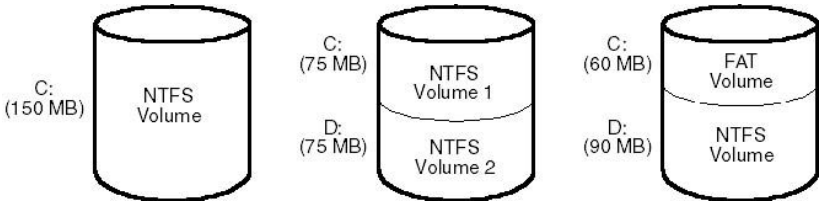
Файловая система NTFS

- 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.

- 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.

- *Простой том (simple)*
- *Составной (набор) 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*) 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*) – том, состоящий из нескольких разделов, по которым равномерными блоками распределены данные. NT использует блоки данных по 64 Кбайт. Первый блок данных размером в 64 Кбайт хранится на первом разделе, вторые 64 Кбайт на втором и т.д. по кругу, возвращаясь к первому разделу. 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 volume*) – это чередующийся набор с дополнительным блоком данных размером в 64 Кбайт для разделов, образующих чередующийся набор. Дополнительный блок содержит информацию о четности, которую система NT может использовать при восстановлении данных, расположенных на одном из разделов чередующегося набора, в случае выхода из строя диска, где был расположен раздел.

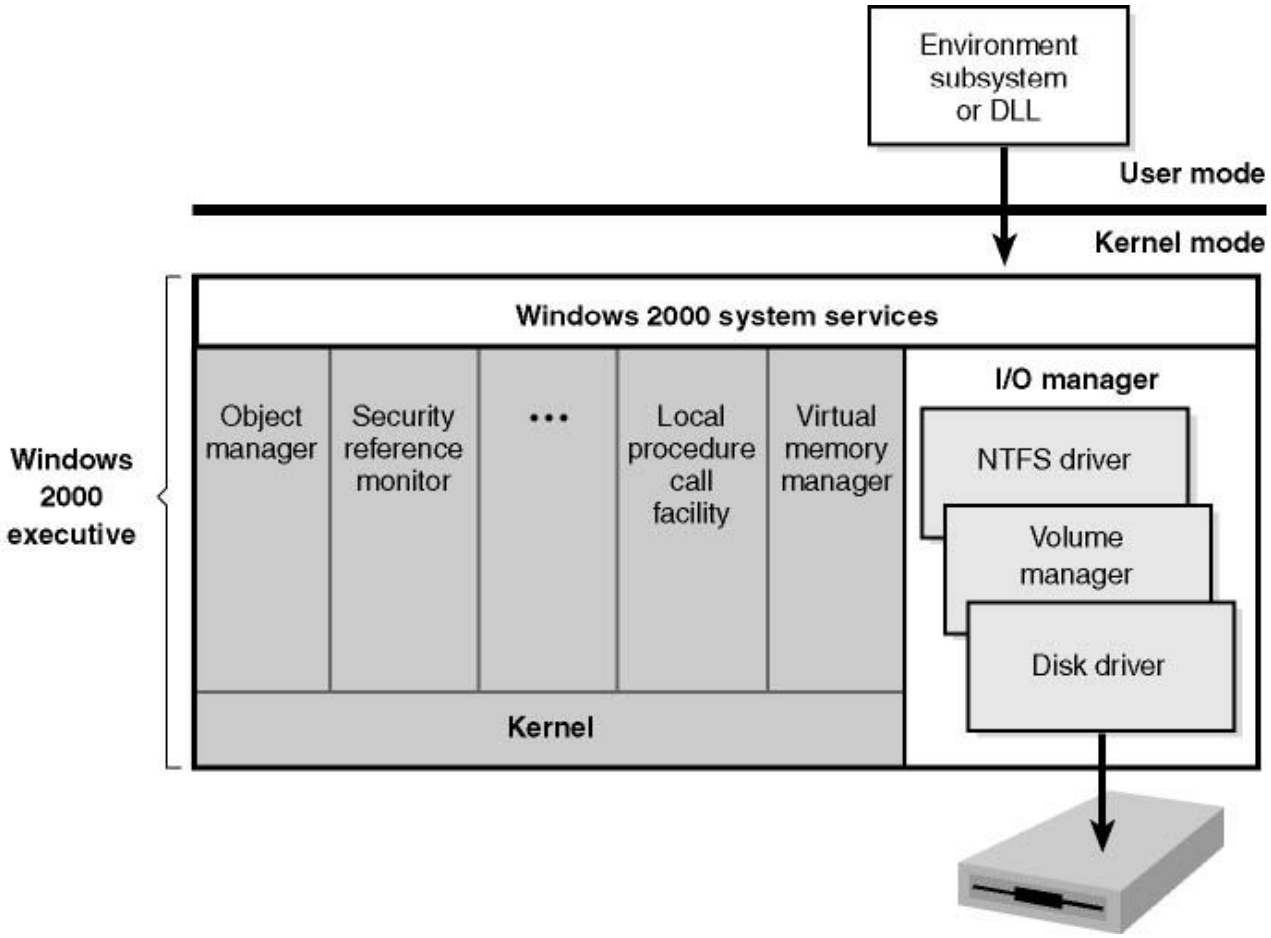
- В разделе HKEY_LOCAL_MACHINE\SYSTEM\MountedDevices системного реестра хранится информация о базовых дисках.
- Внутреннее имя имеет форму \??\Volume{XX-XX-XX-XX}, где X — числа, образующие глобальный уникальный ID (GUID), присвоенный тому операционной системой.
- Системная утилита mountvol

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

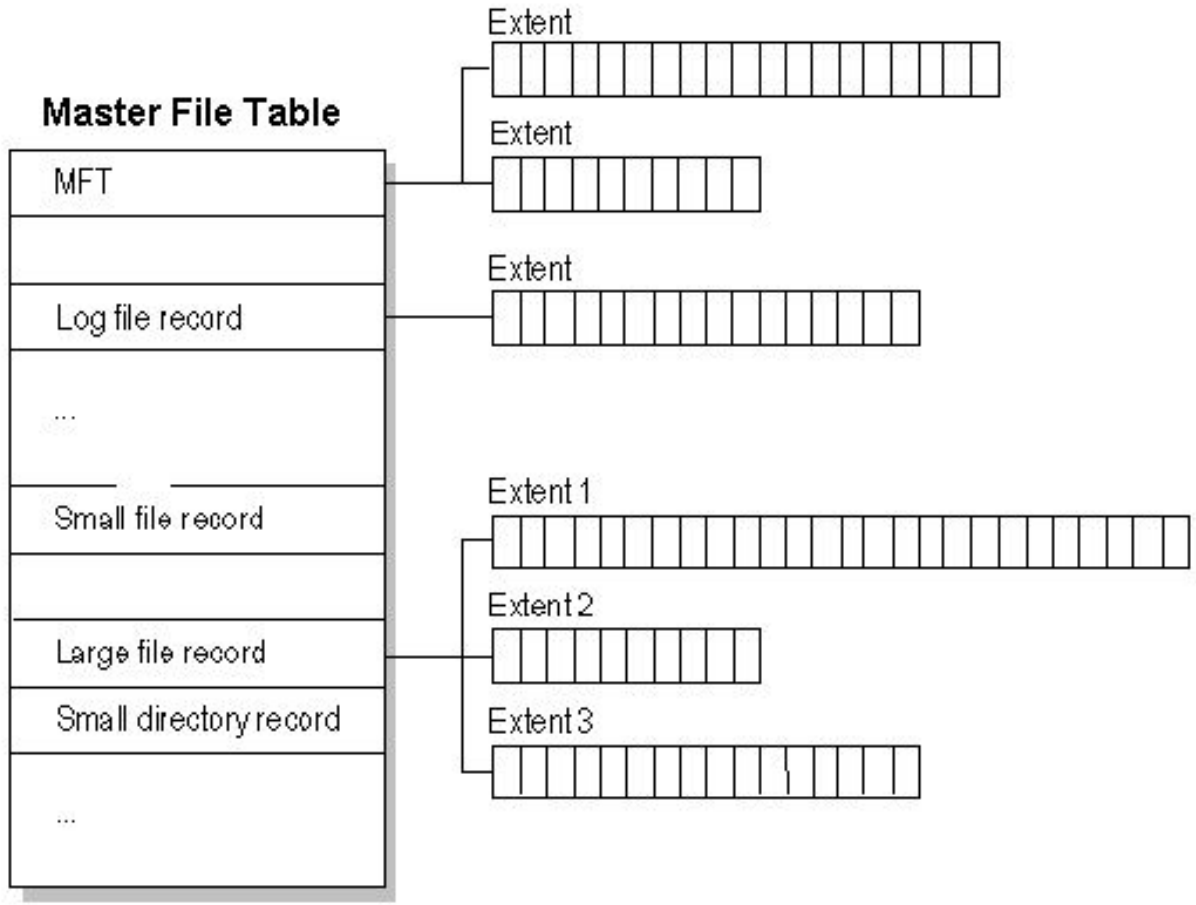
Размер логического диска	Рекомендуемый размер кластера
512 MB и менее	512 б
513 Мб – 1 Гб	1 Кб
1 Гб – 2 Гб	2 Кб
Более 2 Гб	4 Кб

Пользователь может определить размер кластера при форматировании тома NTFS **/a:<size>**, т.е.
format d: /a:1024 /fs:ntfs

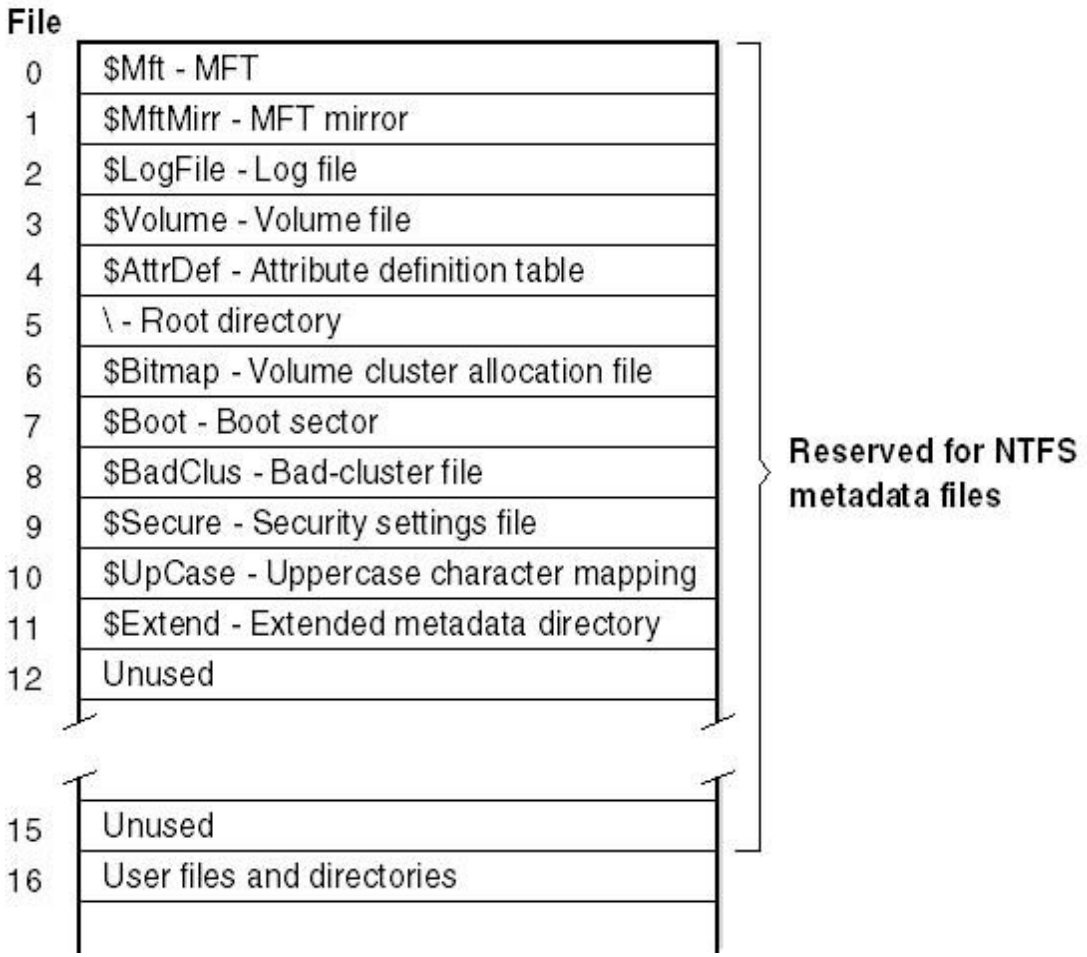
Однако NTFS-сжатие не поддерживается для кластеров, размер которых больше 4 КБ.


Environment subsystem or DLL
User mode
Kernel mode
Windows 2000 system services
Windows 2000 executive
Object manager
Security reference monitor
• • •
Local procedure call facility
Virtual memory manager
I/O manager
NTFS driver
Volume manager
Disk driver
Kernel

NTFS поддерживает размеры кластеров - от 512 байт до 64 Кбайт


Master File Table
MFT
Log file record
...
Small file record
Large file record
Small directory record
...
Extent
Extent
Extent
Extent 1
Extent 2
Extent 3

- 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.


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

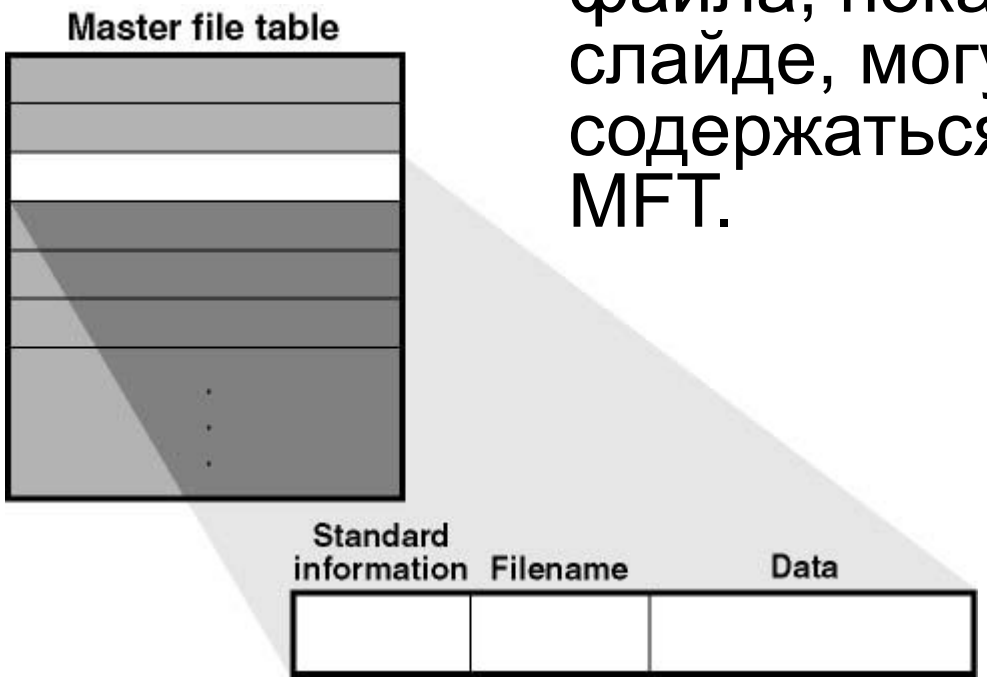
$MFT	список содержимого тома NTFS
$MFTmirr	копия первой записи MFT
$LogFile	файл поддержки журналирования шагов транзакций
$Volume	служебная информация - метка тома, версия файловой системы, т.д.
$AttrDef	список стандартных атрибутов файлов на томе
$.	корневой каталог
$Bitmap	карта свободного места тома,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

