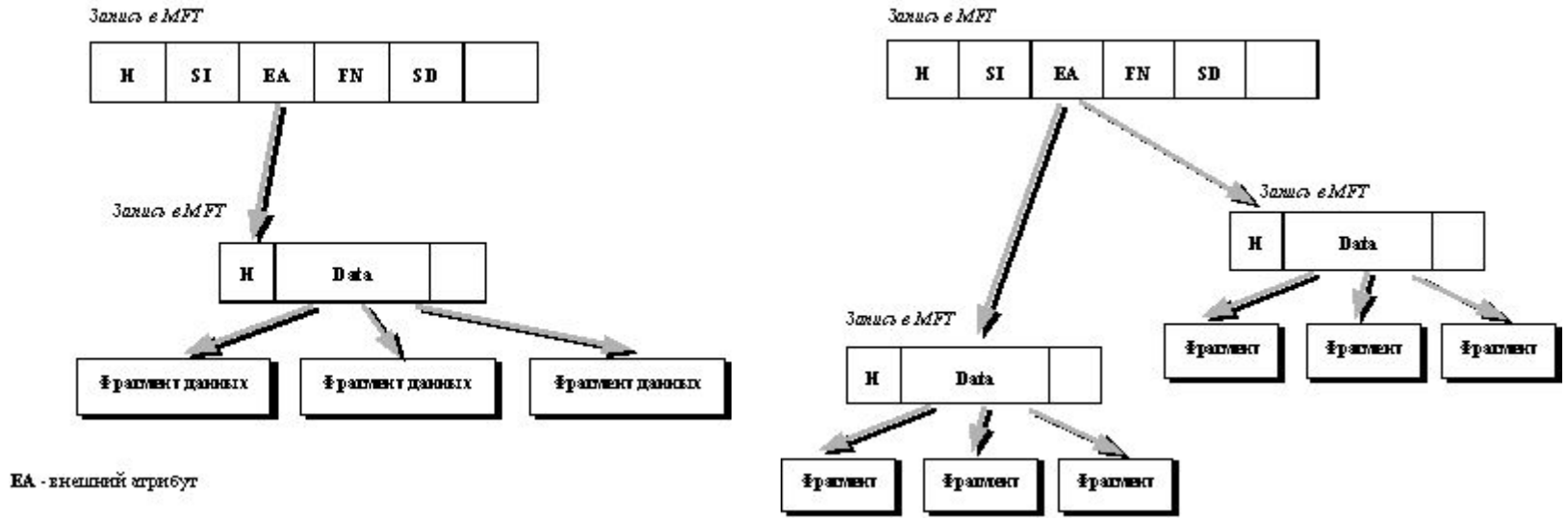


Нерезидентное хранение файлов среднего размера

- На предыдущем слайде показана запись MFT для файла среднего размера.
- Файл состоит из 3-х блоков кластеров: 20 – 23, 64 – 65, 80 – 82. Число таких блоков зависит от того насколько удачно ФС сумела найти место для хранения файла.

Нерезидентное хранение больших и сверхбольших 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. Эта ссылка называется внешним атрибутом (external attribute).

