Оглавление

1 Файловые системы FAT

1.1 *Версии файловой системы*

Физическая реализация файлов в в файловых системах FAT – связанный список с таблицей размещения. Атрибуты файлов хранятся в записях каталога.

Цифры в названиях конкретных версий FAT означают разрядность адреса кластера в элементе таблицы FAT.

Размер кластеров выбирается из диапазона от 512 байтов до 64 кБ.

FAT12 используется для форматирования небольших дисков (до 16 мБ). Поддерживает только короткие имена файлов по схеме «8.3».Эта ФС соответствует первым версиям ОС MS DOS.

FAT16 используется для дисков до 2Гб (небольших винчестеров). Максимальный размер раздела под FAT16 ограничен 4 гБ (65536 кластеров по 64 кБ). Соответствует ОС MS DOS и Windows.

FAT32 – файловая система для дисковых накопителей большой ёмкости. Использует кластеры разного объема: 4 кБ для дисков до 8 гБ; 8, 16, 32 кБ – для дисков большего объема. Максимальный размер раздела практически не ограничен – 2^{32} кластеров по 32 кБ. Поддерживает короткие и длинные имена файлов. Соответствует всем ОС Windows.

1.2 *Структура раздела с файловой системой FAT*

В файловой системе FAT любого логического диска делится на две области:

системная область – создается и инициализируется при форматировании и обновляется при манипулировании файловой структурой.

Системная область состоит из:

загрузочной записи (boot record, BR),

зарезервированных секторов (reserved sector, ResSecs),

таблицы размещения файлов (file allocation table, FAT),

корневого каталога (root directory, Rdir)

область данных – содержит файлы и каталоги, подчиненные корневому. Она, в отличие от системной области, **доступна через пользовательский интерфейс операционной системы.**

1.3 *Концепция файловой системы FAT*

FAT-таблица содержит информацию о кластерах. Размер таблицы равен общему количеству кластеров на логическом диске.

Файл занимает один или несколько кластеров на диске. При этом элементы FAT - таблицы, описывающие пользуемые данным файлом кластеры, связаны в список. Конец списка отмечен в таблице специальным значением. **Номер первого кластера**, распределенного файлу, хранится **в элементе каталога**, описывающего данный файл.

1.4 *Общая схема использования FAT*

1. Получают номер первого кластера файла, для которого необходимо определить его расположение на диске.

2. Номер первого кластера используют как индекс в FAT - таблице для извлечения номера следующего кластера.

3. Повторяют предыдущую процедуру до тех пор, пока извлеченное из FAT - таблицы значение не будет соответствовать концу файла.

Используя описанную выше процедуру можно определить для каждого файла цепочку занимаемых им кластеров.

Для нахождения **первого кластера**, распределенного файлу необходимо прочитать информацию из каталога, в котором содержится данный файл. Для этого необходимо сначала прочитать корневой каталог, а затем все подкаталоги из пути каталогов к данному файлу.

Корневой каталог находится сразу за последней копией FAT. Размер корневого каталога можно определить исходя из значения поля **root_size**. При форматировании диска в это поле записывается максимальное количество файлов и каталогов, которые могут быть в корневом каталоге. Для каждого элемента в каталоге отводится 32 байта, поэтому корневой каталог имеет длину 32*root_size байт. Корневой каталог занимает непрерывную область фиксированного размера (кроме FAT32).

Любой каталог одержит 32-байтовые дескрипторы, описывающие файлы и другие каталоги.

В любом каталоге, кроме корневого, два первых дескриптора имеют специальное назначение. Первый дескриптор содержит в поле имени строку «.». Этот дескриптор указывает на содержащий его каталог, т.е. каталог имеет ссылку на самого себя.

Второй специальный дескриптор содержит в поле имени строку «..». Это дескриптор указывает на каталог более высокого уровня

Если в поле номера первого занимаемого кластера для дескриптора с именем «..» находится нулевое значение, это означает, что данный каталог содержится в корневом каталоге.

