Файловая система NTFS

Защита целостности данных

Защита целостности данных

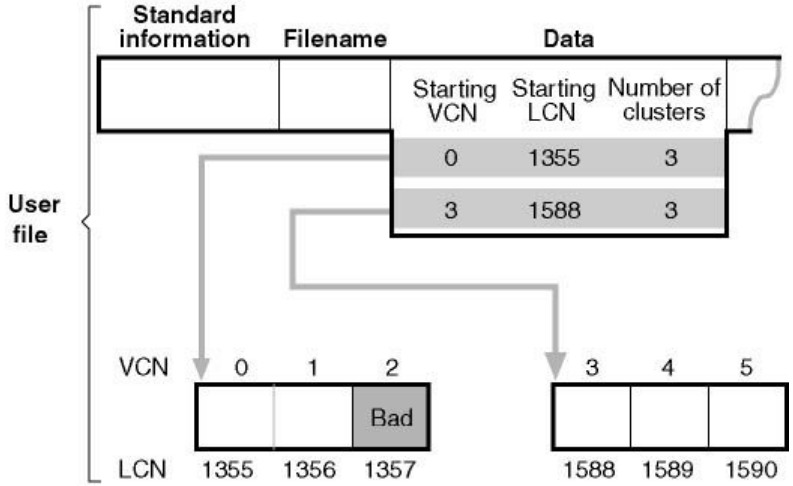
- NTFS является *восстанавливаемой* ФС и поддерживает следующие технологии защиты целостности данных:
  - *Горячая фиксация* – позволяет файловой системе при возникновении ошибки из-за плохого кластера записать информацию в другой кластер и отметить сбойный в качестве плохого.
  - *Механизм транзакций* – каждая операция ввода-вывода, которая изменяет файл на разделе NTFS, рассматривается файловой системой как транзакция и может выполняться только как

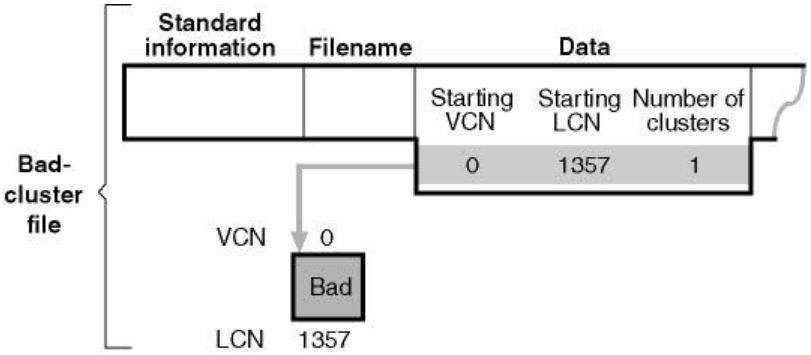
Система восстановления NTFS гарантирует корректность **файловой системы**, а не ваших данных.

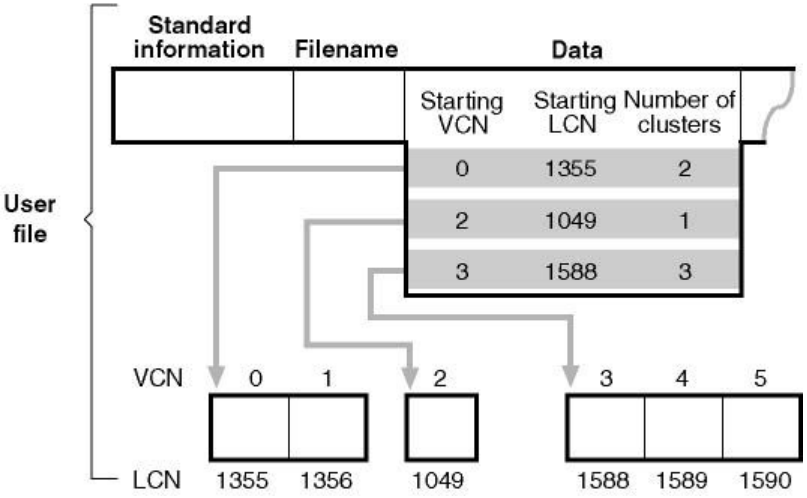
Вопрос

- Какие механизмы защиты целостности данных есть у файловых систем FAT ?

Горячая фиксация







а) исходная MFT-запись файла;

б) обнаружение сбойного кластера 1375;

в) исправленная MFT-запись файла.

Механизм транзакций

- Восстанавливаемость файловой системы 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.

Средства протоколирования транзакций

- В состав средств протоколирования NTFS входят следующие компоненты:
  - журнал транзакций (log file) – это мета-файл $LogFile, создаваемый командой *format*;
  - сервис журнала операций (log file service, LFS) – набор системных процедур, которые NTFS использует для доступа к журналу транзакций (log file), NTFS никогда не выполняет чтение-запись транзакций в журнал напрямую;
  - диспетчер кэша (cache manager) – это системный компонент Windows, поддерживающий кэширование для NTFS и драйверов других файловых систем.