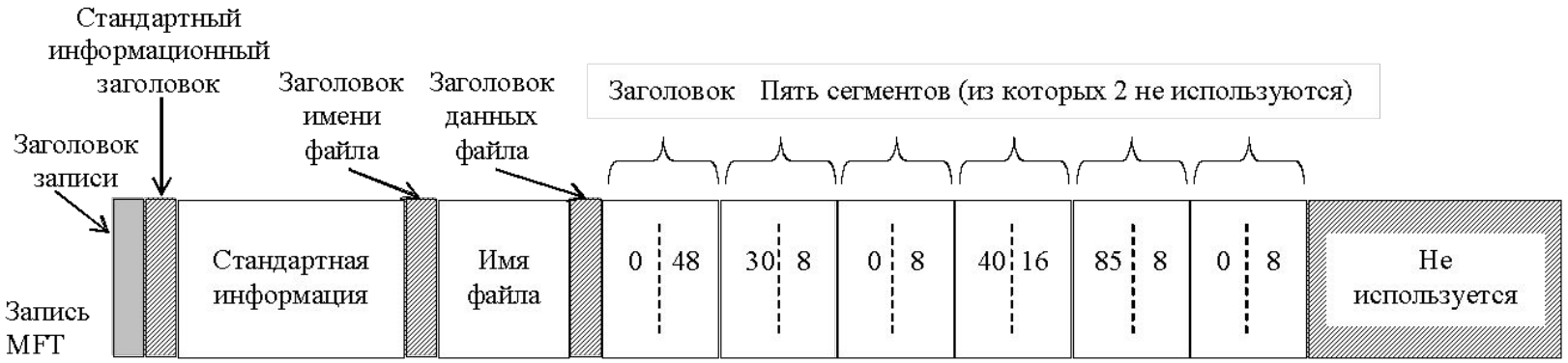


Сжатие файлов

- Файловая система NTFS поддерживает прозрачное сжатие файлов. Сжатие файлов производится следующим образом.
- Когда файловая система NTFS записывает на диск файл, помеченный для сжатия, она изучает первые 16 логических блоков файла, независимо от того, сколько сегментов на диске они занимают. Затем к этим блокам применяется алгоритм сжатия. Если полученные на выходе блоки могут поместиться в 15 или менее блоков, то сжатые данные записываются на диск, предпочтительно в виде одного сегмента. Если получить выигрыш не получается, то данные 16 блоков записываются без сжатия. Затем алгоритм повторяется для следующих 16 блоков и т.д.
- Сжатие файла частями по 16 блоков явилось компромиссом, если бы порции были меньше, то эффективность бы сжатия снизилась. Если размер блока был бы больше, то это замедлило бы произвольный доступ.

Сжатие файлов

Пример 48-блочного файла сжатого до 32 блоков



Запись MFT после сжатия файла

Сжатие файлов

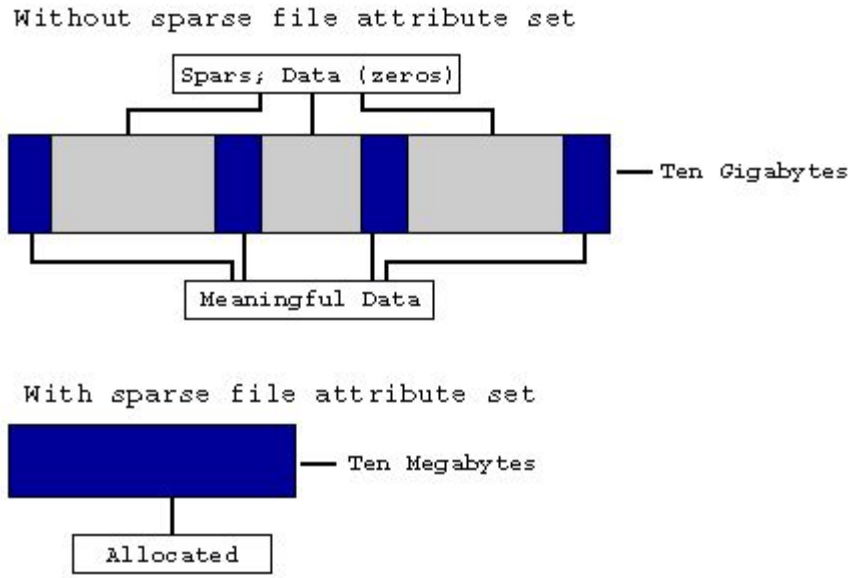
- На слайде показан файл, в котором первые 16 блоков успешно сжаты в 8 блоков, следующие 16 не могут быть сжаты, наконец, последние 16 блоков также успешно сжаты на 50%.
- Эти три части файла записаны в виде трех сегментов, информация о которых хранится в записи MFT. “Пропущенные” блоки обозначаются 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, но не прост. Например, для чтения блока 35 необходимо определить где находится этот блок и распаковать весь сегмент.

Разреженные файлы (sparse files)

- Другой тип сжатия известен как разреженные файлы.
- Если у вас есть файлы, которые содержат множество нулей (попросту говоря в файле есть "пустые области"), то NTFS позволяет сохранять пространство диска, давая таким файлам определение **sparse** (разреженный).
- Так вот при сохранении таких файлов система просто не выделяет место для пустых областей файла - в результате чего и достигается уменьшение размера файла.
  При обращении системы к частям, отмеченным как пустые, NTFS просто возвращает нулевые значения. При просмотре свойств файла система сообщит о зарезервированном для него размере, хотя фактический объем может занимать в сотни тысяч раз меньший объем.
- Разреженные файлы применяются, в частности, в журнале NTFS ($LogFile).

Разреженные файлы (sparse files)

- 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.