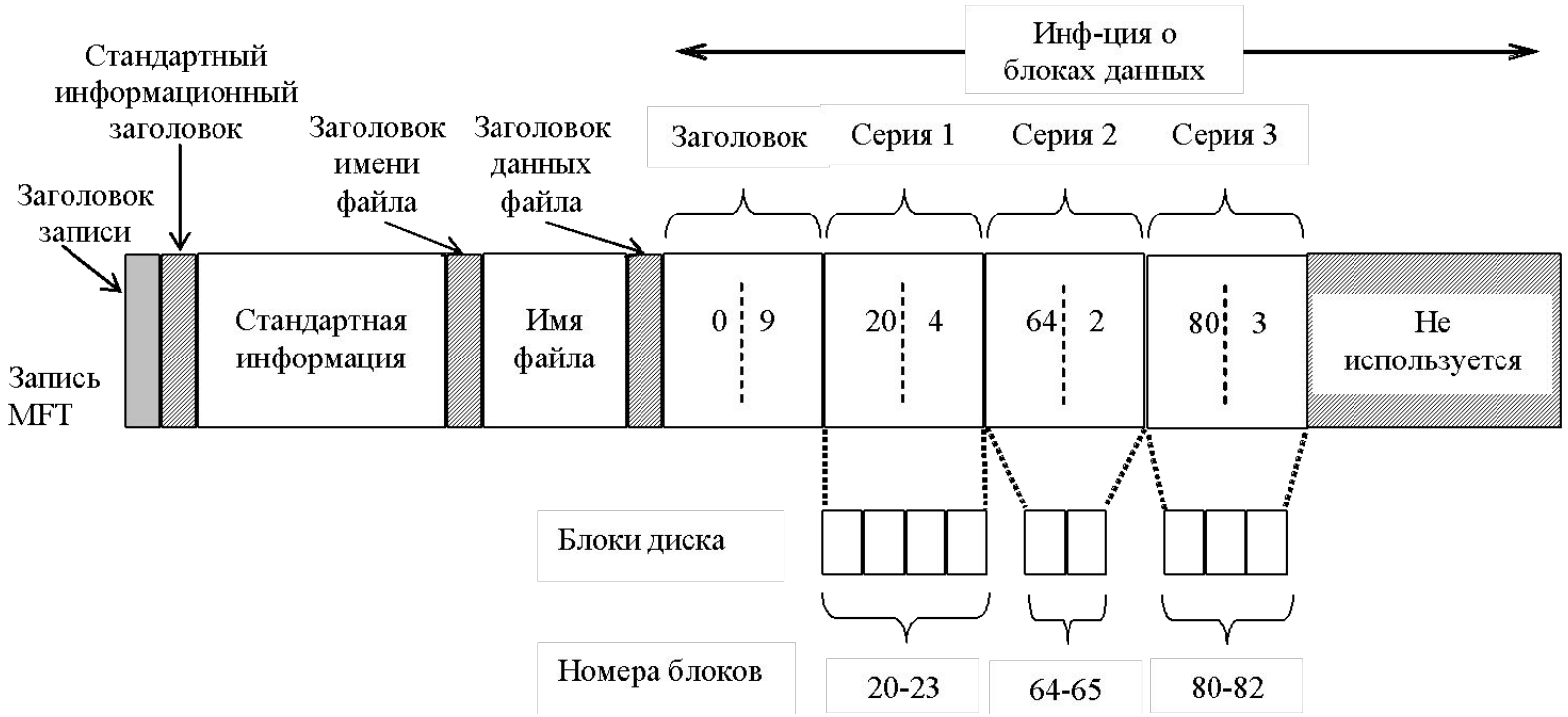
$BadClus	Список всех плохих кластеров тома. Кластер считается плохим, если в нем есть один плохой сектор
$Secure	База данных атрибутов безопасности. Применяется только в NTFS версии 5.0 в среде Microsoft Windows 2000
$Upcase	файл - таблица соответствия заглавных и прописных букв в имен файлов на текущем томе.
$Extend	Файл хранит расширенную информацию файловой системы NTFS версии 5.0, применяемой в среде Microsoft Windows 2000, такую как дисковые квоты, точки монтирования и т.д.

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

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

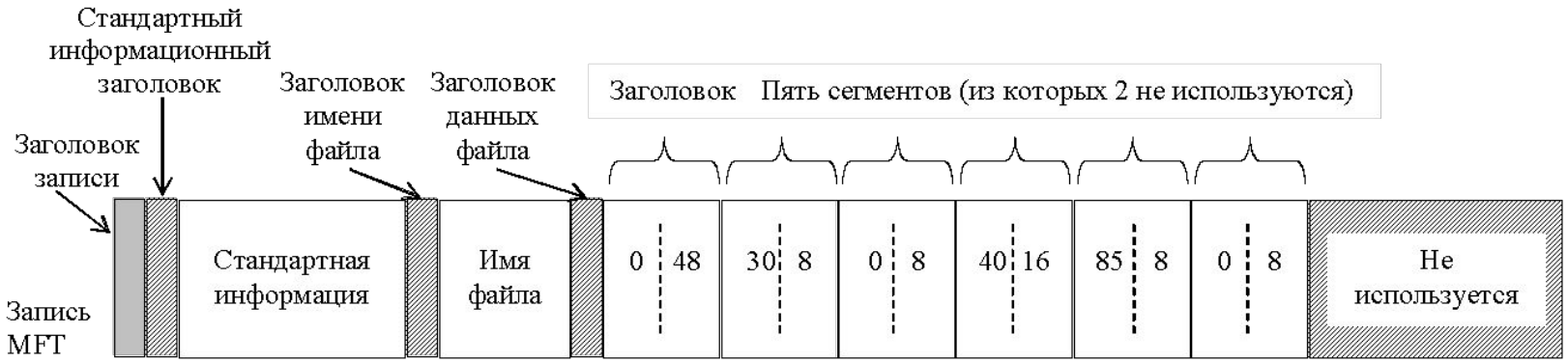
Небольшие файлы и каталоги (меньше 1 Кбайт), типа файла, показанного на слайде, могут полностью содержаться внутри записи MFT.





Запись MFT для файла среднего размера

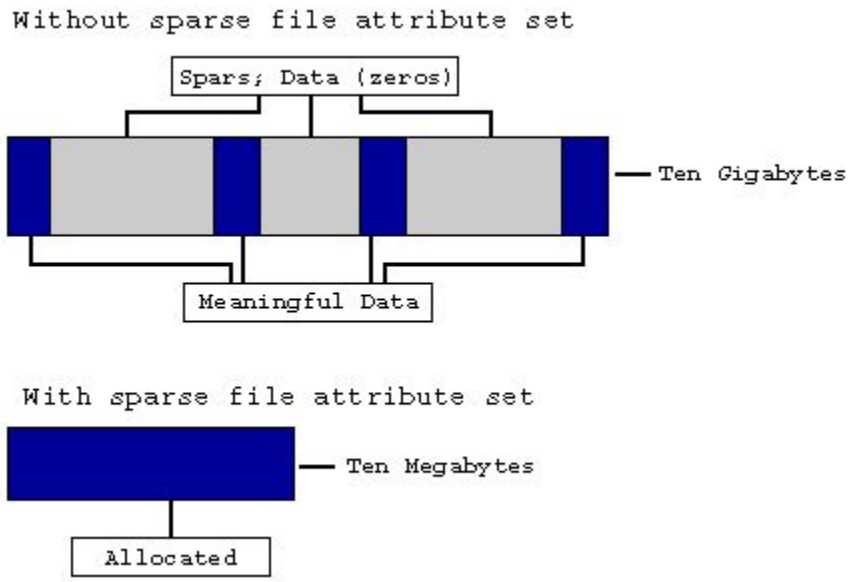
Пример 48-блочного файла сжатого до 32 блоков



Запись MFT после сжатия файла

- 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 (aka сжатие). 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 изменений NTFS.

Разреженные файлы конвертируются с помощью следующей команды: fsutil sparse.

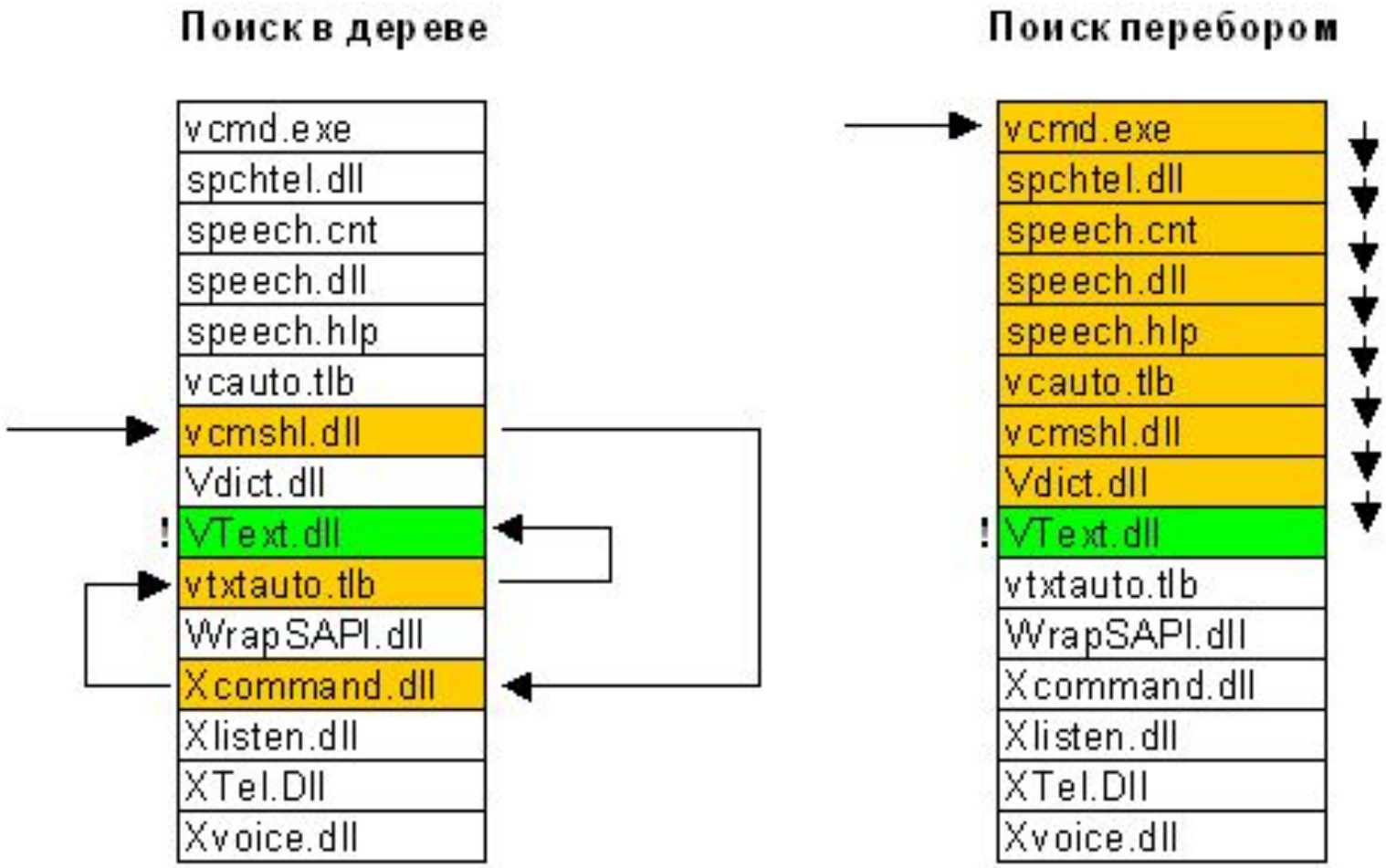


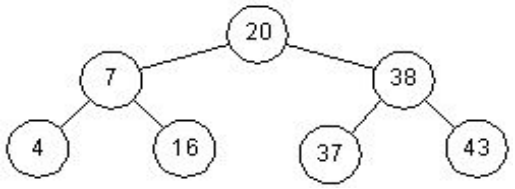
- При необходимости в одном файле, записанном на диске NTFS, можно хранить несколько потоков информации. Это позволяет, в частности, снабжать файлы документов дополнительной информацией, хранить в одном файле несколько версий документов (например, на разных языках), хранить в отдельных потоках одного 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.

- 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. В именах потоков, как и в именах файлов, имеет значение регистр символов.
- Для вывода содержимого именованного потока воспользуемся:

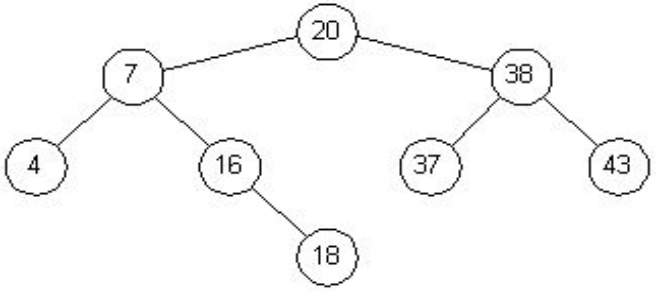
  C:\WINDOWS\Fonts>more < New_Stream.TXT:New_Stream

- 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.
- Вот что это означает: для поиска файла с данным именем в линейном каталоге, таком, например, как у FAT-а, операционной 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. Вопрос, на который бинарное дерево способно дать ответ, таков: в какой группе, относительно данного элемента, находится искомое имя - выше или ниже? Мы начинаем с такого вопроса к среднему элементу, и каждый ответ сужает зону поиска в среднем в два раза. Файлы, скажем, просто отсортированы по алфавиту, и ответ на вопрос осуществляется очевидным способом - сравнением начальных букв. Область поиска, суженная в два раза, начинает исследоваться аналогичным образом, начиная опять же со среднего элемента.
- 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 10-ти (2^10 = 1024). Log2(N)

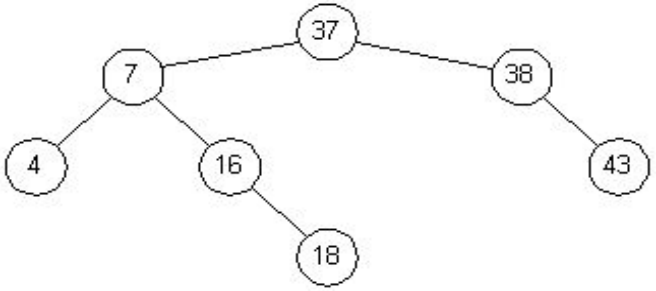

Поиск в дереве
vcmd.exe
spchtel.dll
speech.cnt
speech.dll
speech.hlp
vcauto.tlb
vcmshl.dll
Vdict.dll
!
VText.dll
vtxtauto.tlb
WrapSAPI.dll
Xcommand.dll
Xlisten.dll
XTel.Dll
Xvoice.dll
Поиск перебором
vcmd.exe
spchtel.dll
speech.cnt
speech.dll
speech.hlp
vcauto.tlb
vcmshl.dll
Vdict.dll
!
VText.dll
vtxtauto.tlb
WrapSAPI.dll
Xcommand.dll
Xlisten.dll
XTel.Dll
Xvoice.dll