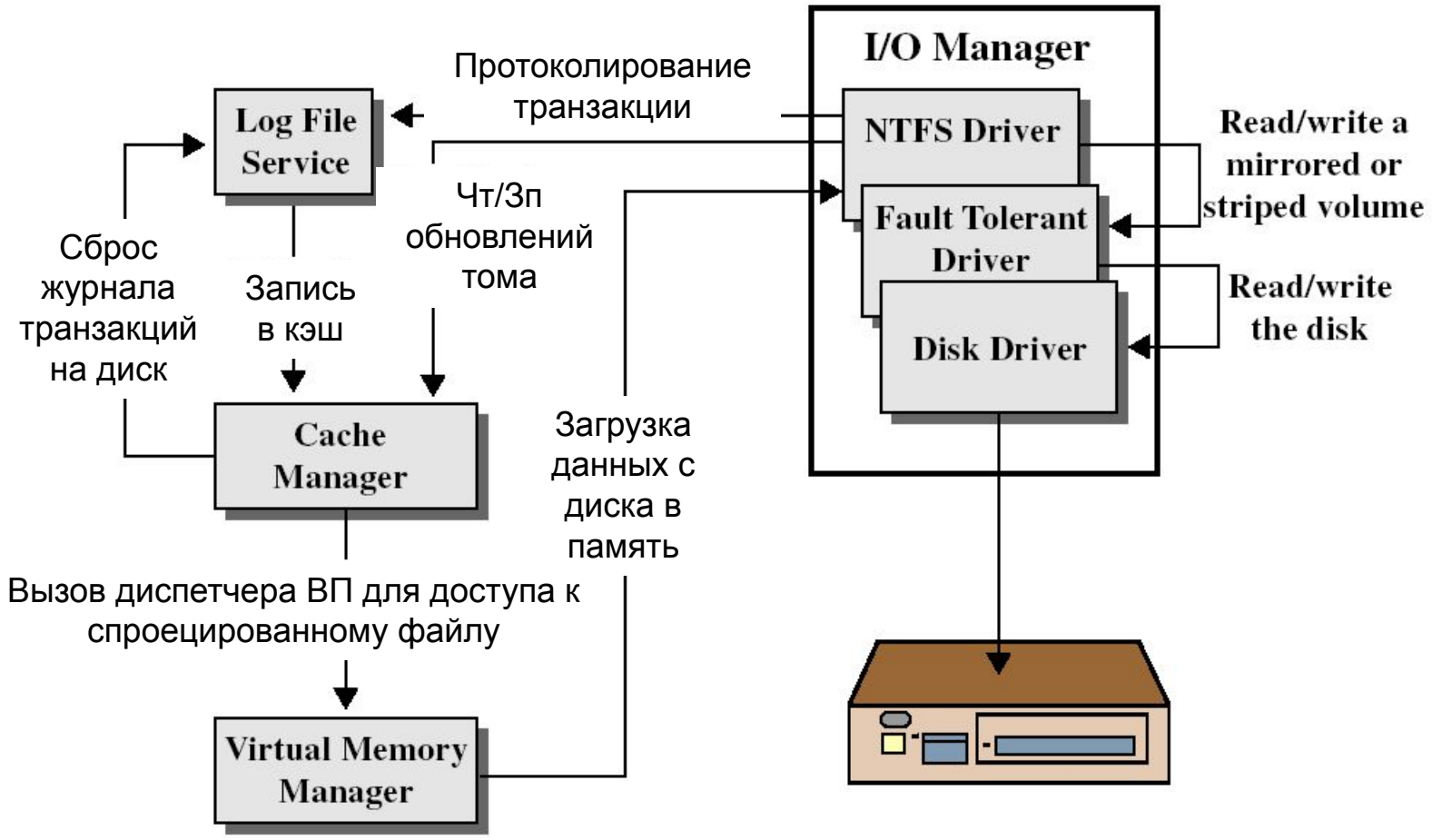
Примеры транзакций NTFS

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

Кэширование в Windows

- Для ускорения операций файлового ввода-вывода в операционных системах Windows используется механизм кэширования.
- Диспетчер кэша обеспечивает файловой системе специализированный интерфейс к диспетчеру виртуальный памяти. 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).

Взаимодействие NTFS со связанными компонентами (1)



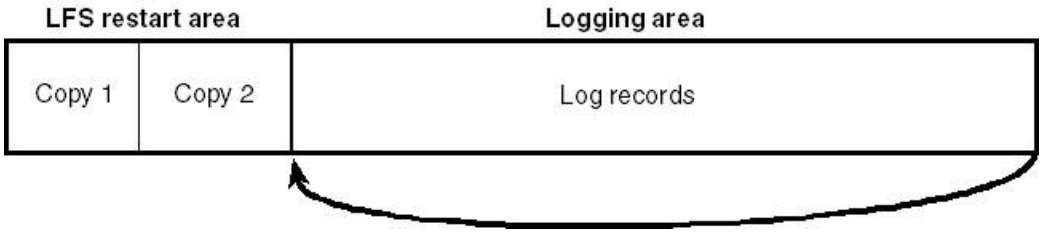
Взаимодействие NTFS со связанными компонентами (2)

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).

Журнал транзакций

- Файл журнала транзакций разбит на две части: область рестарта (restart area) и «бесконечную» область протоколирования (logging area).



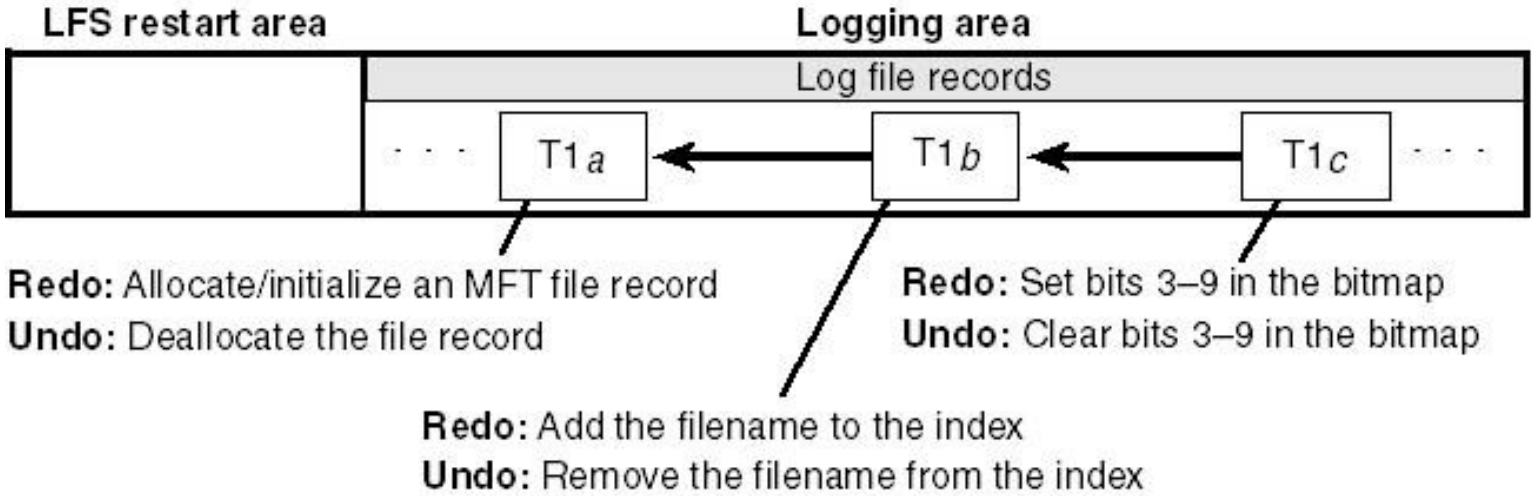
- Файл журнала транзакций является sparse-файлом, что создает иллюзию его бесконечности.

Структура файла журнала

- В области рестарта NTFS хранит информацию контекста, такую как позиция в области протоколирования, откуда она будет начинать чтение при восстановлении после сбоя.
- 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.
- Для идентификации записей, помещенный журнал, LFS использует номера логической последовательности (logical sequence number, LSN).

Записи модификации

- Большинство записей в журнале транзакций – записи модификации.