Многопоточные файлы

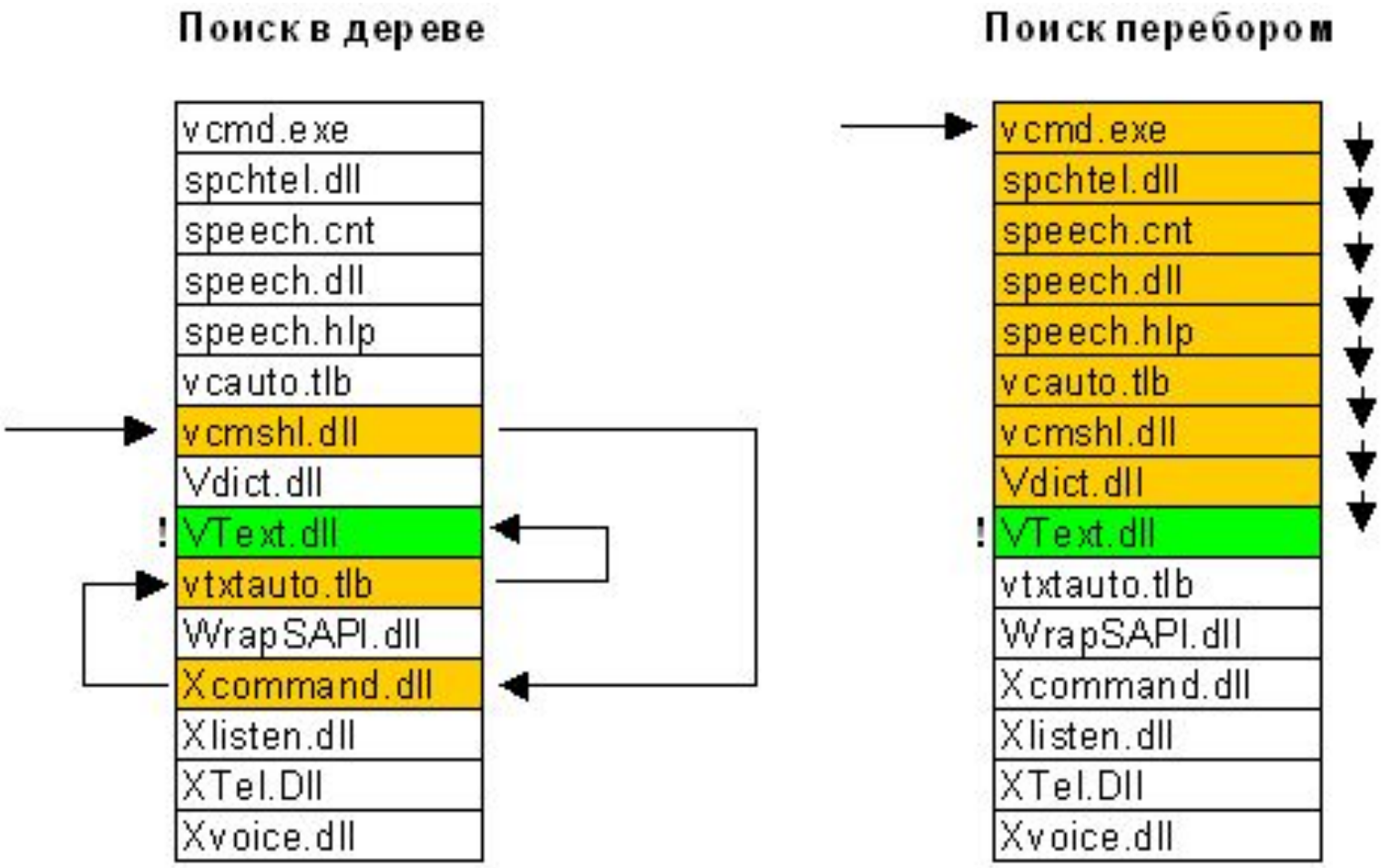
- Проведем следующий эксперимент. На машине Windows 2003 откроем окно командной строки. Перейдем в раздел NTFS (например, в папку, содержащую системные шрифты) и введем следующую команду (не делайте лишних пробелов!):
  C:\WINDOWS\Fonts>dir > New_Stream.TXT:New_Stream
- В результате выполнения этой команды 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

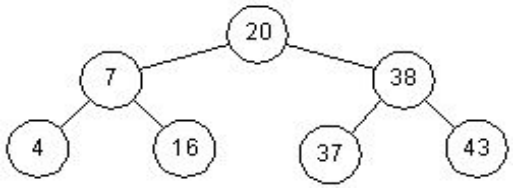
Каталоги NTFS

- 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, в каждом узле которого хранится несколько элементов), элементы которого сортируются по имени.
- 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).