1.5 *Файловая система FAT32*

Основное отличие файловой системы FAT32 заключается в том, что в самой таблице размещения файлов на номер кластера отводится не 2 байта как в FAT16(или 1.5 байта в FAT12), а четыре (надо отметить, что старшие 4 бита не входят в номер кластера, а зарезервированы для системных нужд).

Хотя это и увеличивает размер самой таблицы (максимально доступно не 65 тысяч (2^{16}) 2х байтовых записей, а 2.6 миллиона (2^{28}) 4х байтовых), но это позволило эффективнее использовать дисковое пространство, за счет уменьшения размера кластера и соответственно уменьшения потерь в “хвостах” кластеров (для файлов, чей размер не кратен размеру кластеров).

Максимальная длина имени файла в FAT32 составляет **256 символов** (включая завершающий символ 0x00).

Максимальная длина имени каталога (пути без имени файла) составляет **246 символов**.

Максимальная, полная длина имени файла (включая символ устройства, двоеточие, обратную косую черту и 0x00) составляет **260 символов.**

В системе FAT32, когда создается файл с длинным именем, автоматически для этого длинного имени создается псевдоним в стандартном формате имени 8.3. Этот псевдоним делает файл с длинным именем доступным из обычных MS-DOS приложений.

В псевдониме имени файла или каталога допускается применять следующие символы: 0÷9, A÷Z, a÷z, пробел, символы: $ % ' - _ @ ~ ` ! ( ) { } ^ # &, а также все символы с ASCII кодами более 127. Таким образом в VFAT допускается применение в именах файлов русских символов.

В имени файла кроме символов допустимых в псевдониме можно использовать еще и символы: + , ; = [ ]

При обращении или поиске файлов строчные и прописные буквы отождествляются в соответствии с текущей кодовой страницей ОС.

Физически, на диске, псевдонимы хранятся используя символы OEM набора текущей кодовой страницы ОС, а длинные имена хранятся в Unicode кодировке.

Псевдонимы имен файлов (короткие имена файлов)

При создании на диске файла с длинным именем автоматически создается соответствующее ему короткое имя в формате 8.3, так называемый псевдоним (alias). Применение псевдонимов позволяет приложениям, не поддерживающим длинные имена файлов, все же, получать доступ к файлам и дискам.

Псевдоним создается по следующей схеме:

Если длинное имя файла соответствуют стандарту 8.3, псевдоним будет иметь такое же имя, но записанное прописными буквами (например для длинного имени *Example.Txt*, псевдоним будет таким *EXAMPLE.TXT*).

Если длинное имя файла не соответствуют стандарту 8.3, генерируется уникальный псевдоним:

Производится попытка создать короткое имя, состоящее из 6 символов длинного имени и числового суффикса из тильды (~) и цифры, система сначала использует цифру 1 (суффикс ~1), если файл с таким именем существует, используется цифра 2, затем 3 и т.д. Если длинное имя имеет расширение, в псевдониме используется расширение из первых 3х символов длинного расширения.

Если двух буквенный суффикс не позволяет создать уникальное имя, он увеличивается до 3х символов т.д. (например для имени *Long File Name.File* если певдонимы *LONGFI~1.FIL* до *LONGFI~9.FIL* заняты, будет сгенерирован псевдоним *LONGFI~10.FIL* if the names *LONGFI~1.FIL*).

В длинном имени файла нет расширения, как такового, точка это обычный символ в имени файла, их может быть несколько, однако имя файла не может оканчиваться точкой (все конечные точки отбрасываются). При генерации псевдонима за расширение принимаются символы после последней точки в имени. Так для имени *MyFile.081293.Document* псевдонимом будет

MYFILE~1.DOC. Имя файла может начинаться с точки *.login*, псевдонимом будет *LOGIN~1*.