Бинарное дерево

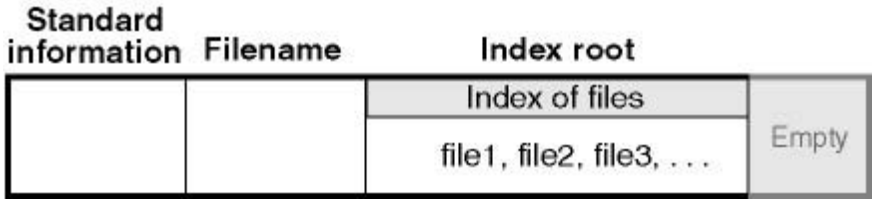


Бинарное дерево после добавления узла 18

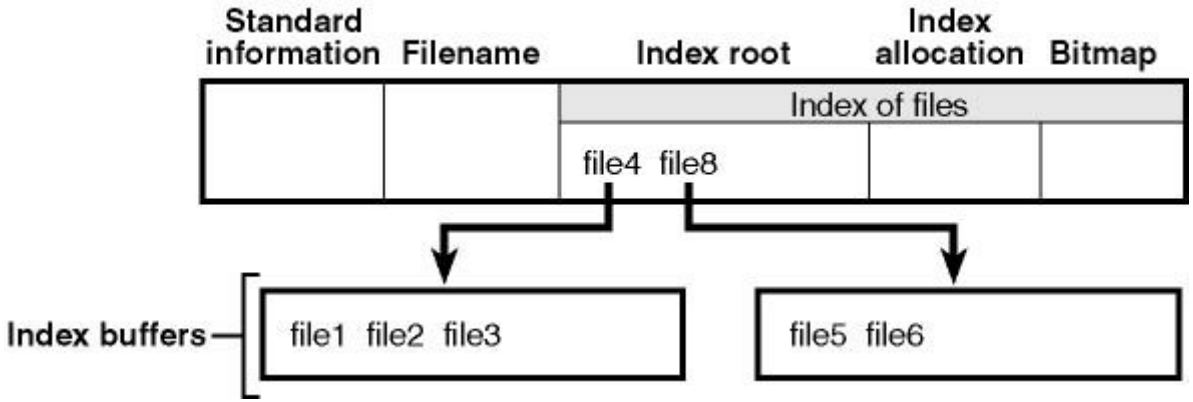


Бинарное дерево после удаления узла 20

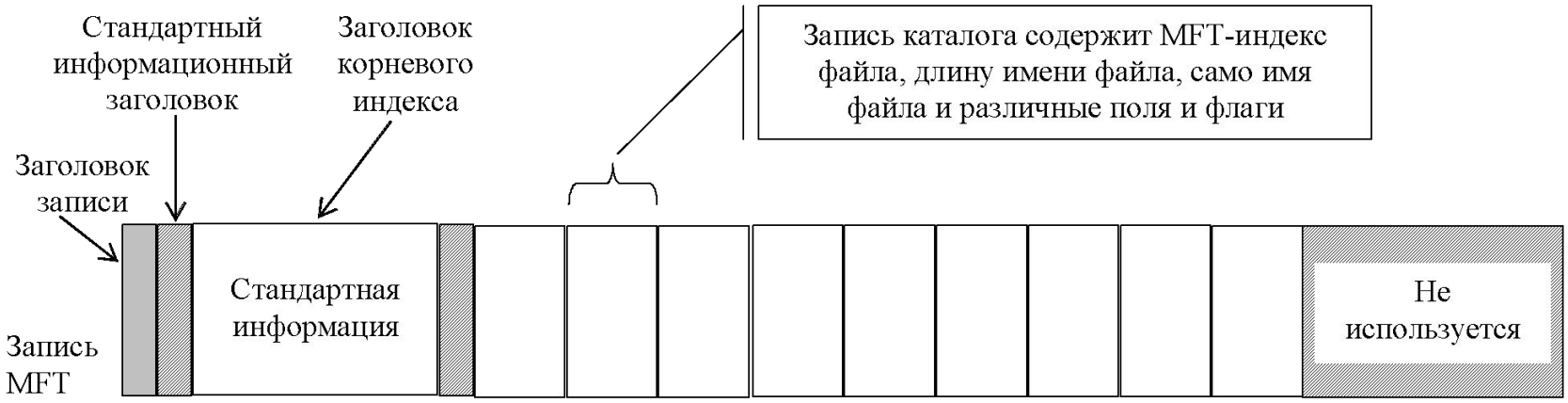
- Небольшие записи каталогов находятся полностью внутри структуры MFT так же, как записи файла (копии атрибута File_Name файлов и подкаталогов). Для хранения используется атрибут $Index_Root (корневой индекс), который всегда резидентный !
- В том случае, когда атрибуты файла (или каталога) не умещаются 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);
  - 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



а) запись MFT для небольшого каталога (резидентное хранение)



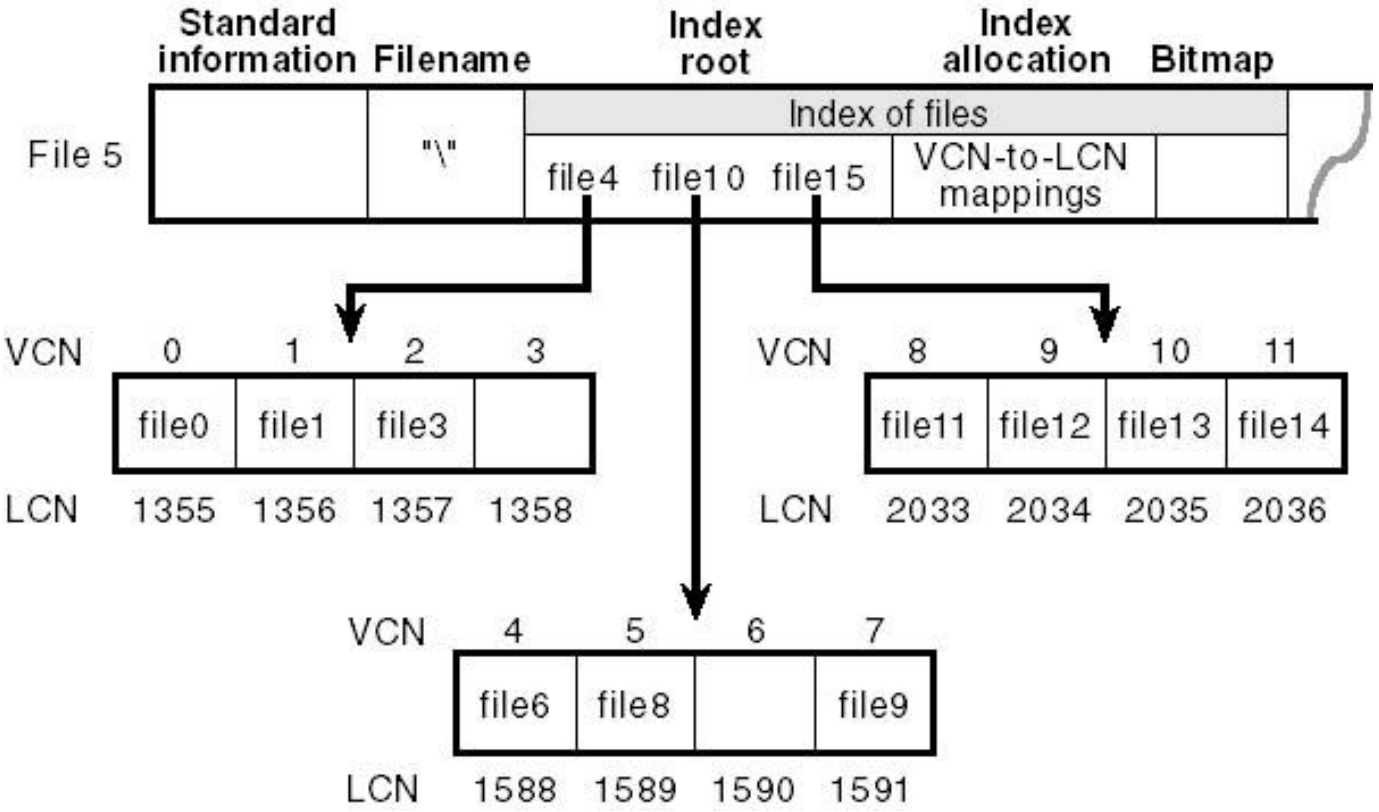
б) запись MFT для “большого” каталога (нерезидентное хранение)



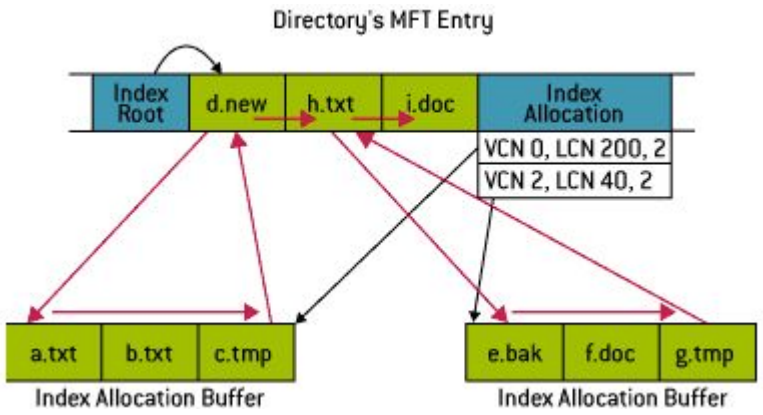
Запись MFT для небольшого каталога содержит несколько каталоговых записей, каждая из которых описывает файл или каталог.

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

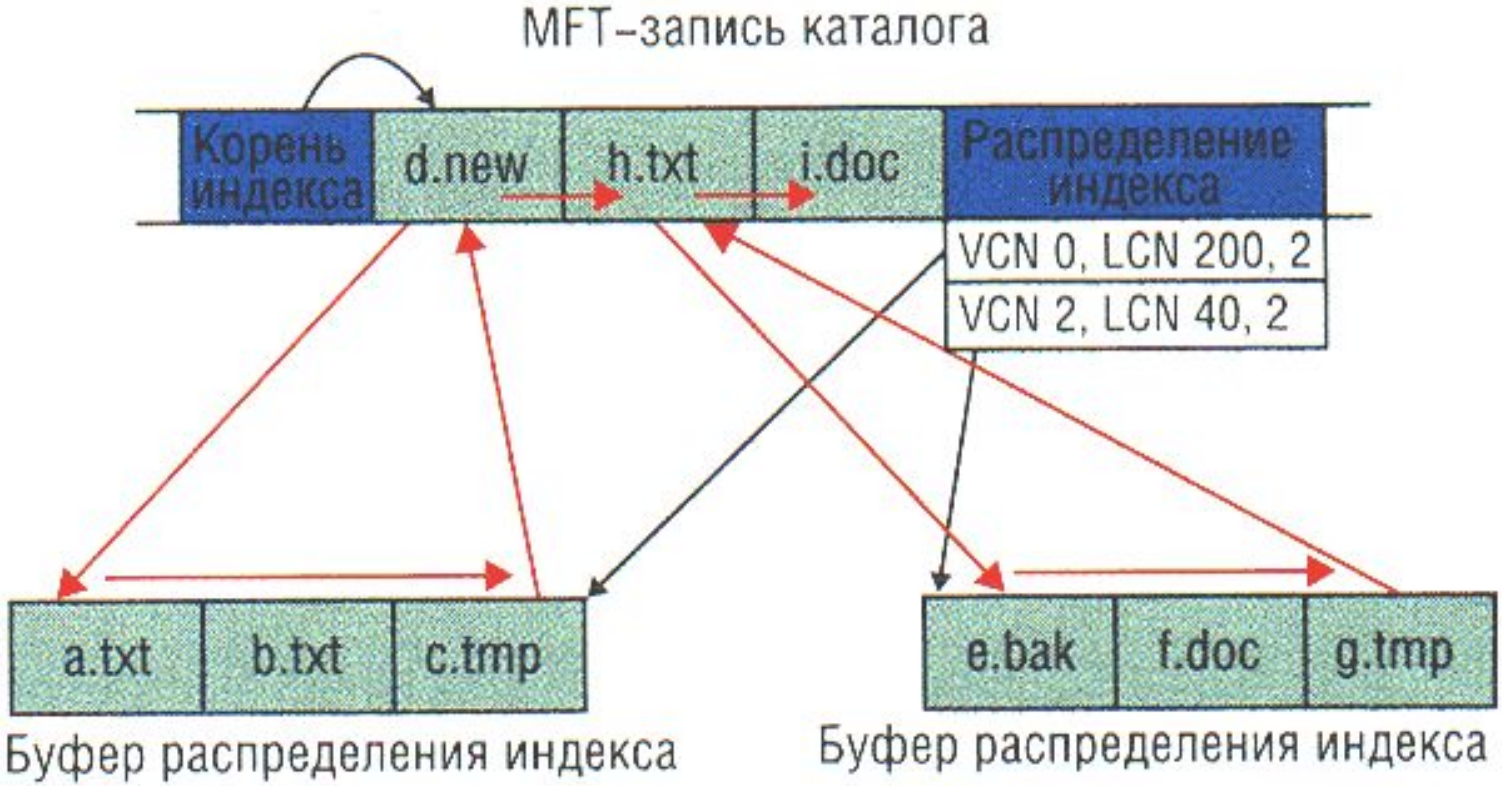
Поиск файла в каталоге по имени состоит в последовательном переборе всех имен файлов.


Standard information
Filename
Index root
Index allocation
Bitmap
File 5
"\"
Index of files
file4
file10
file15
VCN-to-LCN mappings
VCN 0 1 2 3
file0
file1
file3
LCN 1355 1356 1357 1358
VCN 8 9 10 11
file11
file12
file13
file14
LCN 2033 2034 2035 2036
VCN 4 5 6 7
file6
file8
file9
LCN 1588 1589 1590 1591

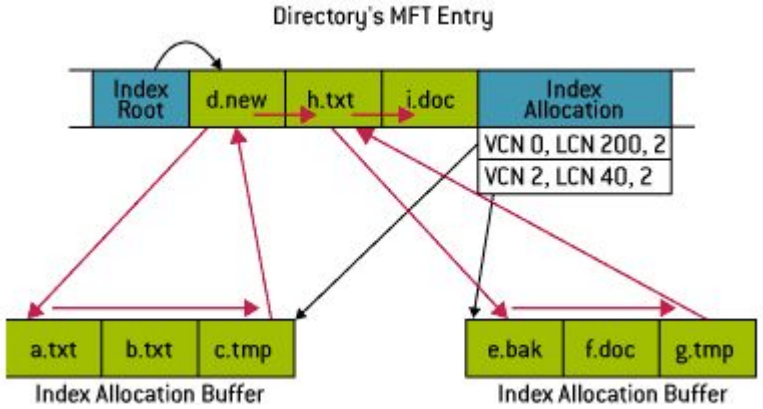
- На рисунке показана запись MFT каталога, в трех узлах которой содержится девять элементов, по три в каждом узле.
- Корень B+ дерева находится в атрибуте index root (корень индекса). В записи MFT каталога девять элементов не умещается, поэтому некоторые элементы приходится хранить в другом месте. Для этого NTFS выделяет два буфера размещения индексов (index allocation) для хранения двух записей (как правило, корень индекса и буферы размещения индексов мо-гут хранить элементы для более чем трех файлов, в зависимости от длины имен).