Структура записи модификации

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

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

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

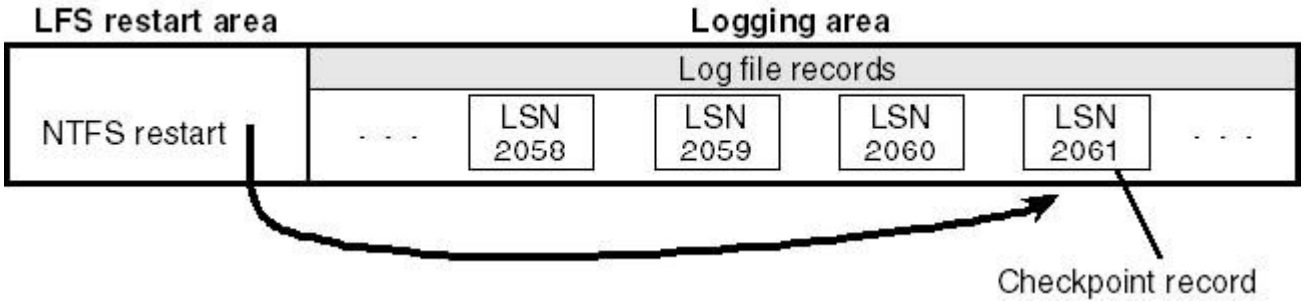
  как устранить изменения, вызванные одной подоперацией транзакции, которая в момент сбоя была запротоколирована лишь частично.

Завершение транзакции

- 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.

Запись контрольной точки

- 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.

Таблицы восстановления

- **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.

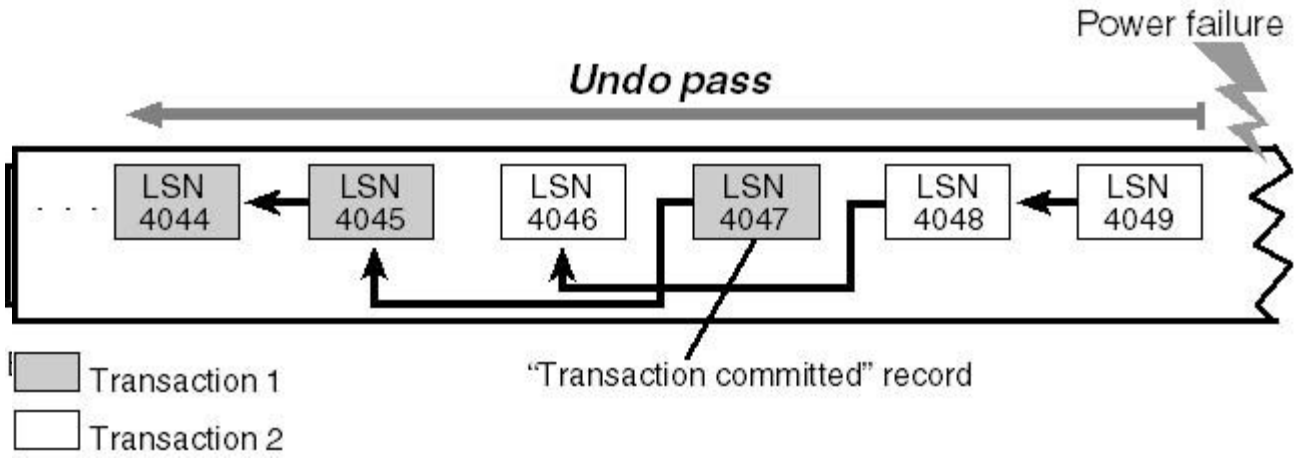
Восстановление после сбоя

- При восстановлении после сбоя системы NTFS просматривает журнал и повторяет все подтвержденные транзакции.
- После повтора всех подтвержденных транзакций NTFS отыскивает в журнале такие, которые не были подтверждены к моменту сбоя, и откатывает (отменяет) каждую запротоколированную подоперацию.

Процесс восстановления

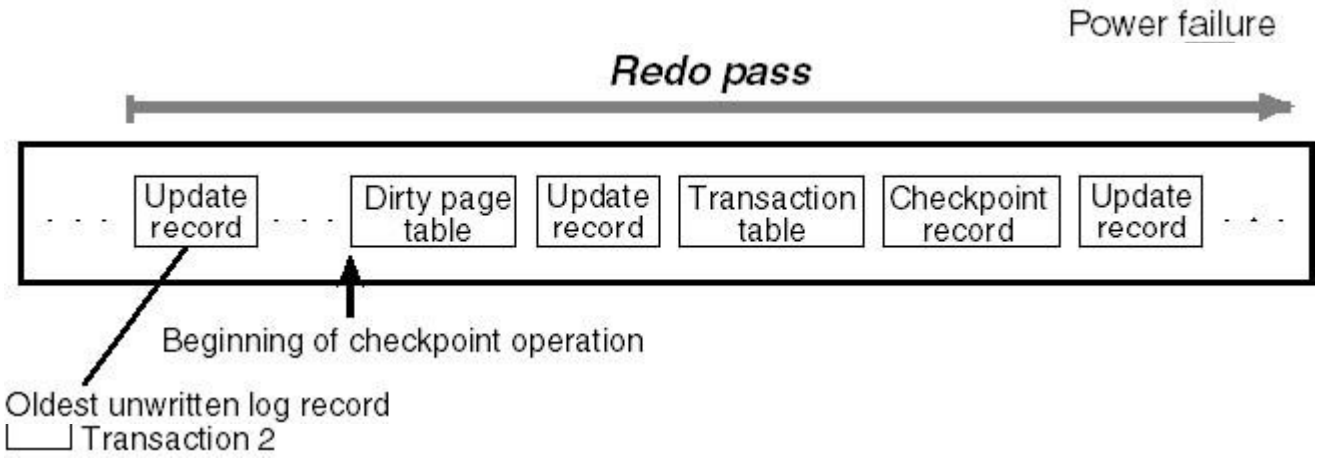
- При восстановлении тома NTFS загружает журнал транзакций в оперативную память и выполняет три прохода:
  - **анализ;**
  - **повтор транзакций;**
  - **отмена транзакций.**

Фаза анализа



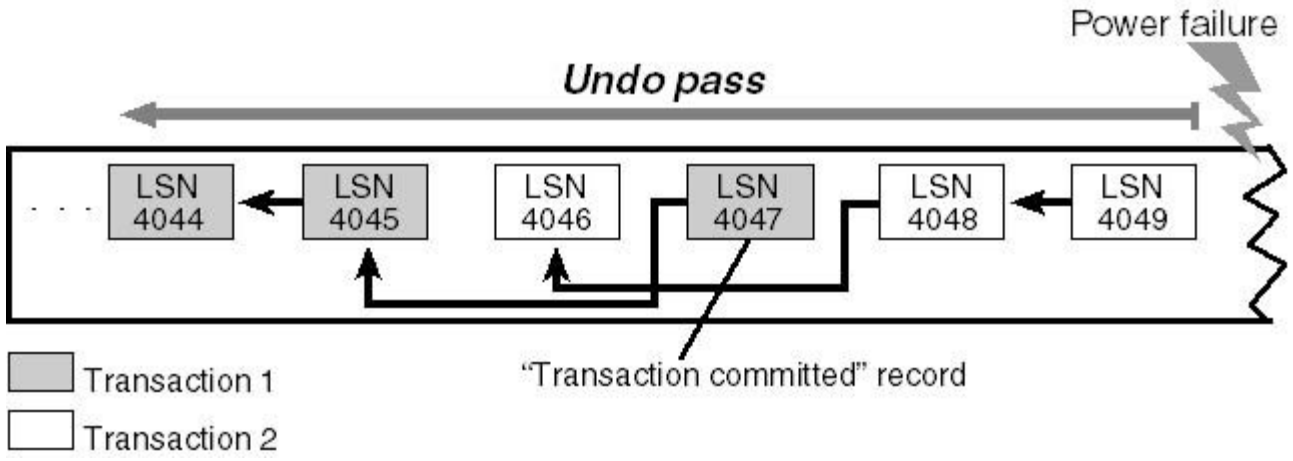
- просмотр журнала транзакций в прямом направлении, начиная с последней операции контрольной точки;
- поиск записей модификации и актуализация таблиц восстановления, начиная с последней контрольной точки;
- определение самой старой записи модификации, которая регистрирует невыполнение диском операции,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