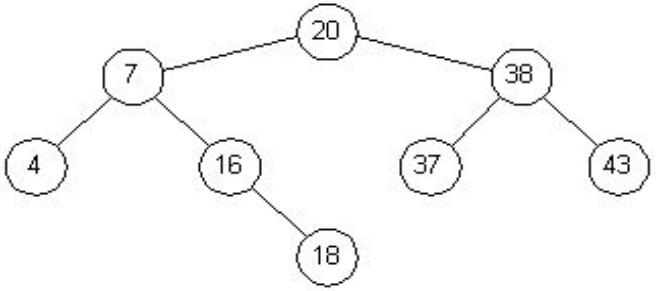
Простой и бинарный поиск



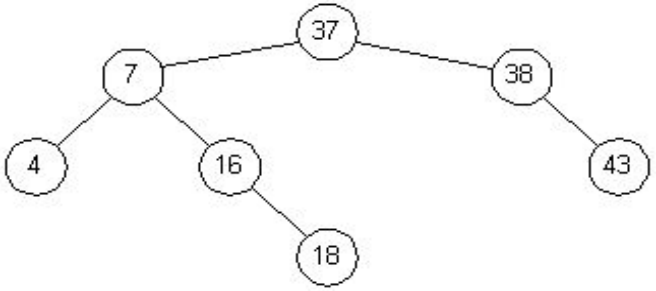
Бинарное дерево



Бинарное дерево



Бинарное дерево после добавления узла 18

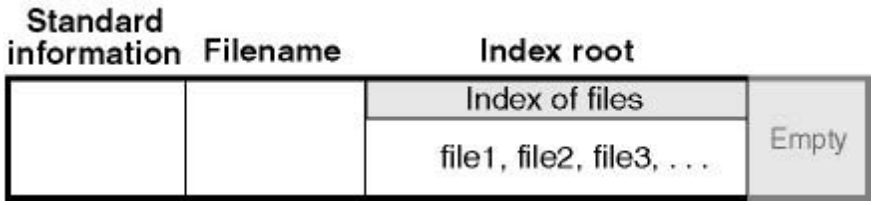


Бинарное дерево после удаления узла 20

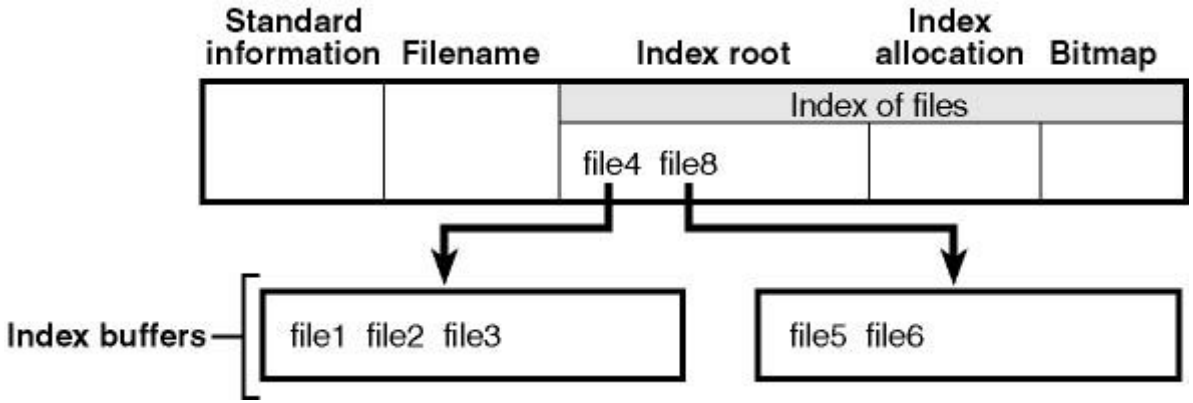
Хранение каталогов

- Небольшие записи каталогов находятся полностью внутри структуры MFT так же, как записи файла (копии атрибута File_Name файлов и подкаталогов). Для хранения используется атрибут $Index_Root (корневой индекс), который всегда резидентный !
- В том случае, когда атрибуты файла (или каталога) не умещаются в MFT и требуется выделение дополнительного пространства:
  - для хранения описания файлов выделяются нерезидентные индексные буферы (4 Кбайт), каждый из которых имеет виртуальный номер кластера (virtual clusters numbers, VCN) для сквозной нумерации кластеров в рамках одной записи MFT;
  - корневой индекс хранит корень дерева B+ и ссылки (VCN) на индексные буферы.
  - соответствие между VCN и LCN хранится в атрибуте каталога $Index_Allocation. Примечание: Логические номера кластеров (LCN), представляют последовательность кластеров всего тома