Как имя файла, так и псевдоним должны быть уникальными в данном каталоге. При копировании или редактировании файла, его псевдоним может изменится и не соответствовать оригинальному псевдониму. Так, если скопировать файл с длинным именем *LongFileName* и псевдонимом *LONGFI~2* в другой каталог (используя длинное имя), то его псевдоним на новом месте будет *LONGFI~1* (если конечно файл с таким псевдонимом уже не существует). В результате копирования система всегда генерирует новый псевдоним, не конфликтующий с уже существующими, таким образом, программы не должны полагаться на псевдоним файла, а всегда использовать его полное, длинное имя. Однако можно открывать, читать и записывать в файл, используя его псевдоним, эти операции не разрушают длинное имя.

Система автоматически пытается сохранить длинные имена, даже когда файл переименовывается по его псевдониму (т.е. когда это делает MS-DOS программа), прежде чем выполнить перемещение или удаление система собирает и сохраняет атрибуты файла и его длинное имя, после выполнения операции удаления или перемещения, система в течении некоторого времени (по умолчанию 15 секунд) ждет создания или переименования файла с таким же псевдонимом, если система определит такое событие, новому файлу назначаются сохраненные ранее атрибуты и длинное имя. Такая схема позволяет сохранить длинное имя при работе с устаревшими программами.

Основные преимущества системы FAT32 над другими версиями FAT:

- **Поддержка дисков размером до 2х терабайт**
- **Более эффективное использование дискового пространства**
- **Высокая устойчивость за счет наличия в файловой системе специальных резервных областей. Автоматическое использование второй копии FAT, если первая повреждена.**

- **Большая гибкость. В FAT32 нет ограничения на число элементов в корневом каталоге, как это было в прежних версиях FAT.**

2 Файловая система NTFS

2.1 *Основные характеристики NTFS.*

□ ***Восстанавливаемость.*** После отказа гарантировано восстановление согласованного состояния файловой системы. На случай повреждения отдельных дисковых блоков, в которых хранятся системные данные, существуют копии всех системных данных, включая журнал обновлений.

□ ***Защищенность.*** Аутентифицированный вход в систему и проверка прав доступа к каждому файлу с использованием списка контроля доступа обеспечивают защиту от несанкционированных файловых операций.

□ ***Избыточность данных и отказоустойчивость***. Необходима приложениям, для которых недопустима потеря данных из-за отказа носителя. Многослойная структура системы ввода-вывода позволяет динамически загружать необходимые дисковые драйверы, в частности обеспечить зеркалирование и чередование дисков.

□ ***Поддержка больших дисков и файлов***. Размер полей данных, в которых хранятся значения объемов диска и файла, достаточен для того, чтобы не служить ограничением. Кроме того, каждый компонент путевого имени (файла или каталога) может иметь длину до 255 символов.

□ ***Поддержка многих потоков данных***. В NTFS каждая единица информации, связанная с файлом, в том числе его имя, а также имя владельца, содержимое и т. д., представлена как атрибут файла. Каждый атрибут состоит из одного потока, то есть последовательности байтов. Можно добавлять к файлу новые атрибуты, включая дополнительные именованные потоки содержимого. В частности, один поток может использоваться для основных операций над данными, а другой для протоколирования. Разделителем между именем файла и именем потока данных служит двоеточие, например: **niyfile.dat:** stream2

□ ***Хранение имен в кодировке Unicode***. Благодаря использованию универсальной 16-разрядной кодировки имен файлы можно свободно переносить с одного компьютера на другой. Такая разрядность обеспечивает уникальное представление каждого символа существующих естественных языков. Имена файлов могут содержать пробелы и произвольное количество точек.

□ ***Универсальное средство индексации***. Архитектура NTFS позволяет индексировать атрибуты файлов тома, благодаря чему возможен быстрый поиск файлов, удовлетворяющих заданному критерию. Кроме того, NTFS может сортировать файлы по заданному атрибуту.