Фаза повтора транзакции



- сканирование журнала транзакций в прямом направлении, начиная с LSN самой старой записи, которая была обнаружена на проходе анализа;
- поиск записей «обновление страницы», содержащих модификации тома, которые были запротоколированы до сбоя системы, но не сброшены из кэша на диск;
- повторение найденных обновлений в кэш, после достижения конца журнала выполняется сброс кэша на диск в фоновом режиме.

Фаза отмены транзакции



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

Файловая система NTFS

Безопасность в NTFS

Безопасность в NTFS

- Защита файлов NTFS на объектном уровне – **диспетчер безопасности**, который 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 для файлов и каталогов

- Full Control (Полный доступ)
- Modify (Изменить)
- Read & Execute (Чтение и выполнение)
- Read (Чтение)
- Write (Запись)

Специальные разрешения для файлов и каталогов (1)

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. 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 ОС.
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.

Специальные разрешения для файлов и каталогов (2)

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

Область действия 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).

Ограничение объектной модели безопасности Windows

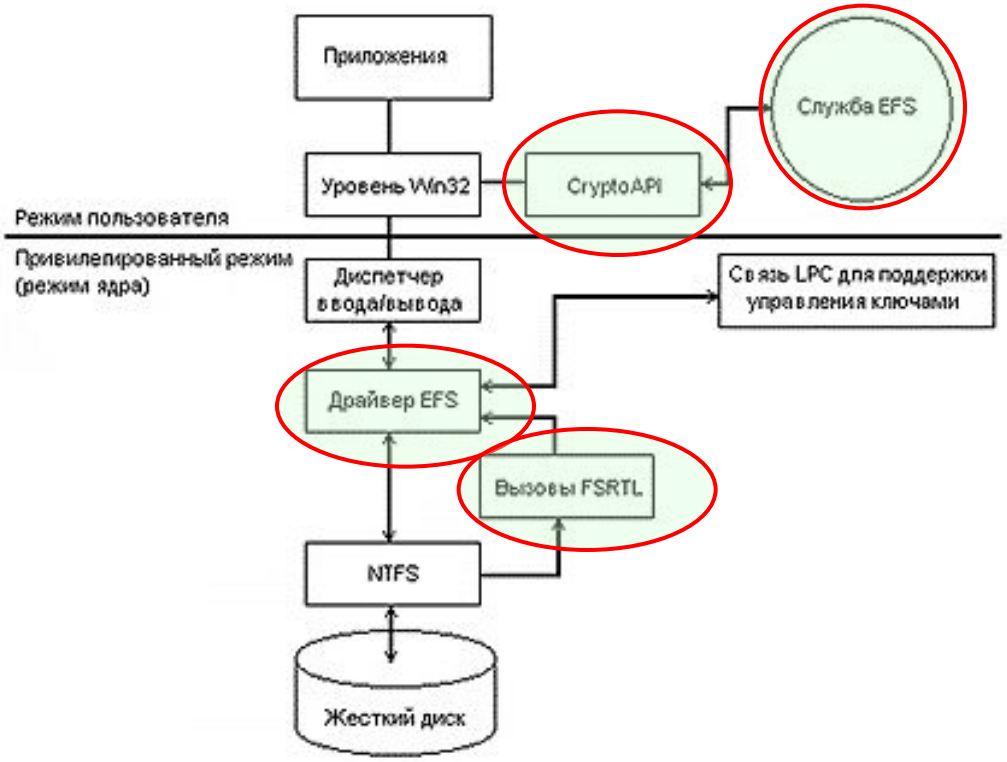
- Если доступ к тому NTFS осуществляется не с помощью средств ОС Windows, а напрямую, на физическом уровне, то средства разграничения доступа и защиты данных от несанкционированного доступа, обеспечиваемые ОС не действуют и данные пользователя могут оказаться беззащитными.
- Такой доступ можно легко 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.

Шифрующая файловая система

- 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 посторонних лиц.
- При использовании EFS исключение возможности несанкционированного доступа к файлам от посторонних лиц выполняется на физическом уровне.

Архитектура EFS

- 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, передает наборы данных.
- 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 FSRTL

- *Библиотека реального времени файловой системы EFS* FSRTL (File System Run Time Library) – 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 NTFS, предназначенный для управления файлами. Это гарантирует, что вся работа с файлами происходит при непосредственном участии NTFS.

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 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.

Алгоритмы шифрования

- В асимметричных системах необходимо применять длинные ключи (512 битов и больше). Длинный ключ резко увеличивает время шифрования. Кроме того, генерация ключей весьма длительна. Зато распределять ключи можно по незащищенным каналам.
- В симметричных алгоритмах используют более короткие ключи, т. е. шифрование происходит быстрее. Но в таких системах сложно распределение ключей.

Технологии шифрования EFS

- Подсистема EFS использует различные симметричные алгоритмы шифрования, зависящие от версии используемой операционной системы Windows 2000+.

Операционная система	Алгоритм шифрования по умолчанию	Альтернативные доступные алгоритмы
Windows 2000	DESX	(none)
Windows XP RTM	DESX	3DES
Windows XP SP1	AES 256	3DES, DESX
Windows Server 2003	AES 256	3DES, DESX
Windows Vista	AES 256	3DES, DESX
Windows Server 2008	AES 256	3DES, DESX (?)

Шифрование с симметричным ключом

- 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.