Пример хранения каталогов

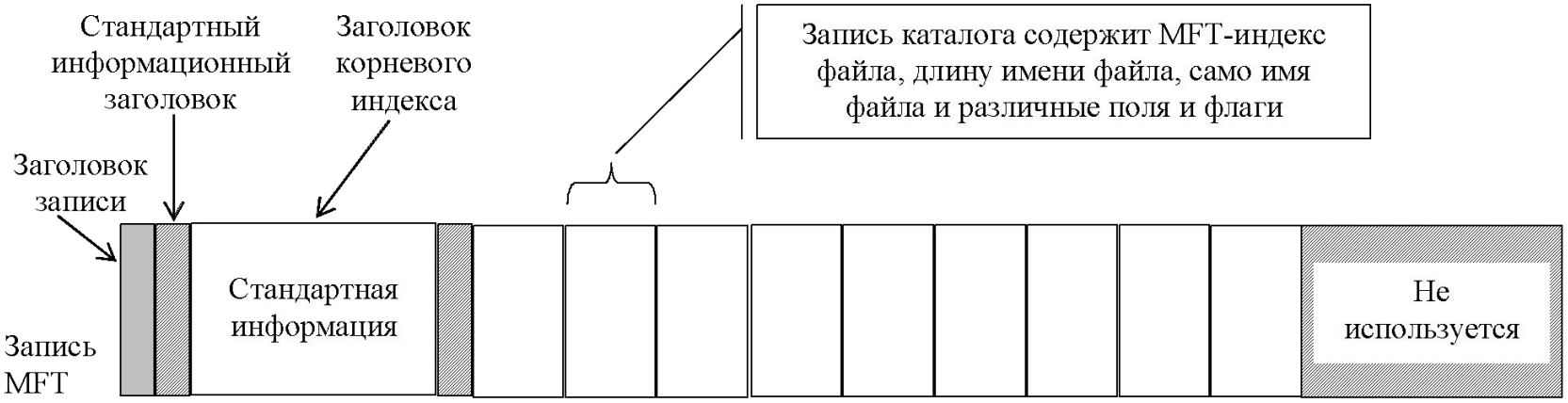


а) запись MFT для небольшого каталога (резидентное хранение)