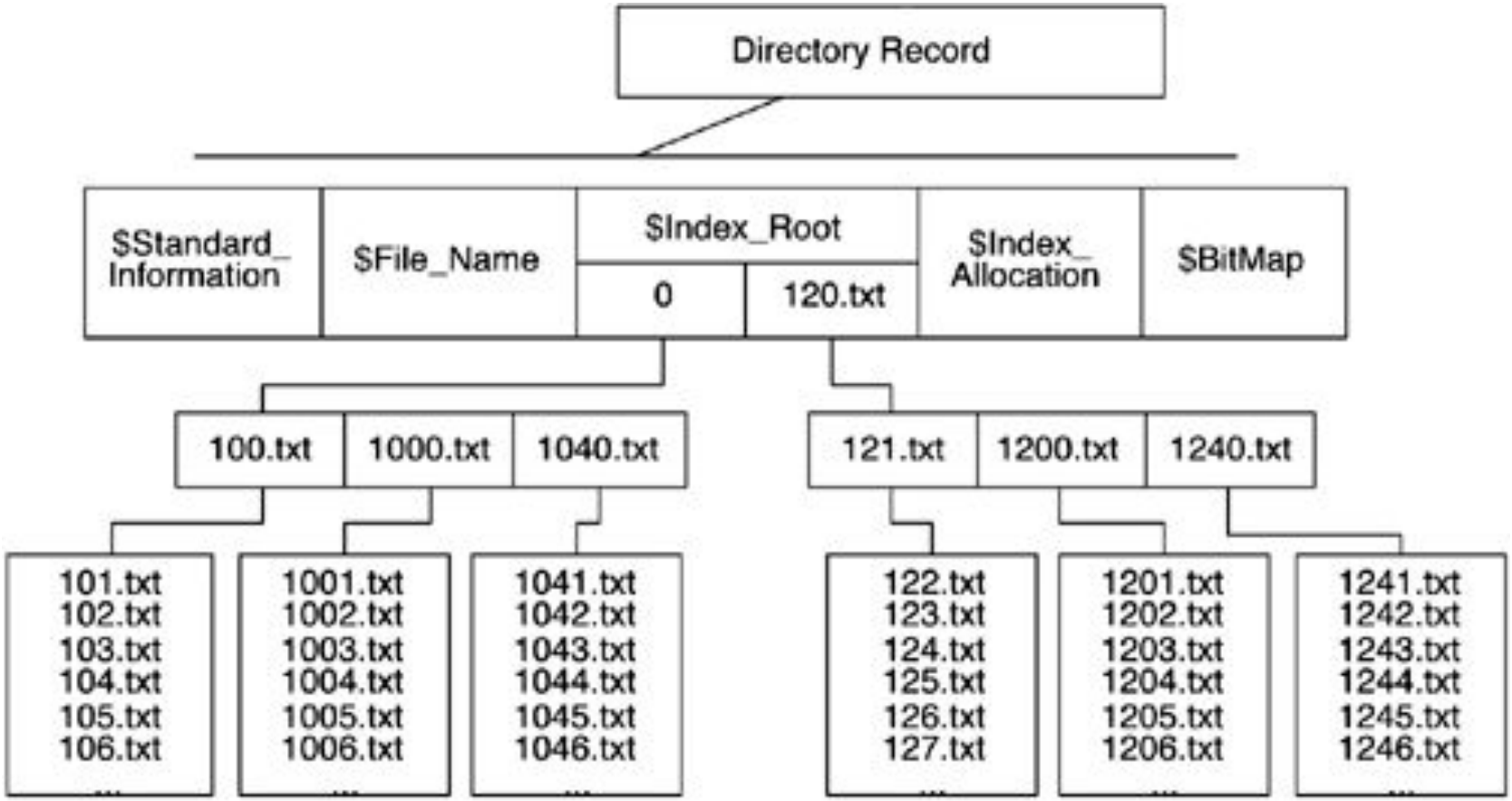
- Размер записи MFT — 1 Кбайт, а размер буферов размещения индексов — 4 Кбайт.
- Красные стрелки указывают, что элементы NTFS хранятся в алфавитном порядке.


MFT-запись каталога
Корень индекса
d.new
h.txt
i.doc
Распределение индекса
VCN 0, LCN 200, 2
VCN 2, LCN 40, 2
a.txt
b.txt
c.tmp
e.bak
f.doc
g.tmp
Буфер распределения индекса
Буфер распределения индекса

- 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.



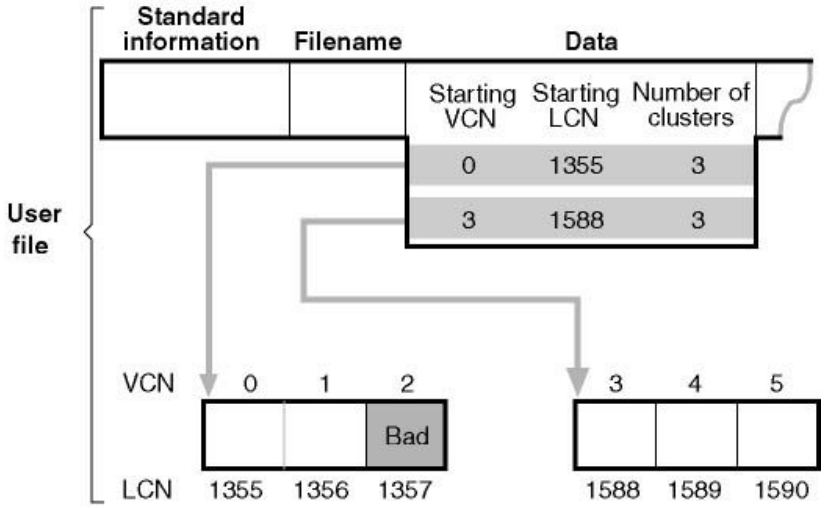
- Получив VCN начального кластера индексного буфера, NTFS читает буфер размещения индексов и просматривает его в поисках совпадений. На рисунке первый же элемент индексного буфера совпадает с критерием поиска, и NTFS читает номер записи MFT e.bak из элемента каталога e.bak. В элементах каталога хранится 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.

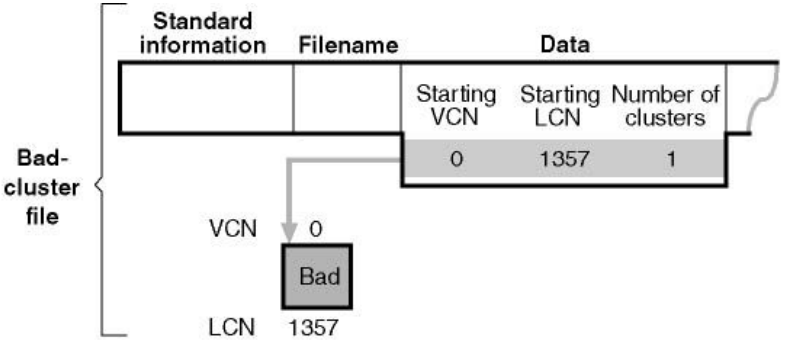

Directory Record
$Standard_ Information
$File_Name
$Index_Root
0
120.txt
$Index_ Allocation
$BitMap
100.txt
1000.txt
1040.txt
121.txt
1200.txt
1240.txt
101.txt 102.txt 103.txt 104.txt 105.txt 106.txt ...
1001.txt 1002.txt 1003.txt 1004.txt 1005.txt 1006.txt ...
1041.txt 1042.txt 1043.txt 1044.txt 1045.txt 1046.txt ...
122.txt 123.txt 124.txt 125.txt 126.txt 127.txt ...
1201.txt 1202.txt 1203.txt 1204.txt 1205.txt 1206.txt ...
1241.txt 1242.txt 1243.txt 1244.txt 1245.txt 1246.txt ...

NTFS является *восстанавливаемой* ФС и поддерживает следующие технологии защиты целостности данных:

- *Горячая фиксация* - позволяет файловой системе при возникновении ошибки из-за плохого кластера записать информацию в другой кластер и отметить сбойный в качестве плохого.
- *Механизм транзакций* - каждая операция ввода-вывода, которая изменяет файл на разделе NTFS, рассматривается файловой системой как транзакция и может выполняться только как неделимый блок.

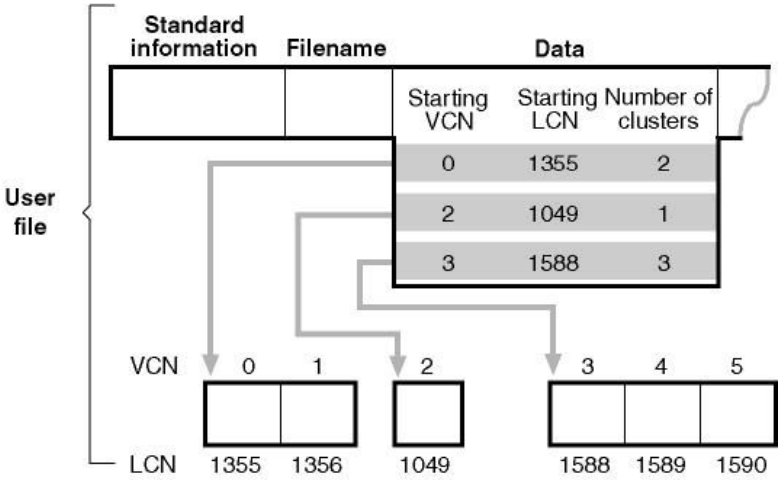
Система восстановления NTFS гарантирует корректность **файловой системы**, а не ваших данных.





а) MFT-запись файла с плохим кластером;

б) исправленная MFT-запись файла;

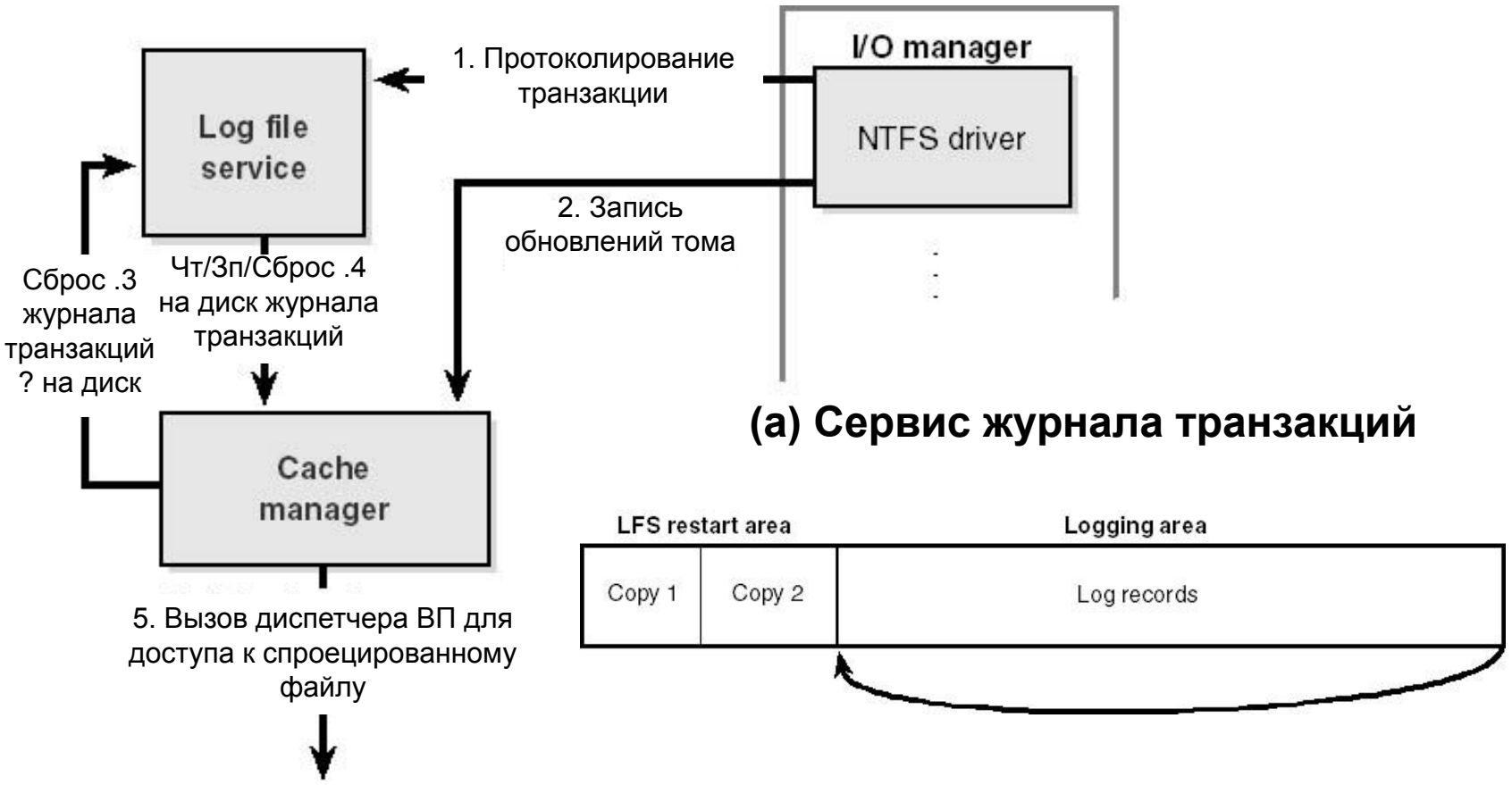


- Восстанавливаемость ФС в NTFS обеспечивается при помощи техники обработки транзакций, называемой протоколированием (logging).
- В процессе протоколирования, прежде чем выполнить над содержимым диска какую-либо подоперацию транзакции, изменяющей важные структуры файловой системы, NTFS записывает ее в файл журнала транзакций.

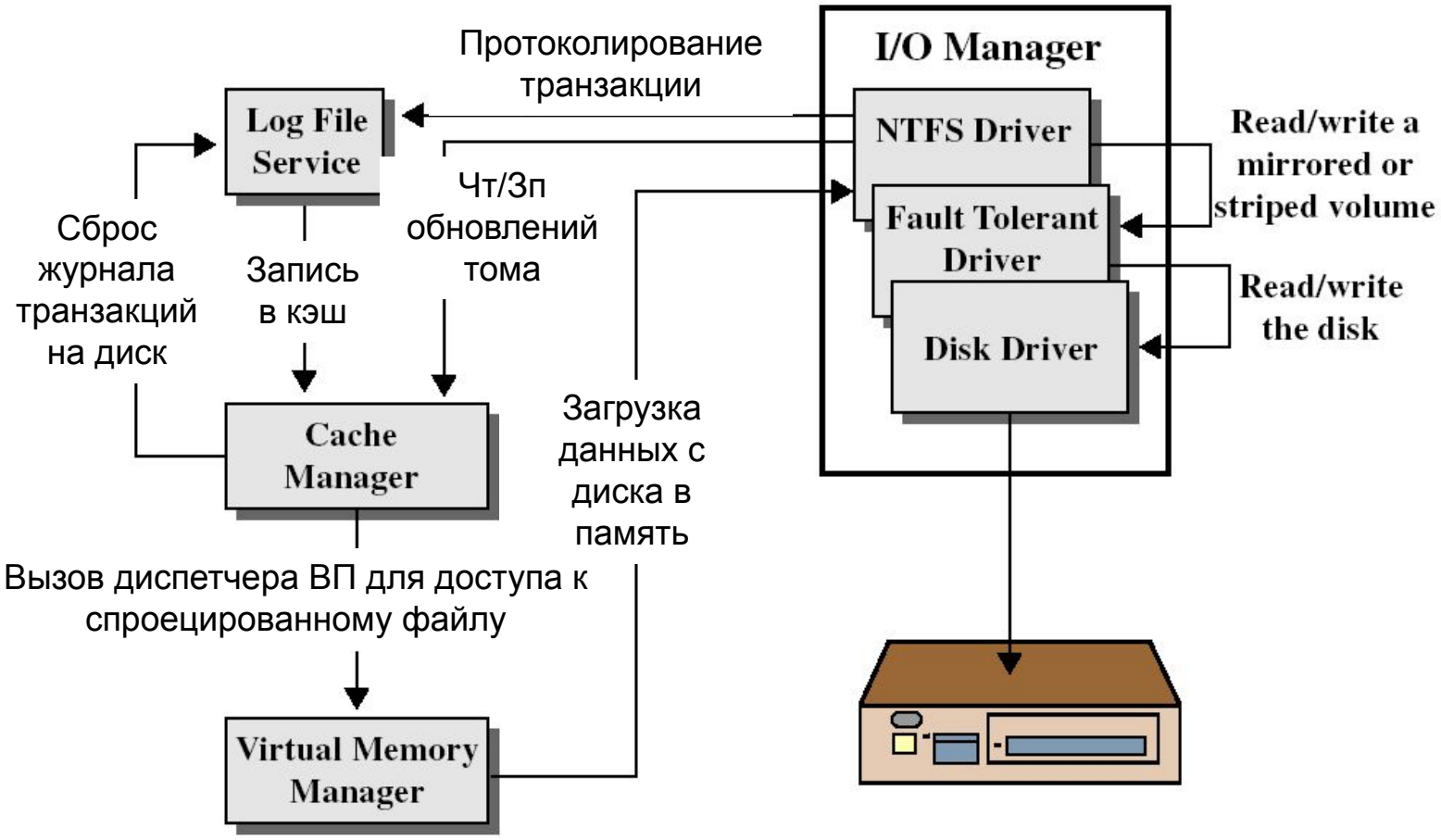
- В состав средств протоколирования NTFS входят следующие компоненты:
  - журнал транзакций (log file) – это системный файл, создаваемый командой Format.
  - сервис журнала операций (log file service, LFS) – набор системных процедур, которые NTFS использует для доступа к журналу транзакций. (log file).
  - диспетчер кэша (cache manager) – это системный компонент Windows, поддерживающий кэширование для NTFS и драйверов других ФС.