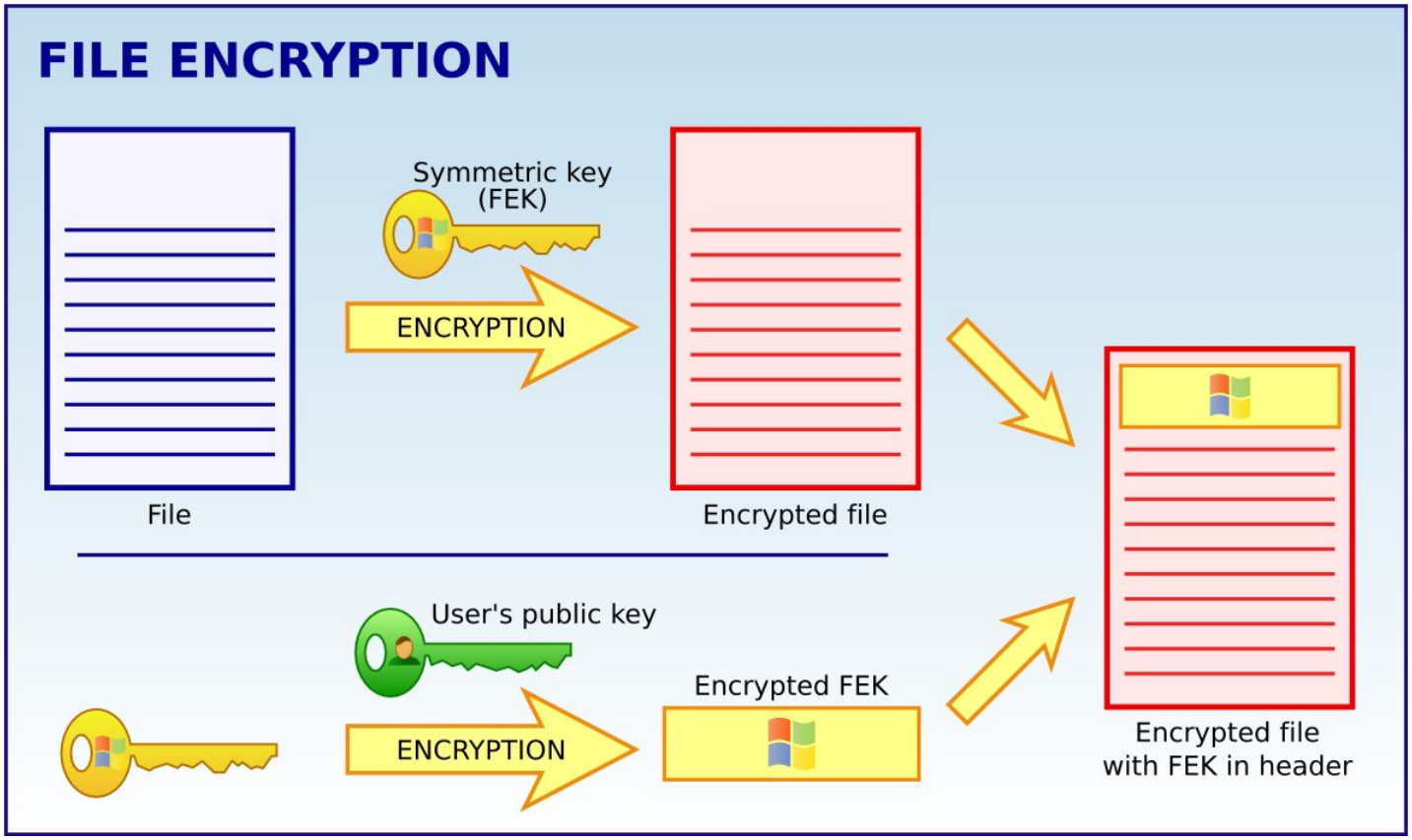
Шифрование и дешифрование

- В EFS используется комбинация симметричного и асимметричного шифрования.
- При использовании симметричного метода файл шифруется и восстанавливается с помощью одного ключа.
- Асимметричный метод заключается в использовании открытого ключа для шифрования и второго, но связанного с ним частного ключа для восстановления данных. Если пользователь, которому предоставляется право восстановления данных, никому не раскрывает частный ключ, защищенному ресурсу ничего не грозит.

Шифрование файлов

- EFS шифрует каждый файл с помощью алгоритма симметричного шифрования, зависящего от версии Windows и настроек.
- При этом используется случайно-сгенерированный для каждого файла ключ симметричного шифрования, называемый **File Encryption Key** (FEK), выбор симметричного шифрования на данном этапе объясняется его скоростью и большей надёжностью по отношению к асимметричному шифрованию.
- FEK защищается путём асимметричного шифрования с использованием открытого ключа пользователя и алгоритма RSA (теоретически возможно использование других алгоритмов асимметричного шифрования).
- Зашифрованный таким образом ключ FEK сохраняется в альтернативном потоке $EFS файла на томе NTFS.