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

Запись MFT для небольшого каталога

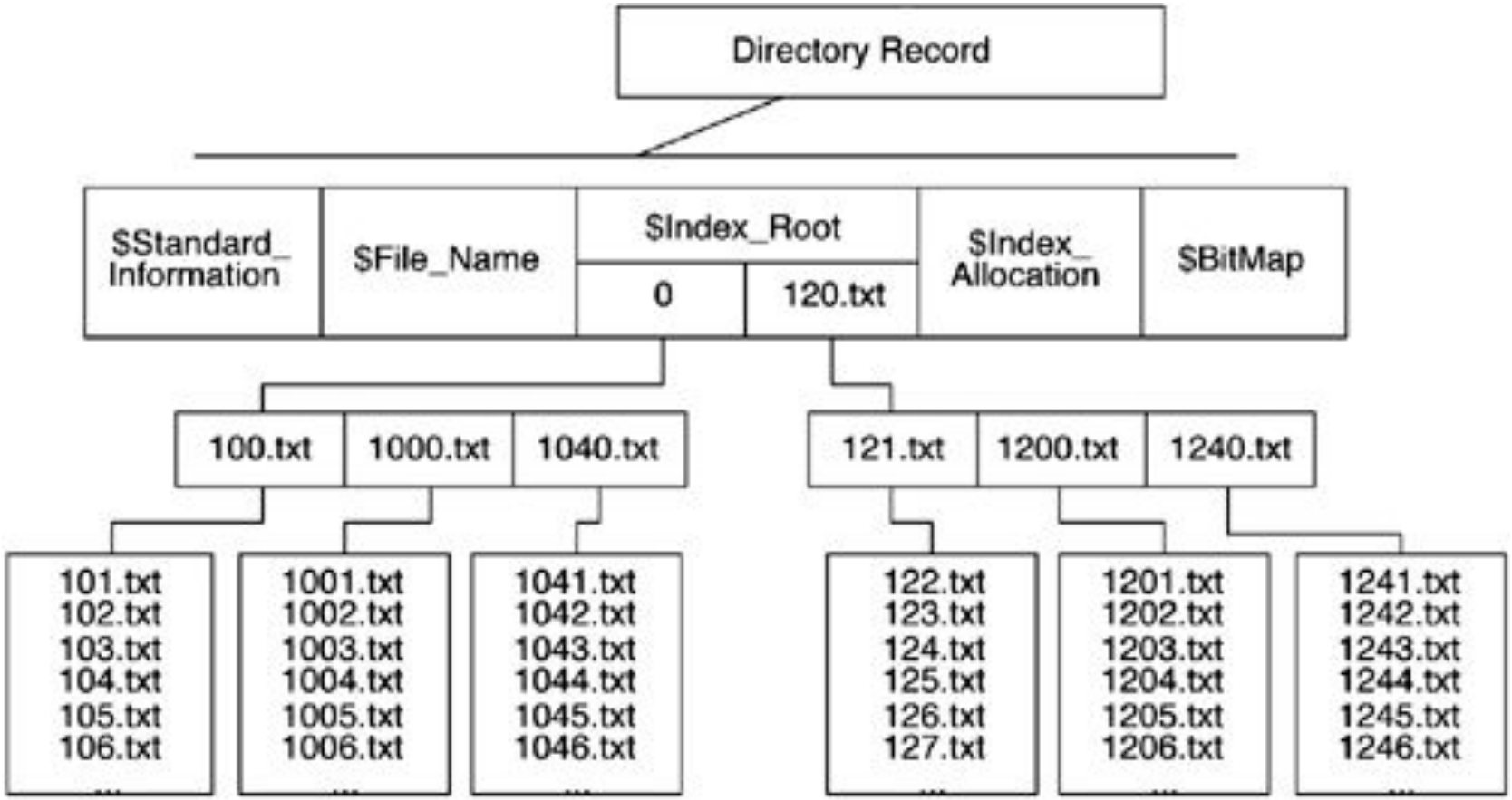


Запись MFT для небольшого каталога содержит несколько каталоговых записей, каждая из которых описывает файл или каталог.

Фиксированная запись содержит индекс записи MFT файла, длину имени файла, а также другие разнообразные поля и флаги.

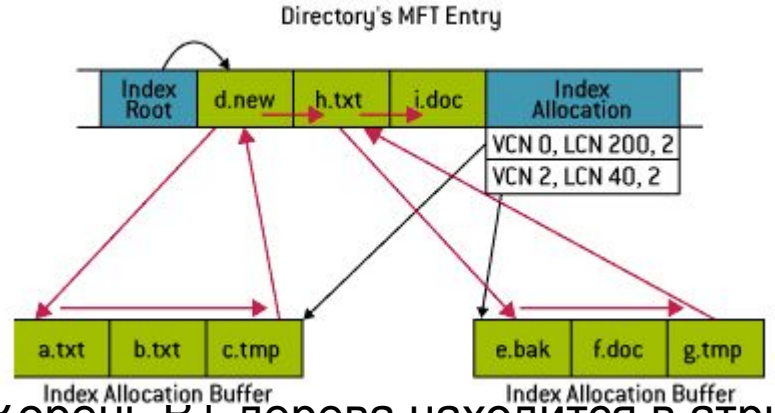
Поиск файла в каталоге по имени состоит в последовательном переборе всех имен файлов.