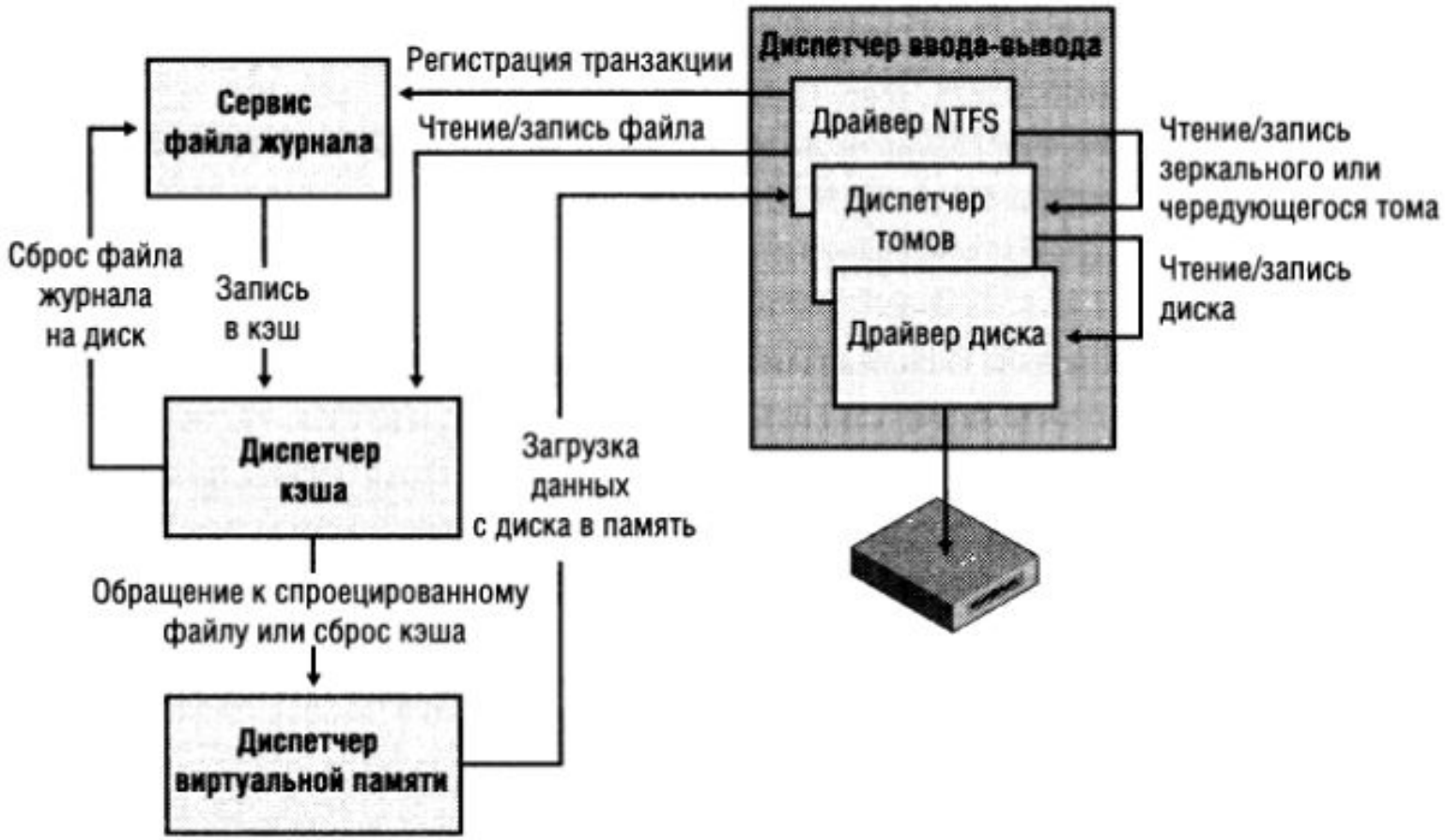
- Все ФС Windows NT осуществляют доступ к кэшированным файлам, отображая последние в виртуальную память выполняя чтение и запись в нее.
- В этих целях диспетчер кэша обеспечивает, ФС специализированный интерфейс к диспетчеру виртуальный памяти Windows NT. Если программа пытается обратиться к части файла, которая не загружен кэш, — так называемый промах кэша (cache miss), — диспетчер виртуальной памяти вызывает NTFS для обращения к драйверу диска и получения содержимого файла с диска.
- Диспетчер кэша оптимизирует дисковый ввод-вывод при помощи средства отложенной записи (lazy writer) — набора системных потоков управления, вызывающих диспетчер виртуальной памяти для сброса содержимого кэша на диск в фоновом режиме (асинхронная запись на диск).

- NTFS никогда не выполняет чтение-запись транзакций в журнал напрямую. LSN обеспечивает сервисы, которые NTFS вызывают для открытия файла журнала, помещения в журнал записей, считывания записей журнала в прямом и в обратном порядке, сброса записей журнала и т.д.
- Система обеспечивает восстановление тома следующим образом:
  - 1. Сначала NTFS вызывает LFS для записи в (кэшированный) файл журнала любых транзакций, модифицирующих структуру тома.
  - 2. NTFS модифицирует том (также в кэше).
  - 3. Диспетчер кэша вызывает LFS для уведомления о необходимости сбросить журнал транзакций на диск (этот сброс реализуется LFS при помощи обратного вызова диспетчера кэша с указанием страниц памяти, подлежащих выводу на диск).
  - 4-5. Сбросив на диск журнал транзакций, диспетчер кэша записывает на диск изменения тома (сами транзакции).

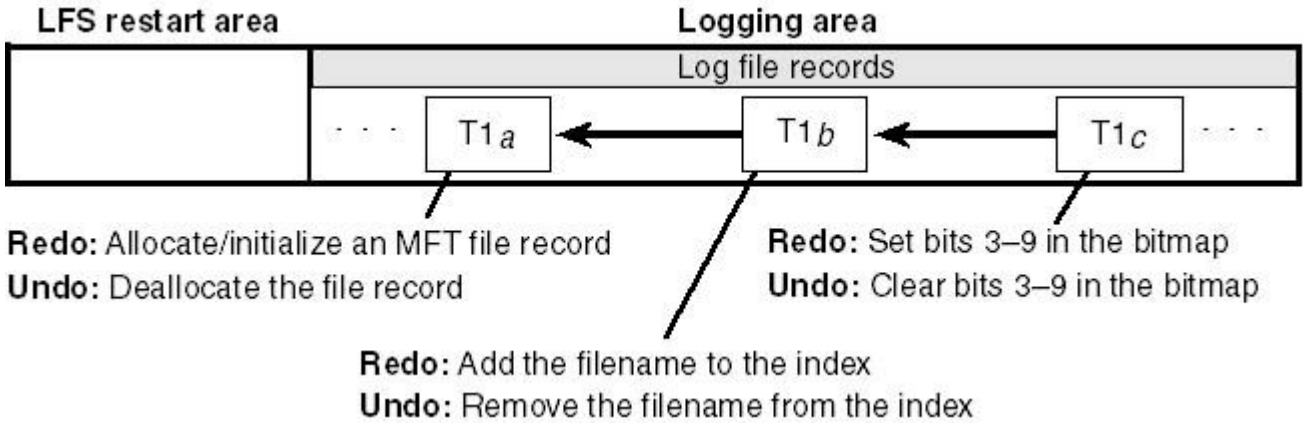

I/O manager
NTFS driver
Log file service
1. Протоколирование транзакции
2. Запись обновлений тома
Сброс .3 журнала транзакций ? на диск
Чт/Зп/Сброс .4 на диск журнала транзакций
Cache manager
5. Вызов диспетчера ВП для доступа к спроецированному файлу
(а) Сервис журнала транзакций
LFS restart area
Logging area
Copy 1
Copy 2
Log records
(б) Журнал транзакций

- Вызывающая программа – NTFS-драйвер, передает LFS указатель на открытый файловый объект, который будет использоваться как журнал транзакций. LFS либо инициализирует новый журнал, либо вызывает диспетчер кэша Windows NT для доступа к существующему журналу через кэш (а).
- LFS делит файл журнала на две части: область рестарта (restart area) и «бесконечную» область протоколирования (logging area) (б).
- NTFS вызывает LFS для чтения и записи области рестарта. В этой области NTFS хранит информацию контекста, такую как позиция в области протоколирования, откуда она будет начинать чтение при восстановлении после сбоя системы. На тот случай, что область рестарта будет разрушена или станет по каким-либо причинам недоступной, LFS создает ее копию. Остальная часть журнала транзакций — это область протоколирования, в которой находятся записи транзакций, обеспечивающие NTFS восстановление после сбоя. LFS создает иллюзию бесконечности журнала транзакций путем ее циклического повторного использования (в то же время гарантируя, что нужная информация не будет затерта). Для идентификации записей, помещенный журнал, LFS использует номера логической последовательности (logical sequence number, LSN).


I/O Manager
Протоколирование транзакции
Log File Service
NTFS Driver
Read/write a mirrored or striped volume
Fault Tolerant Driver
Read/write the disk
Disk Driver
Чт/Зп обновлений тома
Сброс журнала транзакций на диск
Запись в кэш
Cache Manager
Загрузка данных с диска в память
Вызов диспетчера ВП для доступа к спроецированному файлу
Virtual Memory Manager


Диспетчер ввода-вывода
Регистрация транзакции
Сервис файла журнала
Чтение/запись файла
Драйвер NTFS
Чтение/запись зеркального или чередующегося тома
Диспетчер томов
Чтение/запись диска
Драйвер диска
Сброс файла журнала на диск
Запись в кэш
Диспетчер кэша
Загрузка данных с диска в память
Обращение к спроецированному файлу или сброс кэша
Диспетчер виртуальной памяти

- создание файла
- удаление файла
- расширение файла
- урезание файла
- установка файловой информации
- переименование файла
- изменение прав доступа к файлу



- **Информация для повтора (redo info)**

  как вновь применить к то́му одну подоперацию полностью запротоколированной транзакции, если сбой системы произошел до того, как транзакция была переписана из кэша на диск.

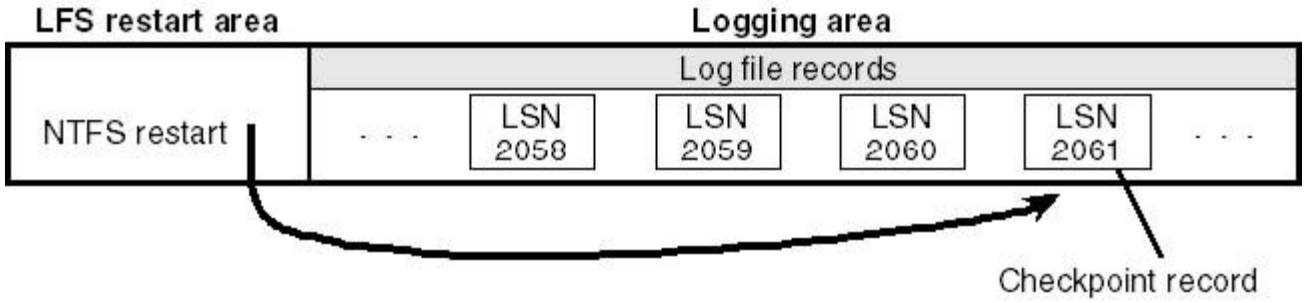
- **Информация для отмены (undo info)**

  как устранить изменения, вызванные одной подоперацией транзакции, которая в момент сбоя была запротоколирована лишь частично.

- После протоколирования транзакции NTFS выполняет ее подоперации непосредственно над томом – в кэше.
- По окончании обновления кэша NTFS помещает в журнал еще одну запись – подтверждение транзакции (committing a transaction). После того как транзакция подтверждена, NTFS гарантирует, что все вызванные ею модификации будут отражены на томе, даже если после подтверждения произойдет сбой ОС.

Периодически (5 сек.) NTFS помещает в журнал транзакций **записи контрольной точки**:

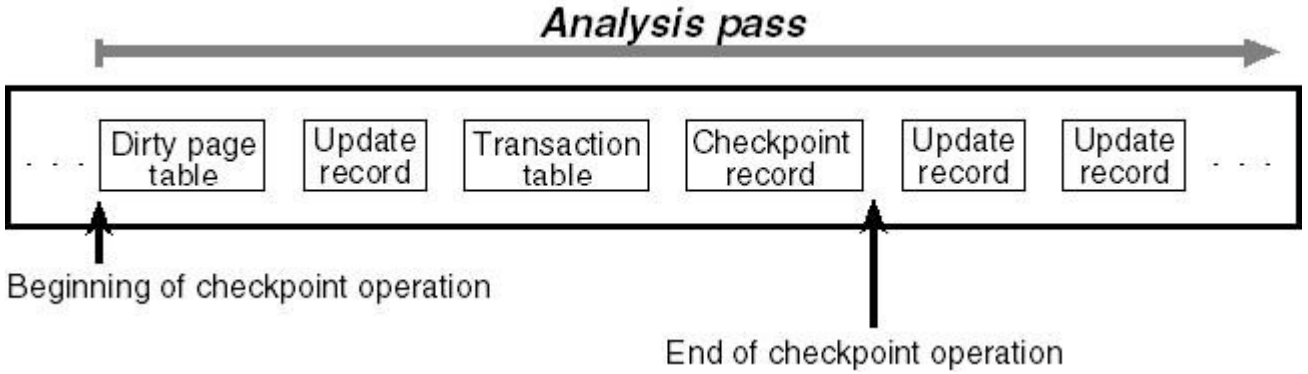
- Запись контрольной точки помогает NTFS определить, какая обработка необходима для восстановления тома, если сбой произошел “сразу” после помещения этой записи в журнал.
- LSN контрольной точки записывается в область рестарта.



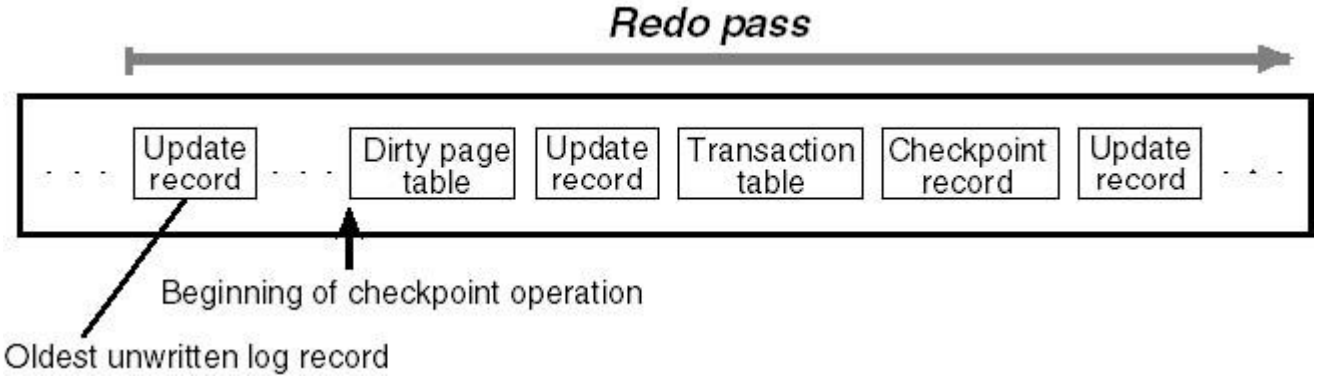
- **Таблица транзакций** (transaction table) предназначена для отслеживания транзакций, которые были начаты, но еще не подтверждены. Их надо удалить в процессе восстановления.
- В **таблицу измененных страниц** (dirty page table) записывается информация о том, какие страницы кэша содержат изменения структуры файловой системы, еще не записанные на диск. Эти данные в процессе восстановления должны быть сброшены на диск.