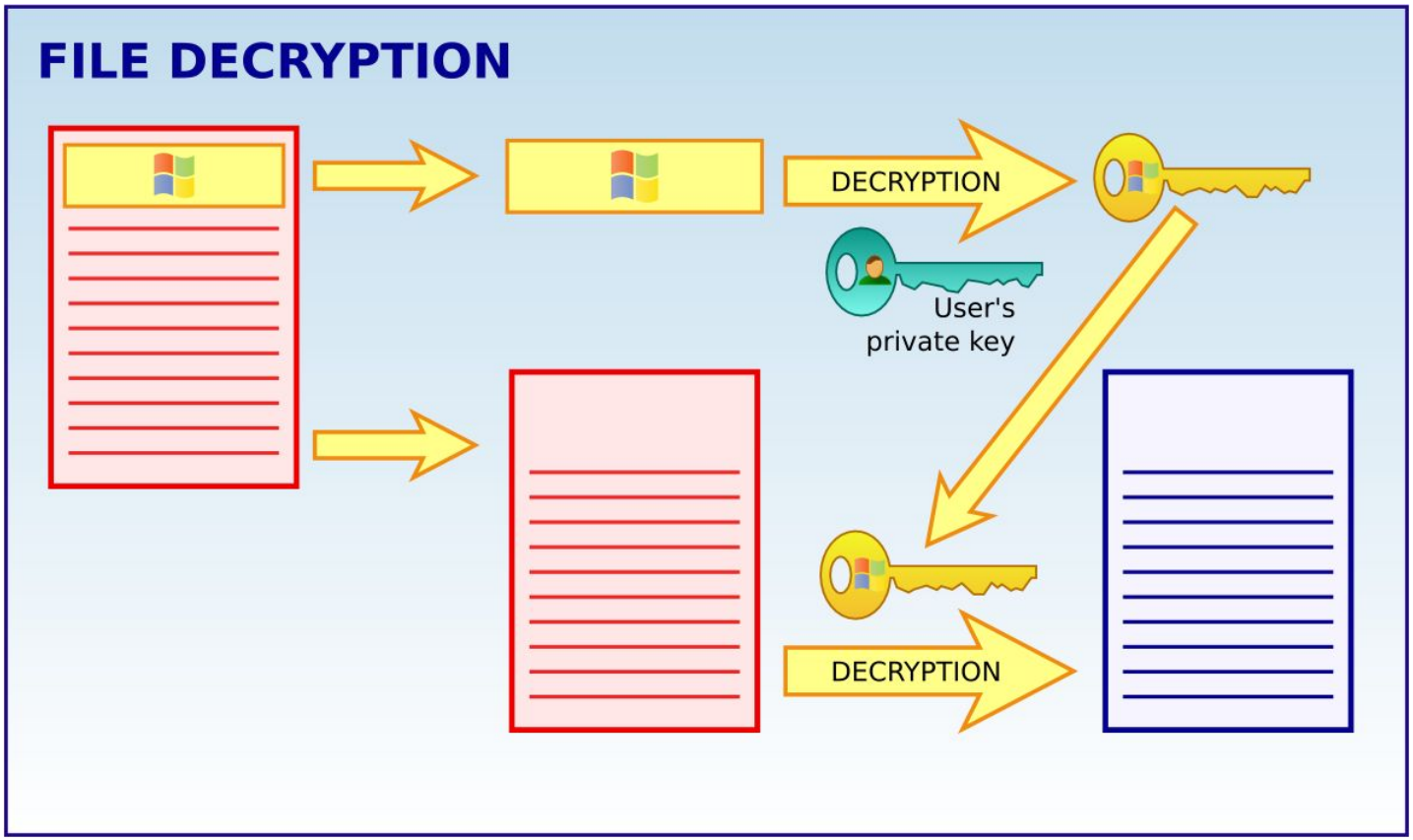
Шифрование файла



Дешифрование файлов

- Для расшифрования данных драйвер EFS, используя закрытый ключ пользователя (RSA 1024), расшифровывает **File Encryption Key** (FEK), сохраненный в альтернативном потоке файла $EFS.
- Затем с помощью расшифрованного файлового ключа драйвер EFS выполняет дешифрацию непосредственно самого файла.

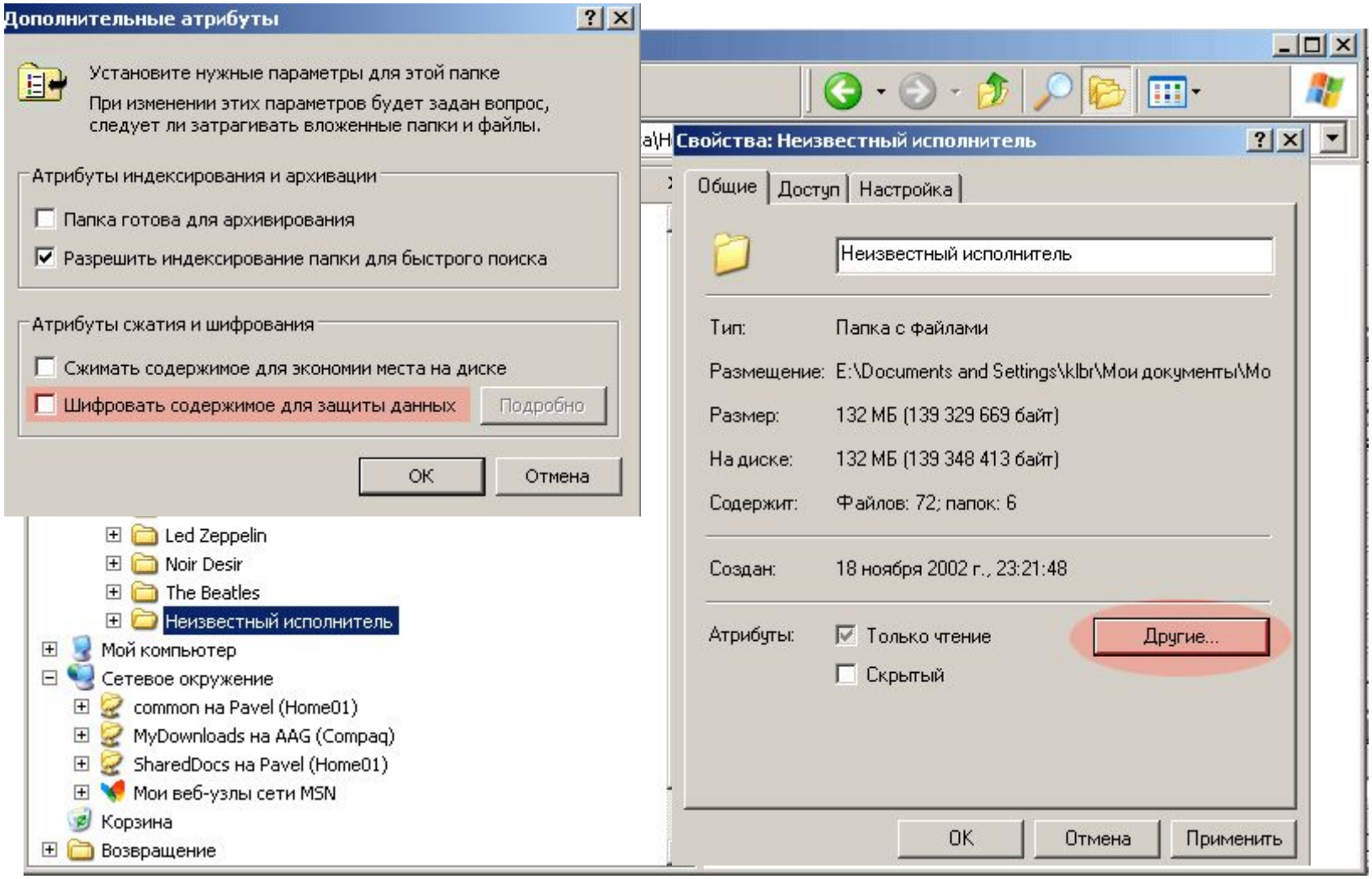
Дешифрование файла



Способы шифрования

- шифрование файлов и папок с использованием Проводника (установка специального атрибута папки/файла)
- шифрование файлов и папок с использованием CryptoAPI
- шифрование файлов и папок с использованием cipher
- копирование файлов и папок в папку, поддерживающую шифрование

Шифрование файлов и папок



Использование утилиты *cipher*

- При необходимости зашифровать файл/папку необходимо использовать:
  - cipher /E <путь к папке>
  - При необходимости расшифровать файл:
  - cipher /D <путь к папке>

Формат утилиты *cipher*

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.