Хранение корневого каталога



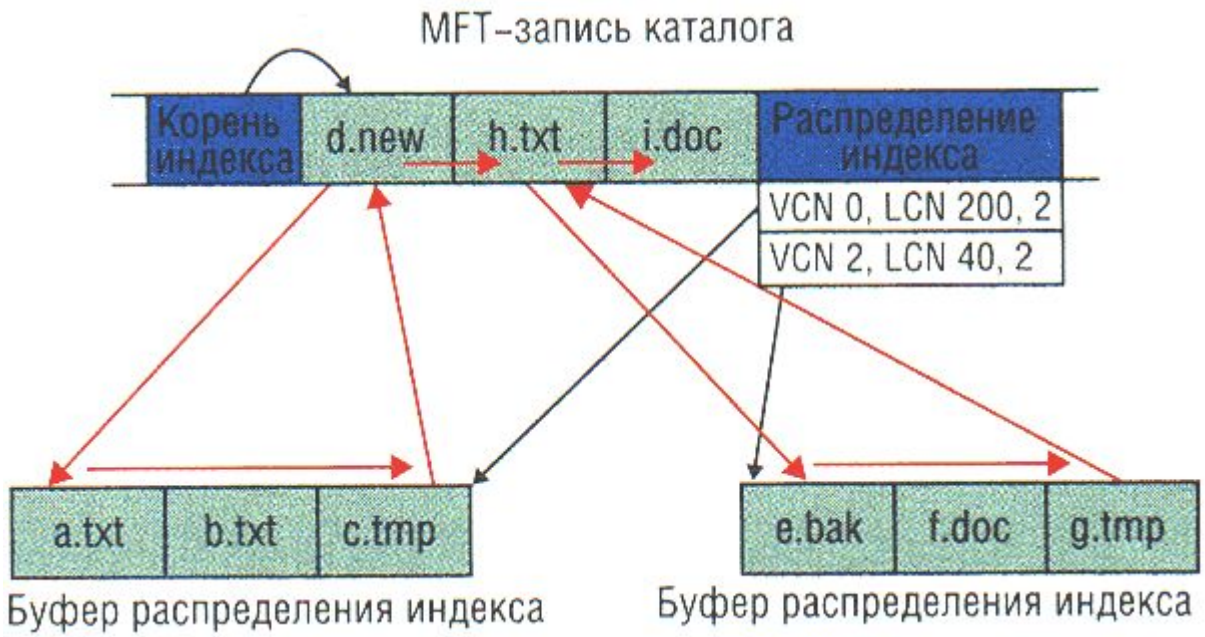
Пример нерезидентного хранения каталогов

- На рисунке показана запись MFT каталога, в трех узлах которой содержится девять элементов, по три в каждом узле.



- Корень B+ дерева находится в атрибуте index root (корень индекса). В записи MFT каталога девять элементов не умещается, поэтому некоторые элементы приходится хранить в другом месте. Для этого NTFS выделяет два буфера размещения индексов (index allocation) для хранения двух записей. Обычно, корень индекса и буферы размещения индексов могут хранить элементы для более чем трех файлов, в зависимости от длины имен.
- Красные стрелки указывают, что элементы NTFS хранятся в алфавитном порядке

Пример нерезидентного хранения каталогов



Пример нерезидентного хранения каталогов

- Если запустить программу, которая открывает файл e.bak в показанном на рисунке каталоге, то NTFS читает атрибут индексного корня, содержащий элементы для d.new, h.txt и i.doc, и сравнивает строку e.bak с именем первого элемента, d.new. NTFS делает вывод, что алфавитный номер e.bak больше, чем d.new, и переходит к следующему элементу — h.txt. Повторив операцию сравнения, NTFS выясняет, что алфавитный номер e.bak меньше, чем h.txt. Затем NTFS отыскивает в записи каталога h.txt номер виртуального кластера (virtual cluster number, VCN) индексного буфера, содержащего элементы каталога, алфавитные номера которых меньше, чем h.txt, но больше, чем d.new. VCN представляет собой порядковый номер кластера в файле или каталоге. На основании информации о размещении кластеров NTFS преобразует VCN в логический номер кластера (Logical Cluster Number, LCN), т. е. номер кластера относительно начала тома. Если элемент каталога для h.txt не содержит VCN индексного буфера, NTFS делает вывод, что каталог h.txt не содержит файла e.bak и сообщает о неудачном завершении поиска.
- Получив VCN начального кластера индексного буфера, NTFS читает буфер размещения индексов и просматривает его в поисках совпадений. На рисунке первый же элемент индексного буфера совпадает с критерием поиска, и NTFS читает номер записи MFT e.bak из элемента каталога e.bak. В элементах каталога хранится и другая информация: в частности, временные отметки (например, время создания и последнего изменения), размер и атрибуты. NTFS хранит эту информацию и в записи MFT файла, но, благодаря дублированию информации в элементе каталога, читать запись MFT файла при составлении списков каталогов и выполнении простых файловых запросов не требуется.