При восстановлении тома NTFS загружает журнал транзакций в оперативную память и выполняет три прохода:

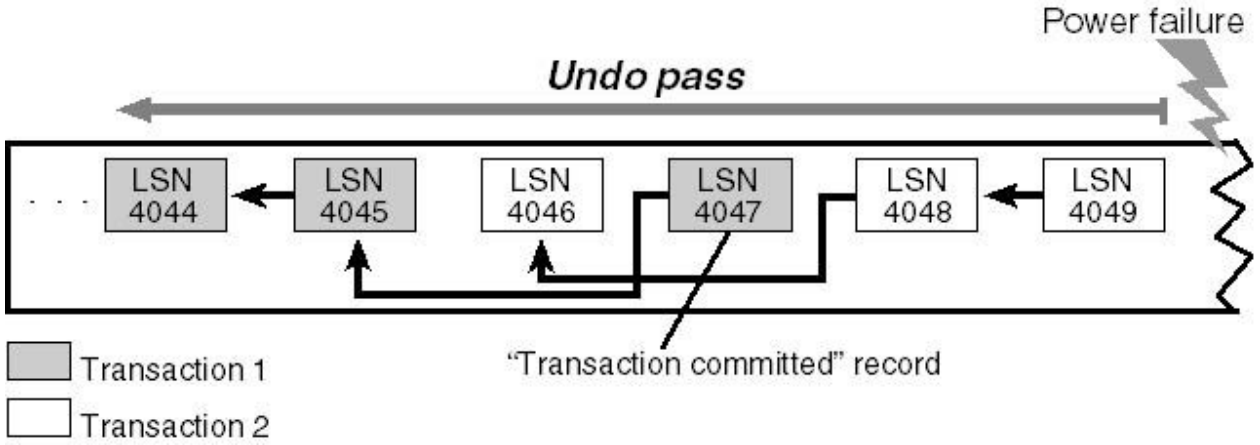
- **анализ;**
- **повтор транзакций;**
- **отмена транзакций.**



- просмотр журнала транзакций в прямом направлении, начиная с последней операции контрольной точки;
- поиск записей модификации и актуализация таблиц восстановления для последней контрольной точки;
- определение самой старой записи модификации, которая регистрирует невыполнение диском операции. LSN этой записи является началом для фазы повтора транзакции.



- сканирование журнала транзакций в прямом направлении, начиная с LSN самой старой записи, которая была обнаружена на проходе анализа;
- поиск записей "обновление страницы", содержащие модификации тома, которые были запротоколированы до сбоя системы, но не сброшены из кэша на диск;
- повторение найденных обновлений в кэш.



- откат всех транзакций, не подтвержденных к моменту сбоя системы, с протоколированием в журнале транзакций;
- сброс на диск изменений кэша;
- запись пустой области рестарта.

- Защита файлов NTFS на объектном уровне – Security Reference Monitor определяет, имеет ли пользователь необходимые права для вызова какого-либо из этих методов.
- Шифрование файлов с помощью специального драйвера EFS (Encrypting File System).

- **Стандартные разрешения:**
  - Full Control (Полный доступ)
  - Modify (Изменить)
  - Read & Execute (Чтение и выполнение)
  - Read (Чтение)
  - Write (Запись)
- **Специальные разрешения:**
  - Traverse Folder/Execute File (Обзор папок/ Выполнение файлов)
  - List Folder/Read Data (Содержание папок/ Чтение данных)
  - Read Attributes (Чтение атрибутов)
  - Read Extended Attributes (Чтение дополнительных атрибутов)
  - Create Files/Write Data (Создание файлов/Запись данных)
  - Create Folders/Append Data (Создание папок/ Дозапись данных)
  - Write Attributes (Запись атрибутов)
  - Write Extended Attributes (Запись дополнительных атрибутов)
  - Delete (Удаление)
  - Read Permissions (Чтение разрешений)
  - Change Permissions (Смена разрешений)
  - Take Ownership (Смена владельца)

Traverse Folder / Execute File	Определяет возможность перемещения по каталогам файловой системы вне зависимости от того, имеет или не имеет пользователь разрешения для просмотра пересекаемых в процессе перемещения каталогов. На работу этого разрешения влияет политика безопасности Bypass Traverse Checking (Обход перекрестной проверки) (см. узел Local Policies | User Rights Assignment в параметрах безопасности). Разрешение Execute File (Выполнение файлов) определяет возможность исполнения программ
List Folder / Read Data	Определяет возможность просмотра имен файлов или подкаталогов данного каталога (относится только к каталогу). Разрешение Read Data (Чтение данных) определяет возможность просмотра данных файла
Read Attributes	Определяет возможность просмотра атрибутов файла или каталога. Сами атрибуты определяются операционной системой
Read Extended Attributes	Определяет возможность просмотра дополнительных атрибутов файла или каталога. Сами дополнительные атрибуты определяются операционной системой
Create Files /Write Data	Определяет возможность создания файлов внутри каталога (относится только к каталогам). Разрешение Write Data (Запись данных) определяет возможность изменения содержимого файлов или перезаписи существующих данных файла новой информацией (относится только к файлам)
Create Folders / Append Data	Определяет возможность создавать подкаталоги внутри данного каталога (относится только к каталогам). Разрешение Append Data (Дозапись данных) определяет возможность присоединения новых данных к существующему файлу без изменения, уничтожения или перезаписи существующей информации (относится только к файлам)

Write Attributes	Определяет возможность изменения атрибутов файла или каталога. Атрибуты определяются операционной системой
Write Extended Attributes	Определяет возможность изменения дополнительных атрибутов файла или каталога. Дополнительные атрибуты определяются программой и могут быть ею изменены
Delete Subfolders and Files	Определяет возможность удаления подкаталогов и файлов, находящихся в данном каталоге, даже если для этих подкаталогов и файлов нет разрешения Delete (Удаление). Это разрешение имеется только у каталогов
Delete	Определяет возможность удаления файла или каталога. Если вам отказано в разрешении Delete (Удаление) для данного каталога или файла, вы все же можете удалить их, получив разрешение Delete Subfolders and Files (Удаление подпапок и файлов) на родительский каталог
Read Permissions	Определяет возможность чтения таких разрешений для файлов и каталогов, как Full Access, Read и т. д.
Change Permissions	Определяет возможность изменения таких разрешений для файлов и каталогов, как Full Access, Read и т. д.
Take Ownership	Определяет возможность вступления во владение данным файлом или каталогом. Владелец файла или каталога может всегда изменить разрешения к этому объекту, независимо от других разрешений

- This folder only (Только для этой папки);
- This folder, subfolders and files (Для этой папки, ее подпапок и файлов);
- This folder and subfolders (Для этой папки и ее подпапок);
- This folder and files (Для этой папки и ее файлов);
- Subfolders and files only (Только для подпапок и файлов);
- Subfolders only (Только для подпапок);
- Files only (Только для файлов).

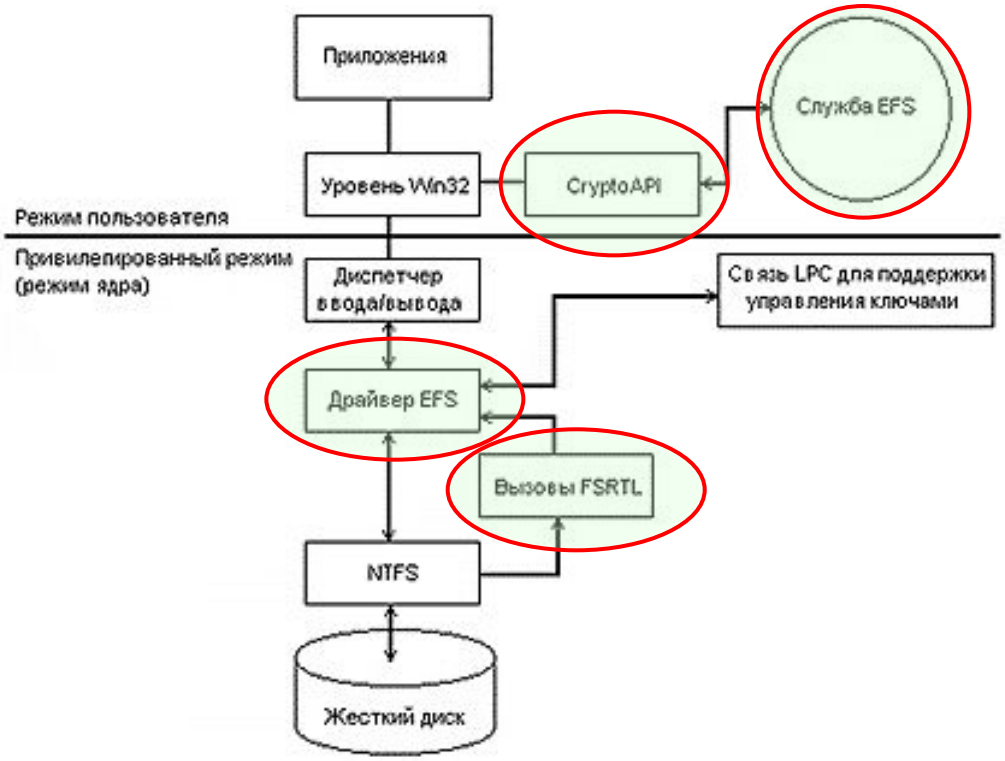
- Шифрование файлов - полезная возможность для людей, которые беспокоятся за свои секреты – каждый файл или каталог может также быть зашифрован, что не даст возможность прочесть его другой инсталляцией NT.
- Шифрующая файловая система (Encrypting File System – EFS) применяется для защиты файлов, хранящихся в томах NTFS. Ее необходимость вызвана следующими соображениями. Если доступ к разделу NTFS осуществляется не с помощью средств ОС Windows 2000-2003, а напрямую, на физическом уровне, то средства разграничения доступа и защиты данных от несанкционированного доступа, обеспечиваемые ОС не действуют и данные пользователя могут оказаться беззащитными. Такой доступ можно легко организовать, загрузившись с дискеты и используя специальные утилиты. Если же злоумышленник может извлечь жесткий диск и подключить его к другому компьютеру, то его задача еще более упрощается – он может свободно овладеть конфиденциальной информацией, которая хранится на жестком диске.
- Единственный способ защиты от физического чтения данных это шифрование файлов. Система EFS была разработана, чтобы обеспечить надежный и простой доступ пользователя к зашифрованным файлам, исключив при этом возможность несанкционированного доступа к ним (даже на физическом уровне) посторонних лиц. Шифрующая файловая система может быть особенно выгодна мобильным пользователям, которые сталкиваются с повышенным риском кражи и потери компьютера.

Применяется для защиты файлов, хранящихся в томах NTFS:

- надежный и простой доступ пользователя к зашифрованным файлам
- на физическом уровне исключение возможности несанкционированного доступа к файлам от посторонних лиц

EFS располагается в ядре Windows 2000-2003

- Драйвер EFS
- Библиотека реального времени EFS
- Служба EFS
- Набор CryptoAPI для Win32



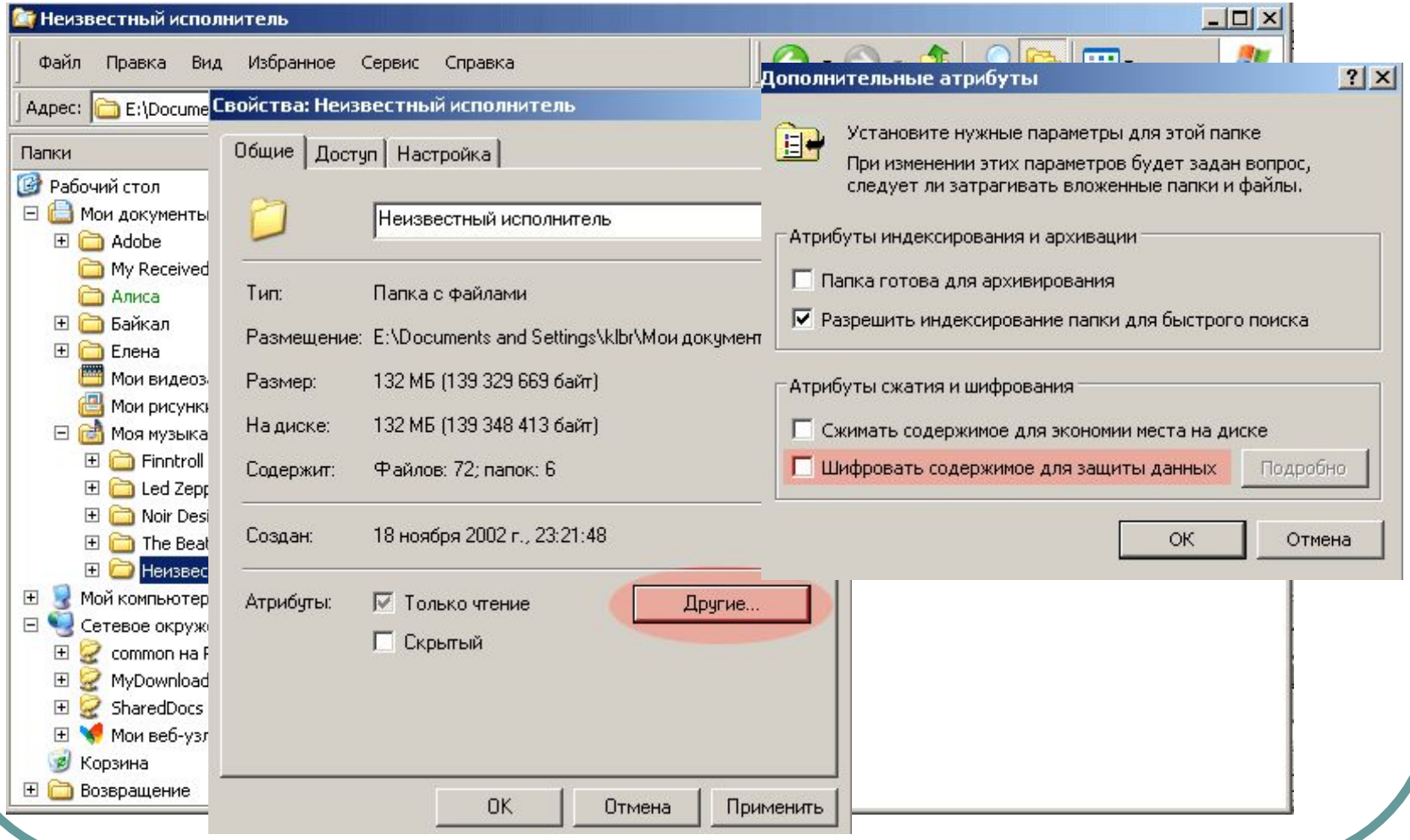
- *Драйвер EFS.* Драйвер EFS является надстройкой над файловой системой NTFS. Он обменивается данными со службой EFS — запрашивает ключи шифрования, наборы DDF (Data Decryption Field) и DRF (Data Recovery Field), — а также с другими службами управления ключами. Полученную информацию драйвер EFS передает библиотеке реального времени файловой системы EFS (File System Run Time Library, FSRTL), которая прозрачно для операционной системы выполняет различные операции, характерные для файловой системы (чтение, запись, открытие файла, присоединение информации).
- *Библиотека реального времени файловой системы EFS.* FSRTL — это модуль, находящийся внутри драйвера EFS, реализующий вызовы NTFS, выполняющие такие операции, как чтение, запись и открытие зашифрованных файлов и каталогов, а также операции, связанные с шифрованием, дешифрованием и восстановлением файлов при их чтении или записи на диск. Хотя драйверы EFS и FSRTL реализованы в виде одного компонента, они никогда не обмениваются данными напрямую. Для передачи сообщений друг другу они используют механизм вызовов (callouts) NTFS, предназначенный для управления файлами. Это гарантирует, что вся работа с файлами происходит при непосредственном участии NTFS. С помощью механизма управления файлами операции записи значений атрибутов EFS (DDF и DRF) реализованы как обычная модификация атрибутов файла. Кроме того, передача ключа шифрования файла FEK (см. ниже), полученного службой EFS, в FSRTL выполняется так, чтобы он мог быть установлен в контексте открытого файла. Затем контекст файла используется для автоматического выполнения операций шифрования и дешифрования при записи и чтении информации файла.
- *Служба EFS.* Служба EFS (EFS Service) является частью системы безопасности операционной системы. Для обмена данными с драйвером EFS она использует порт связи LPC, существующий между локальным администратором безопасности (Local Security Authority, LSA) и монитором безопасности, работающим в привилегированном режиме. В режиме пользователя для создания ключей шифрования файлов и генерирования данных для DDF и DRF служба EFS использует CryptoAPI. Она также поддерживает набор API для Win32.
- *Набор API для Win32.* Этот набор интерфейсов прикладного программирования позволяет выполнять шифрование файлов, дешифрование и восстановление зашифрованных файлов, а также их импорт и экспорт (без предварительного дешифрования). Эти API поддерживаются стандартным системным модулем DLL — advapi32.dll.