Функции CryptoAPI

- EncryptFile
- DecryptFile
- FileEncryptionStatus
- AddUsersToEncryptedFile
- RemoveUsersFromEncryptedFile
- EncryptionDisable
- …

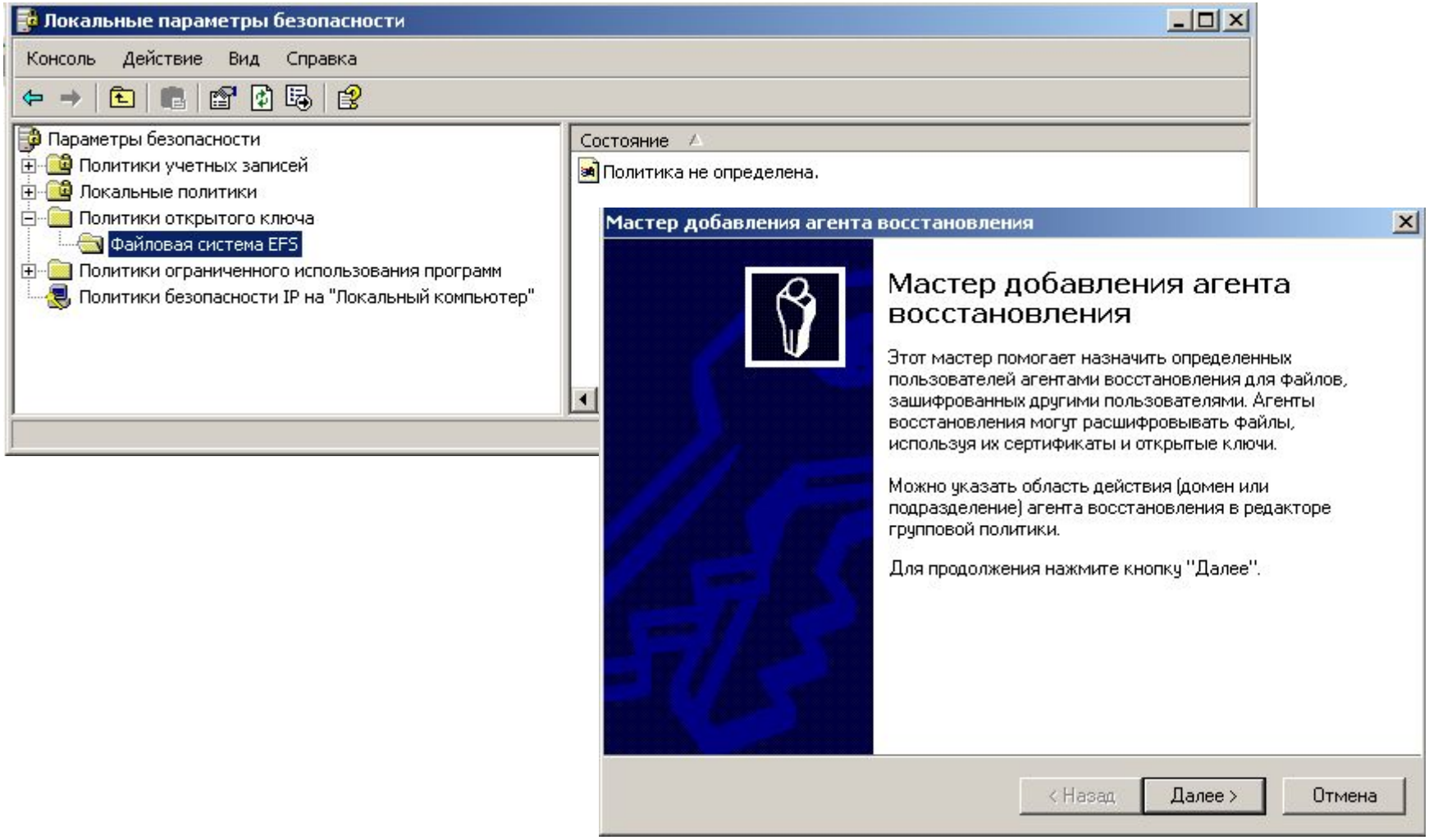
Ограничения EFS (1)

- Могут быть зашифрованы только файлы и папки, находящиеся **на томах NTFS**.
- **Сжатые файлы и папки не могут быть зашифрованы**. Если шифрование выполняется для сжатого файла или папки, файл или папка преобразуются к состоянию без сжатия.
- При перемещении незашифрованных файлов в зашифрованную папку они **автоматически шифруются** в новой папке. Однако обратная операция не приведет к автоматической расшифровке файлов. Файлы необходимо явно расшифровать.
- Не могут быть зашифрованы файлы **с атрибутом «Системный» и файлы в структуре папок системный корневой каталог**.

Ограничения EFS (2)

- Зашифрованные файлы могут стать **расшифрованными**, если файл копируется или перемещается на том, **не являющийся томом NTFS**.
- Шифрование папки или файла не защищает их от удаления. Любой пользователь, имеющий права на удаление, **может удалить зашифрованные папки или файлы**. По этой причине рекомендуется использование EFS в комбинации с разрешениями системы NTFS.
- Могут быть зашифрованы или расшифрованы файлы и папки на удаленном компьютере, для которого разрешено удаленное шифрование. Однако если зашифрованный файл открывается по сети, передаваемые при этом по сети **данные не будут зашифрованы**.

Политика восстановления данных



Шифрование диска с помощью BitLocker

- Программа шифрования BitLocker тесно встроена в ОС Windows, начиная с Windows Vista, Windows Server 2008:
  - BitLocker является встроенным в Microsoft Windows решением для шифрования томов, что обеспечивает повышенную защиту от кражи данных, например, в случаях похищения или утери компьютеров или жестких дисков.
  - BitLocker шифрует все пользовательские и системные файлы на томе ОС, включая файлы подкачки и гибернации. Также возможно шифрование томов данных.
  - Когда BitLocker работает, любой сохраняемый на жестком диске файл будет автоматически зашифрован.

Файловая система NTFS

Дополнительные возможности

Дополнительные возможности NTFS

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

Создание Hard Link

fsutil hardlink create <новый файл> <существующий файл>

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

Точки подсоединения

- Точки подсоединения позволяют выполнять подмонтирование дисковых устройств к папкам на NTFS-томах.
- 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*.
- Использование точек подсоединения позволяет 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.

Утилита *mountvol*

- С помощью утилиты *mountvol* можно:
  - Отобразить корневую папку локального тома в некоторую папку NTFS (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*.

Синтаксис *mountvol*

- mountvol *[устройство:]путь имя_тома*
  - *[устройство:]путь* – определяет существующую папку NTFS, 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.

Фрагментация файлов в NTFS

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

Вопрос

- Почему диск, заполненный более чем на 88%, дефрагментировать практически невозможно?