- EFS использует архитектуру Windows CryptoAPI, использующую технологию шифрования с открытым ключом.
- Для шифрования каждого файла случайным образом генерируется ключ шифрования файла.
- Для шифрования файла по умолчанию используется симметричный алгоритм DESX с длиной ключа *56* бит. Windows XP и Windows Server 2003 позволяют переключить алгоритм на TripleDES.
- Ключи шифрования EFS хранятся в резидентном пуле памяти, что исключает несанкционированный доступ к ним через файл подкачки.

- Шифрование с симметричным ключом представляет данные в недоступном для третьих лиц виде, используя единый секретный ключ для шифрования и дешифрования.
  - Область назначения – групповое шифрование больших объемов данных
  - Достоинства – скорость и безопасность
  - Недостатки – обмен ключами

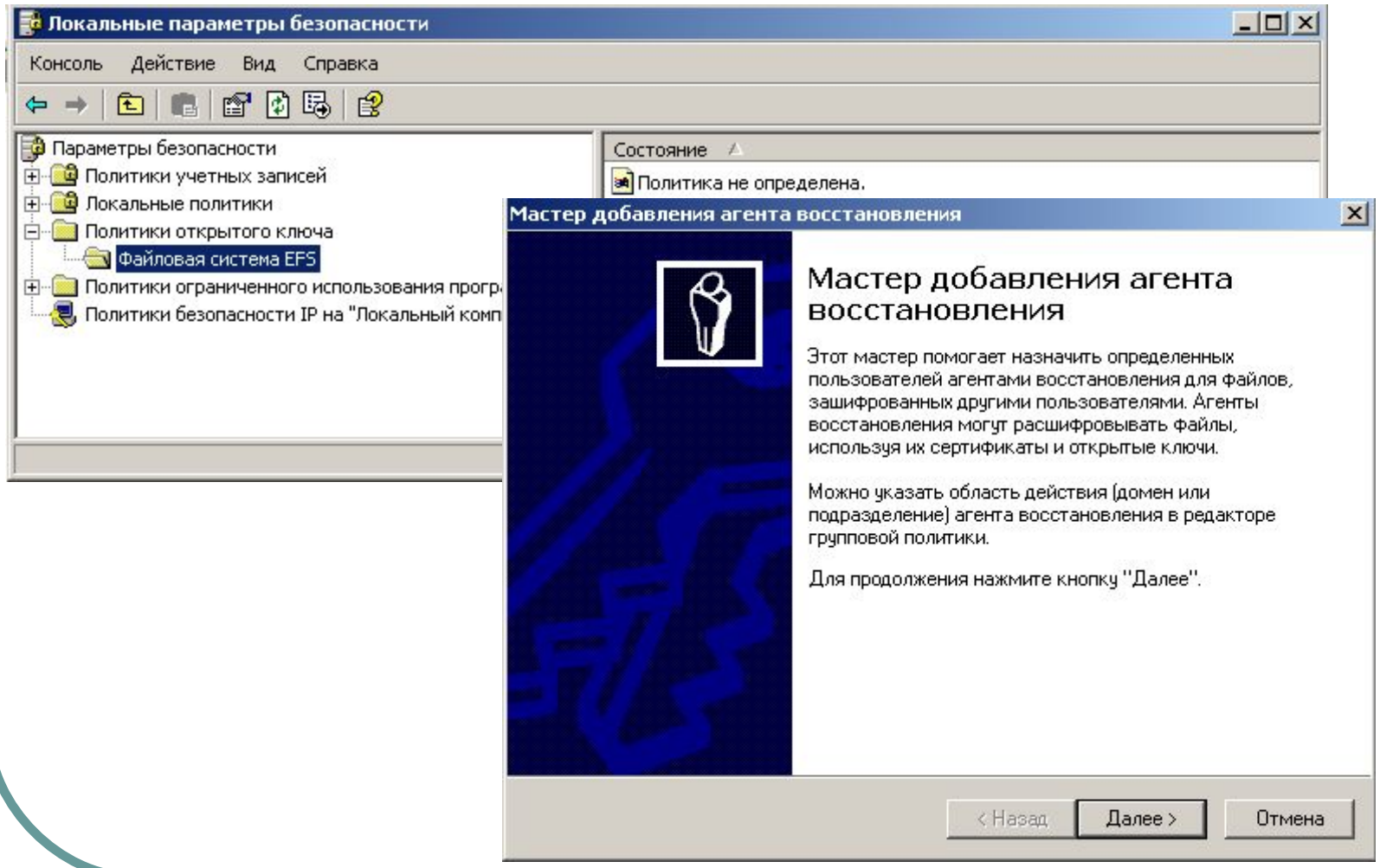
- Среди симметричных алгоритмов можно выделить следующие: DES (Data Encryption Standard), DES 2, различные вариации TripleDES.
- Алгоритм TripleDES имеет различные конфигурации, немного отличающиеся по принципу работы:
  - DES-EEE3 – тройное шифрование с различными ключами.
  - DES-EDE3 – шифрует, дешифрует и ещё раз шифрует с различными ключами.
  - DES-EEE2 – тройное шифрование, но одинаковые ключи только при первой и третьей итерациях.
  - DES-EDE2 – шифрование, дешифрование и ещё раз шифрование с одинаковыми ключами при первой и третьей итерациях.

- шифрование файлов и папок
- шифрование удаленных файлов
- восстановление данных в EFS
- работа с зашифрованными файлами
- резервное копирование зашифрованных файлов
- усовершенствованная система EFS в Windows Server 2003


Неизвестный исполнитель
Файл Правка Вид Избранное Сервис Справка
Адрес: E:\Docume
Папки
Рабочий стол
Мои документы
Adobe
My Received
Алиса
Байкал
Елена
Мои видеоз
Мои рисунки
Моя музыка
Finntroll
Led Zepp
Noir Desi
The Beat
Неизвес
Мой компьютер
Сетевое окруж
common на F
MyDownload
SharedDocs
Мои веб-узл
Корзина
Возвращение
Свойства: Неизвестный исполнитель
Общие
Доступ
Настройка
Неизвестный исполнитель
Тип: Папка с файлами
Размещение: E:\Documents and Settings\klbr\Мои документ
Размер: 132 МБ (139 329 669 байт)
На диске: 132 МБ (139 348 413 байт)
Содержит: Файлов: 72; папок: 6
Создан: 18 ноября 2002 г., 23:21:48
Атрибуты: Только чтение
Скрытый
Другие...
OK
Отмена
Применить
Дополнительные атрибуты
Установите нужные параметры для этой папке
При изменении этих параметров будет задан вопрос, следует ли затрагивать вложенные папки и файлы.
Атрибуты индексирования и архивации
Папка готова для архивирования
Разрешить индексирование папки для быстрого поиска
Атрибуты сжатия и шифрования
Сжимать содержимое для экономии места на диске
Шифровать содержимое для защиты данных
Подробно
OK
Отмена

cipher [/E | D] [t/S:каталог] [/A] [/I] [/F] [/Q] [/H] [/К] [путь [...]]

/E	Шифрует указанные в качестве параметра *путь* файлы.
/D	Дешифрует все указанные после ключа файлы.
/S	Выполняет заданную операцию с каталогом *каталог* и всеми его подкаталогами, файлы при этом не обрабатываются.
/A	Выполняет определенную ключом операцию как для каталогов, так и для отдельных файлов.
/I	Продолжает выполнение указанной операции даже после возникновения ошибочной ситуации.
/F	Осуществляет принудительное шифрование всех файлов, указанных после ключа, даже если они уже зашифрованы.
/Q	Выдает только краткую информацию.
/H	Отображает файлы, для которых установлены атрибуты Hidden и System.
/К	Создает новый ключ шифрования файлов для пользователя, запустившего команду; при этом все другие ключи команды игнорируются.


Локальные параметры безопасности
Консоль
Действие
Вид
Справка
Параметры безопасности
Политики учетных записей
Локальные политики
Политики открытого ключа
Файловая система EFS
Политики ограниченного использования прогр
Политики безопасности IP на "Локальный комп
Состояние
Политика не определена.
Мастер добавления агента восстановления
Мастер добавления агента восстановления
Этот мастер помогает назначить определенных пользователей агентами восстановления для файлов, зашифрованных другими пользователями. Агенты восстановления могут расшифровывать файлы, используя их сертификаты и открытые ключи.
Можно указать область действия (домен или подразделение) агента восстановления в редакторе групповой политики.
Для продолжения нажмите кнопку "Далее".
< Назад
Далее >
Отмена

- Hard Link – несколько имен для одного файла
- Точки соединения NTFS (junction point)

fsutil hardlink create <новый файл>
<существующий файл>

Пример: fsutil hardlink create
c:\foo.txt c:\bar.txt

- Другим новшеством в Windows 2000 стало монтирование устройств. Утилита Disk Administrator Windows NT позволяла назначить тому букву латинского алфавита. Этот довольно простой метод дает возможность обратиться к любому дисковому устройству из стандартного меню открытия файла. Естественным ограничением на количество локальных и подключенных сетевых устройств было число 26, соответствующее числу букв латинского алфавита.
- Подмонтирование возможно только к пустым папкам на NTFS-томах, а точки монтирования вы можете создать или из оснастки «Управление дисками», или из командной строки при помощи команды **mountvol**. Для того, чтобы отличить подмонтированные накопители от обычных папок, Explorer показывает их иконками соответствующих устройств. Для чего это может понадобиться? Во-первых, можно таким образом преодолеть ограничение на количество доступных логических дисков (ранее их не могло быть больше 26 - по числу букв латинского алфавита), повысить ёмкость существующих томов не используя динамические и… создавать отказоустойчивые папки на обычных томах.
- Например, при монтировании нового основного раздела к папке D:\My Work Stuff все последующие обращения к этой папке будут автоматически переадресованы на соответствующий новый основной раздел, даже если он расположен на другом физическом диске, чем устройство D:. Если новый том является отказоустойчивым, то и папка D:\My Work Stuff считается отказоустойчивой, даже если само устройство D: этим качеством не обладает.

- С помощью утилиты mountvol.exe можно:
  - Отобразить корневую папку локального тома в некоторую папку NTFS 5.0 (другими словами, подключить том). Вывести на экран информацию о целевой папке точки соединения NTFS, использованной при подключении тома. Просмотреть список доступных для использования томов файловой системы. Уничтожить точки подключения томов, созданных с помощью mountvol.
  - Применяя утилиту mountvol, можно избежать использования большого количества имен устройств, поскольку обращение к необходимому тому происходит через корневую папку. Утилита mountvol создает неизменные связи с корневыми папками локального тома файловой системы. Она применяет в работе новую технологию, гарантирующую, что при изменениях в параметрах оборудования целевая папка не изменяется.
- Синтаксис вызова утилиты mountvol;
  - mountvol *[устройство:]путь Имя_тома*
  - где:
  - *[устройство:]путь* — определяет существующую папку NTFS 5.0, являющуюся точкой подключения тома; *имя_тома* — определяет имя подключаемого тома.
  - Параметры утилиты mountvol:
  - /o — уничтожение существующей точки подключения у указанной папки.
  - /l — отображение списка томов, подключенных к данной папке.

- NTFS полностью не предотвращает фрагментацию
- NTFS снижает возможность возникновения фрагментации (например, в многозадачном режиме)
- NTFS снижает отрицательное влияние фрагментации на быстродействие

defrag <том> [-a] [-f] [-v] [-?]

том Буква диска, или точка подключения (например, d: или d:\vol\mpoint)

-a Только анализ

-f Дефрагментация даже при ограниченном месте на диске

-v Подробные результаты

-? Вывод справки

- Версия NTFS, поставляемая с Windows NT, ограничивает число разделов 26-ю (диски от A до Z). Кроме того, изменение раздела всегда требует перезагрузки. К тому же, информация о томах NTFS хранится в реестре, что усложняет использование диска с другой системой.
- Проблема была решена в Windows 2000 с помощью Logical Disk Manager (LDM), который больше не требует присвоения букв дискам. Эта система NTFS способна также сохранять информацию о системе на жёстком диске, что решает проблему замены дисков.
- Улучшения NTFS в Windows XP незначительны по сравнению с Windows 2000. Была улучшена пропускная способность данных, и вместо фиксированного размера кластера по 512 байт можно устанавливать другие значения. Также были улучшены и административные функции, к примеру, индексация папок и ограничение непредвиденных расходов памяти.

fsutil fsinfo

---- Поддерживаемые команды FSINFO----

drives Отображение всех драйверов

drivetype Отображение типа привода для устройств

volumeinfo Отображение информации о томе

ntfsinfo Отображение информации о NTFS

statistics Отображение статистики файловой системы

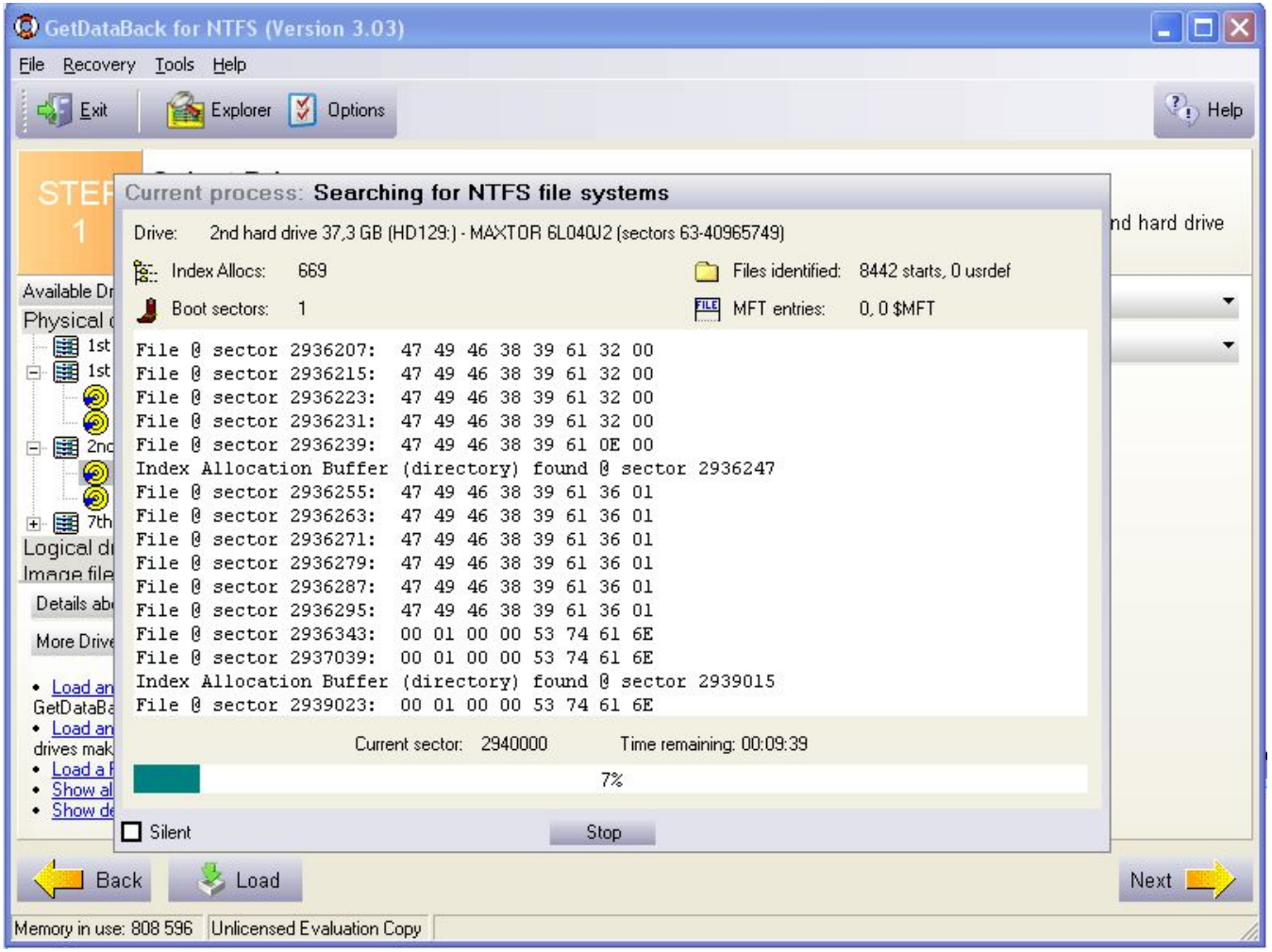
```
C:\Documents and Settings\avtimofeev>fsutil fsinfo ntfsinfo C:
Серийный номер тома NTFS:        0x405c3fc95c3fb90c
Версия:                          3.1
Число секторов:                  0x0000000002711636
Всего кластеров:                 0x00000000004e22c6
Свободных кластеров:             0x00000000001de2db
Всего зарезервировано:           0x0000000000000000
Байт на сектор:                  512
Байт на кластер:                 4096
Байт на сегмент FileRecord:      1024
Кластеров на сегмент FileRecord: 0
Допустимая длина данных MFT:     0x0000000006148400
Начальный LCN таблицы MFT:       0x00000000000c0000
Начальный LCN таблицы MFT2:      0x0000000000271163
Начало зоны таблицы MFT:         0x00000000002265c0
Конец зоны таблицы MFT:          0x0000000000271180
```

- Отключите обновление сведений о последнем доступе к файлу

  fsutil behavior set disablelastaccess 1
- Зарезервируйте необходимое пространство для MFT

  fsutil behavior set mftzone <значение>
- Отключите создание коротких имен файлов 8.3

  fsutil behavior set disable8dot3 1


GetDataBack for NTFS (Version 3.03)
File Recovery Tools Help
Exit
Explorer
Options
Help
Current process: Searching for NTFS file systems
Drive: 2nd hard drive 37,3 GB (HD129:) - MAXTOR 6L040J2 (sectors 63-40965749)
Index Allocs: 669
Files identified: 8442 starts, 0 usrdef
Boot sectors: 1
MFT entries: 0, 0 $MFT
File @ sector 2936207: 47 49 46 38 39 61 32 00
File @ sector 2936215: 47 49 46 38 39 61 32 00
File @ sector 2936223: 47 49 46 38 39 61 32 00
File @ sector 2936231: 47 49 46 38 39 61 32 00
File @ sector 2936239: 47 49 46 38 39 61 0E 00
Index Allocation Buffer (directory) found @ sector 2936247
File @ sector 2936255: 47 49 46 38 39 61 36 01
File @ sector 2936263: 47 49 46 38 39 61 36 01
File @ sector 2936271: 47 49 46 38 39 61 36 01
File @ sector 2936279: 47 49 46 38 39 61 36 01
File @ sector 2936287: 47 49 46 38 39 61 36 01
File @ sector 2936295: 47 49 46 38 39 61 36 01
File @ sector 2936343: 00 01 00 00 53 74 61 6E
File @ sector 2937039: 00 01 00 00 53 74 61 6E
Index Allocation Buffer (directory) found @ sector 2939015
File @ sector 2939023: 00 01 00 00 53 74 61 6E
Current sector: 2940000
Time remaining: 00:09:39
7%
Silent
Stop
Back
Load
Next
Memory in use: 808 596
Unlicensed Evaluation Copy

- Проект **Linux-NTFS**. Включает модуль ядра, а также набор утилит для различных операций с файловыми системами NTFS (проверка целостности, восстановление удалённых файлов, изменение размера и др.). Именной этот драйвер включается в ядро Linux ещё с версии 2.2 (с тех пор он был практически полностью переписан). Модулем ядра поддерживается практически только чтение (запись — лишь в существующие файлы без изменения их размера), но недавно в рамках проекта утилита ntfsmount, использующая FUSE и позволяющая монтировать NTFS-разделы на запись. Это первый полностью свободный продукт, имеющий такую возможность.
- http://www.linux-ntfs.org

Базовые (обычные) и динамические диски в NTFS

- В Windows 2000 введены понятия базового и динамического дисков. Базовые диски построены с использованием схемы работы с разделами DOS. Другими словами, базовые диски являются унаследованными от Windows NT. На обычных дисках могут располагаться простые тома, например основные разделы диска, дополнительные разделы и логические устройства. При модернизации Windows 2000 с предыдущих версий Windows NT в число обычных дисков попадут все тома с защитой от сбоев, такие, как тома с дублированием и дисковые массивы с чередованием и четностью. На обычных дисках нельзя создать новые многодисковые тома.
- Динамические диски представляют собой новый тип дисков в Windows 2000, которые позволяют использовать многодисковые тома в Windows 2000.

- Различие между базовым и динамическим дисками проявляется при обращении с томами, состоящими из нескольких разделов.
  - Для базовых дисков вся информация о настройках сложных типов томов хранится в реестре (уязвимость и недостаточная надежность).
  - Для динамических дисков информация о настройках сохраняется на диске. Такой способ хранения привязывает динамический диск к устройству хранения, на котором он определен.
  - Используя динамическое хранение, вы можете управлять дисками и томами без перезагрузки операционной системы.

- Windows 2000 рассматривает все диски как базовые, до тех пор пока вручную не будет создан динамический диск или базовый диск (при наличии на нем достаточного объема свободного пространства) не будет конвертирован в динамический.

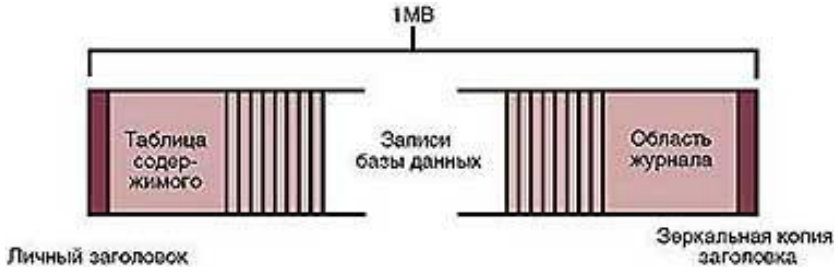
Базовый диск	Динамический диск
Системный (system partition) и загрузочный разделы (boot partition)	Системный (system volume) и загрузочный тома (boot volume)
Основной раздел (primary partition)	Простой том (simple volume)
Дополнительный раздел (extended partiotion)	Простые тома и свободное пространство диска
Логический диск (logical drive)	Простой том
Набор томов (volume set)	Составной том (spanned volume)
Чередующийся набор (stripe set without parity)	Чередующийся том (stripped volume)
Зеркальный набор (mirror set)	Зеркальный том (mirrored volume)
Чередующийся набор с четностью (stripe set with parity)	Том RAID-5 (RAID-5 volume)

- Подсистема логического менеджера дисков Logical Disk Manager (LDM), состоящая из компонентов пользовательского режима и драйверов устройств, ответственна за динамические диски. LDM был лицензирован Microsoft у компании VERITAS Software, которая первоначально разработала эту технологию для UNIX.
- Совместно с Microsoft компания VERITAS перенесла LDM в Windows 2000, чтобы реализовать в ней новые возможности управления разделами. Механизмы управления разделами в стиле DOS и в стиле LDM различаются прежде всего тем, что LDM управляет одной унифицированной базой, хранящей информацию обо всех динамических дисках системы, включая настройку расширенных разделов.

- База данных LDM размещается в зарезервированной области размером 1 Мбайт в конце каждого динамического диска. Соответственно, для выполнения конвертации в конце каждого базового диска должно быть свободное пространство.
- Кроме того менеджер дисков создает и обычную DOS-таблицу разделов, чтобы унаследованные утилиты управления дисками, в случае системы с многовариантной загрузкой ошибочно не посчитали динамический диск неразбитым на разделы.
- Таблица разделов нужна также для того, чтобы программа загрузки Windows 2000 могла находить системный и загрузочный тома, даже если они расположены на динамических дисках.

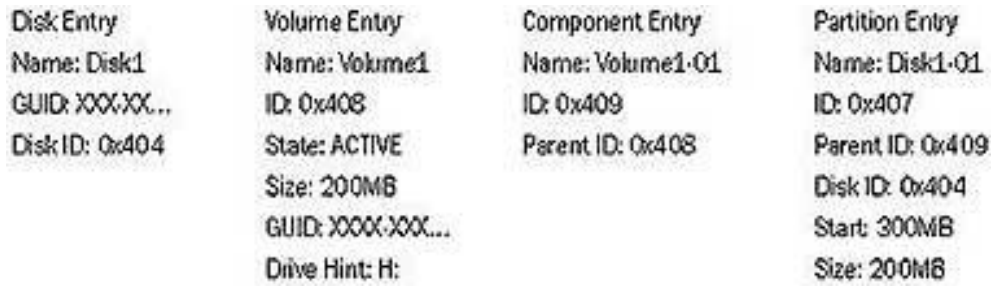
- Если диск содержит системный или загрузочный тома, разделы указывают на расположение этих томов. В противном случае один раздел начинается на первом цилиндре диска (при 63 секторах на диске) и продолжается до начала базы данных LDM. В этой области разделов и создаются разделы LDM, информация о которых хранится в базе данных диска.

- База данных LDM состоит из четырех областей:
  - сектора заголовка, называемого в LDM личным заголовком (Private Header);
  - таблицы содержимого;
  - записей базы данных;
  - журнала транзакций.
- В целях защиты от сбоев LDM сохраняет копию Private Header в последнем секторе диска.



- Таблица содержимого базы занимает 16 секторов и содержит информацию о расположении данных базы.
- Область записей начинается сразу после этой таблицы, с сектора, выделенного под заголовок базы.
- В данном секторе хранится информация об области записей: число записей, имя и GUID дисковой группы, к которой относится база, и порядковый номер, используемый LDM для каждой следующей записи базы данных.
- Вслед за заголовком базы размещены сектора, содержащие записи фиксированной длины размером 128 байт, в которых хранится информация о разделах дисковой группы и томах.

- Запись базы данных может быть одного из четырех типов: раздел, диск, компонент и том.
- LDM использует указанные типы записей и трехуровневую систему описания томов, сопоставляя записи базы данных с внутренними идентификаторами объектов.



- Для описания простого тома LDM нужно три записи: запись о разделе, запись типа компонент и запись о томе.
- На рисунке представлено содержимое простой базы LDM, которая определяет один 200-мегабайтный том на одном разделе. Запись о разделе описывает область на диске, выделенную системой для тома, запись типа компонент соединяет записи о разделе и томе, а запись о томе содержит GUID, используемый внутри Windows 2000 для идентификации тома.
- Более сложные тома требуют более трех записей. Например, чередующийся набор имеет как минимум две записи о разделах, запись типа компонент и запись о томе.

- Последняя область базы LDM – зона журнала транзакций, состоящая из нескольких секторов для хранения резервной копии информации о структуре базы в процессе ее изменения.
- Данная область может помочь при разрушении LDM или сбое в питании, поскольку с помощью этого журнала базу можно вернуть в рабочее состояние.

NTFS vs. FAT

- **NTFS** способна обеспечить быстрый поиск фрагментов, поскольку вся информация хранится в нескольких компактных записях. Если файл очень сильно фрагментирован – NTFS придется использовать много записей, которые могут храниться в разных местах.
- **FAT32**, из-за большой области самой таблицы размещения будет испытывать огромные трудности, если фрагменты файла разбросаны по всему диску. Для доступа к фрагменту файла в системе FAT16 и FAT32 приходится обращаться к соответствующей ячейке таблицы FAT.
- В системе **FAT16**, где максимальный размер области FAT составляет 128 Кбайт, это не составит проблемы – вся область FAT просто хранится в памяти, или же считывается с диска целиком за один проход и буферизируется. FAT32 же, напротив, имеет типичный размер области FAT порядка сотен килобайт, а на больших дисках – даже несколько мегабайт.

- Для определения того, свободен ли данный кластер или нет, системы на основе FAT должны просмотреть одну запись FAT, соответствующую этому кластеру. Для поиска свободного места на диске может потребоваться просмотреть почти всего FAT – это 128 Кбайт (максимум) для **FAT16** и до нескольких мегабайт (!) – в **FAT32**. Для того, чтобы не превращать поиск свободного места в катастрофу (для FAT32), ОС приходится идти на различные ухищрения.
- **NTFS** имеет битовую карту свободного места, одному кластеру соответствует 1 бит. Для поиска свободного места на диске приходится оценивать объемы в десятки раз меньшие, чем в системах FAT и FAT32.

- **FAT16** и **FAT32** имеют очень компактные каталоги, размер каждой записи которых предельно мал. Работа с каталогами FAT производится достаточно быстро, так как в подавляющем числе случаев каталог не фрагментирован и находится на диске в одном месте. Единственная проблема – высокая трудоемкость поиска файлов в больших каталогах. Система хранения данных – линейный массив – не позволяет организовать эффективный поиск файлов в таком каталоге.
- **NTFS** использует гораздо более эффективный способ адресации – бинарное дерево. Сам каталог NTFS представляет собой *гораздо* менее компактную структуру, чем каталог FAT – это связано с гораздо большим (в несколько раз) размером одной записи каталога. Это обстоятельство приводит к тому, что каталоги на томе NTFS могут быть сильно фрагментированы. Размер типичного каталога в FAT укладывается в один кластер, тогда как сотня файлов (и даже меньше) в каталоге на NTFS уже приводит к размеру файла каталога, превышающему типичный размер одного кластера. Это, в свою очередь, может привести к фрагментации файла каталога, и уменьшить положительный эффект от более эффективной организации самих данных.

Параметр	Лидер	Аутсайдер
Поиск данных файла	FAT16, NTFS	FAT32
Поиск свободного места	NTFS	FAT32
Работа с каталогами и файлами	NTFS(*)	

- Тома FAT32 могут теоретически быть больше 2 ТБ, но операционные системы Windows Server 2003, Windows 2000 и Windows XP могут форматировать диски объемом только до 32 ГБ. (Windows Server 2003, Windows 2000 и Windows XP Professional могут читать и записывать на большие по объему тома FAT32, отформатированные другими операционными системами).
- Тома NTFS могут теоретически быть объемом до 16 эксабайт (ЭБ), но практический предел составляет 2 ТБ.
- Пользователь может определить размер кластера при форматировании тома NTFS. Однако NTFS-сжатие не поддерживается для кластеров, размер которых больше 4 КБ.