Дефрагментация NTFS

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

Получение справочной информации об NTFS

fsutil fsinfo

---- Поддерживаемые команды FSINFO----

drives	Отображение всех устройств
drivetype	Отображение типа привода для устройств
volumeinfo	Отображение информации о томе
ntfsinfo	Отображение информации о NTFS
statistics	Отображение статистики файловой системы

Оптимизация NTFS

- **fsutil behavior query** {**disable8dot3**|**allowextchar**|**disablelastaccess**|**quotanotify**|**mftzone**}
- **fsutil behavior set** [{**disable8dot3** {**1**|**0**}|**allowextchar** {**1**|**0**}|**disablelastaccess** {**1**|**0**}|**quotanotify** *частота*|**mftzone** *значение*}]

Примеры оптимизации NTFS

- Отключите обновление сведений о последнем доступе к файлу

  fsutil behavior set disablelastaccess 1

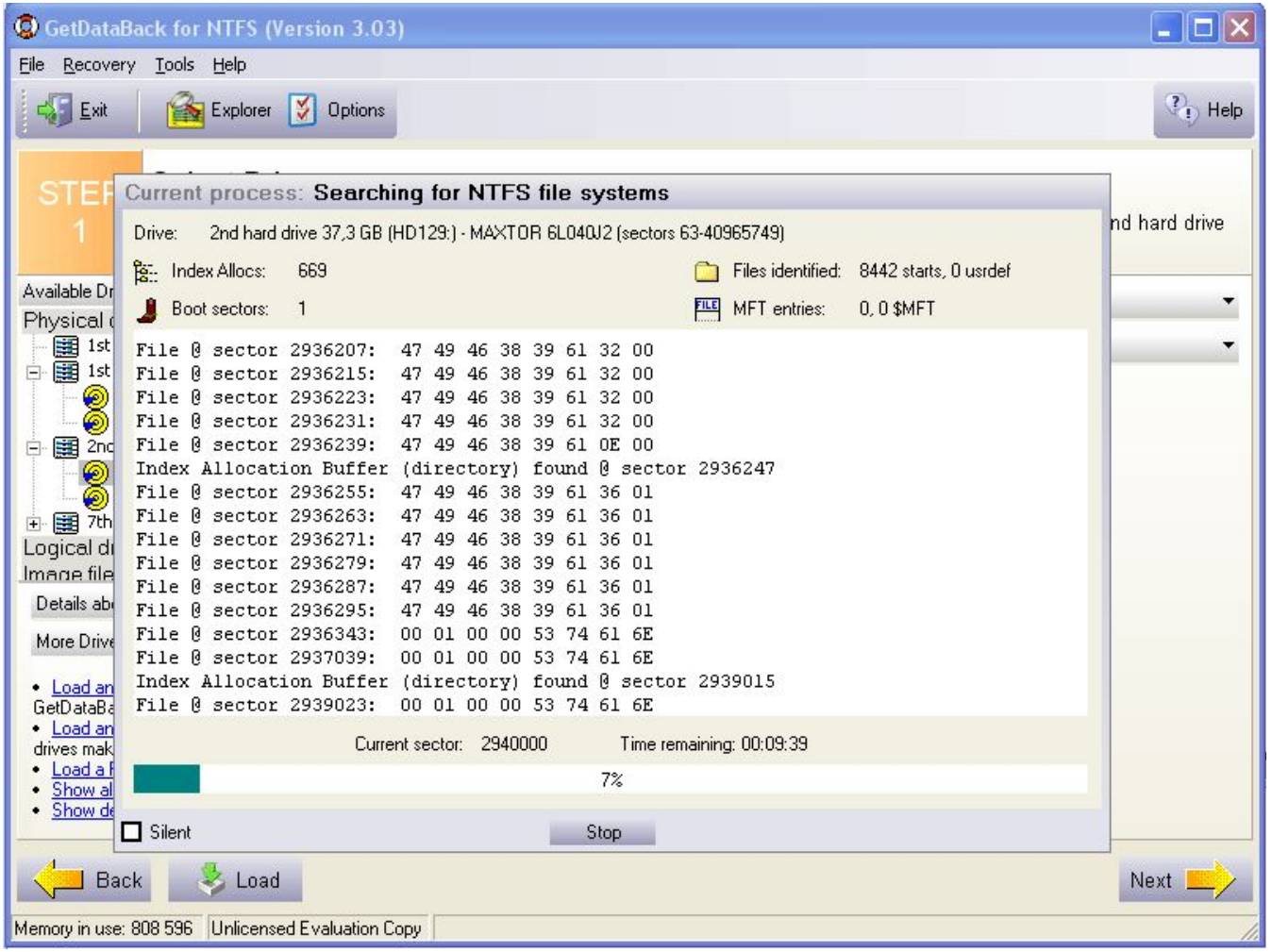
- Зарезервируйте необходимое пространство для MFT

  fsutil behavior set mftzone <значение>

- Отключите создание коротких имен файлов 8.3

  fsutil behavior set disable8dot3 1

GetDataBack for NTFS



Развитие NTFS

- 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.

Файловая система NTFS

NTFS vs. FAT

Поиск данных файла

- **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.
- В системе **FAT16**, где максимальный размер области FAT составляет 128 Кбайт, вся FAT считывается с диска целиком и буферизируется в оперативной памяти.
- **FAT32** же, напротив, имеет типичный размер области FAT порядка сотен килобайт, а на больших дисках – даже несколько мегабайт. Для доступа к фрагменту файла в системе FAT16 и FAT32 приходится обращаться к соответствующей ячейке таблицы FAT.

Поиск свободного места

- 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.

Работа с каталогами и файлами

- **FAT16** и **FAT32** имеют очень компактные каталоги, размер каждой записи которых всего 32 байта, а сам каталог FAT обычно укладывается в один кластер. Благодаря этому в подавляющем числе случаев каталог не фрагментирован и работа с каталогами производится достаточно быстро. Единственная проблема – высокая трудоемкость поиска файлов в больших каталогах.
- **NTFS** использует гораздо более эффективный способ адресации – бинарное дерево. Размер каталога NTFS зачастую превышает типичный размер одного кластера, поэтому каталоги могут быть сильно фрагментированы. Это, в свою очередь, может уменьшить положительный эффект от более эффективной организации самих данных.

Итоги сравнения

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

Итоги сравнения

- 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.
- Тома NTFS могут теоретически быть объемом до 16 эксабайт (ЭБ), но предел при разметке диска с использованием разметки MBR составляет 2 ТБ. Чтобы обойти это ограничение, необходимо использование динамических дисков или разметки GPT, поддерживающей разделы диска размером до 9.4 зетабайт (ЗБ).
- Пользователь может определить размер кластера при форматировании тома NTFS. Однако NTFS-сжатие не поддерживается при размере кластера больше 4КБайт.

Файловая система NTFS

Развитие NTFS

ReFS (Resilient File System)

- ReFS основана на NTFS и сохраняет совместимость по ключевым направлениям, предназначена для серверных версий ОС Windows.
- Некоторые возможности NTFS ликвидированы, в том числе поддержка коротких имён, компрессия, шифрование на уровне файлов (EFS), дисковые лимиты (квоты), потоки данных, транзакции, разреженные файлы, расширенные атрибуты и жёсткие ссылки.
- Максимальный размер единого тома – до 2^{78} байт с размером кластеров 16 КБ ($2^{64} * 16 * 2^{10}$).
- Максимальный размер файла = 2^{64} – 1 байт.
- Максимальная длина имени файла = 32 767 символов Юникод.