□ ***Замена секторов***. Суть технологии заключается в динамической замене потерянных данных, когда сектор диска становится нечитаемым. Для этого используется метод их избыточного хранения. Если отказоустойчивый драйвер не был загружен, NTFS заменяет поврежденный сектор и больше его не использует, но восстановить его данные не может.

□ ***Поддержка POSIX***. В соответствии с требованиями этого стандарта в NTFS реализована поддержка имен файлов и каталогов, различающихся только регистром букв и отметкой времени изменения файла.

□ ***Сменные диски***. Сменные диски, отформатированные для NTFS, защищены теми же механизмами контроля доступа, что и постоянные.

2.2 *Структура базового формата файловой системы*

Каждый дисковый раздел NTFS содержит файлы, каталоги, битовые массивы и другие структуры данных.

Каждый раздел организован как линейная последовательность **блоков** (кластеров). Размер **блока** фиксирован для каждого раздела и варьируется в пределах от 512 байт до 64 Кбайт в зависимости от размера раздела.

Для большинства дисков NTFS используются блоки размером в 4 Кбайт, как компромисс между большими блоками (для эффективности операций чтения/записи) и маленькими блоками (для **уменьшения** потерь дискового пространства на внутреннюю фрагментацию).

Адресация блоков осуществляется по их смещению от начала раздела, используются 64-разрядные числа.

Главной структурой данных в каждом раздела является **главная файловая таблица MFT (Master File Table)**, представляющая собой линейную последовательность записей фиксированного **(1 Кбайт)** размера

2.3 *Структура таблицы MFT.*

Каждая запись MFT описывает один файл или один каталог.

В ней содержатся

- **атрибуты файла**,
- **список дисковых адресов** блоков файла.

Если файл очень большой, то используются две и более записей MFT, чтобы вместить список всех блоков файла. В этом случае первая запись MFT, называемая **базовой записью,** указывает на другие записи MFT

Свободные элементы MFT учитывается в битовом массиве.

Сама главная файловая таблица представляет собой файл и, как и любой файл может располагаться в любом месте тома, тем самым устраняется проблема дефектных секторов на первой дорожке дискового раздела.

Кроме того, этот файл может, при необходимости, расти до максимального размера в **218** записей.

Первые 16 записей MFT зарезервированы для описания **файлов метаданных NTFS.**

Каждая такая запись описывает обычный файл, у которого есть атрибуты и блоки данных, как у любого файла. У каждого файла метаданных есть имя, начинающееся с символа доллара, указывающего на то, что это файл метаданных.

Первая запись MFT (Запись 0) описывает сам файл MFT. В частности, она содержит информацию о расположении блоков файла **MFT,** что позволяет системе найти файл **MFT**.**Номер первого блока файла MFT содержится в загрузочном блоке, куда он помещается при установке системы.**

Запись *1* указывает на дубликат первой части файла MFT. Наличие второй копии может быть необходимо на случай, если один из первых блоков главной файловой таблицы вдруг станет дефектным.

Запись *2* указывает на журнал. Когда в файловой системе производятся изменения (такие как добавление нового каталога или удаление существующего каталога, изменения атрибутов файлов), информация о предстоящей операции регистрируется в журнале. Таким образом, увеличивается вероятность корректного восстановления файловой системы в случае сбоя во время выполнения операции. В этом журнале не регистрируются только изменения данных пользователя.

Запись *3* содержится информация о томе (размер, метка и версия).

Запись *4 (* Файл ***$AttrDef***) список стандартных атрибутов файлов .

Запись*5* Содержатся данные о корневом каталоге, который сам представляет собой файл и может произвольно увеличиваться в размерах.

Запись *6.* Свободное место на диске учитывается с помощью битового массива. Битовый массив сам является файлом, и его атрибуты и дисковые адреса хранятся в таблицы MFT.

Запись *7.* указывает на файл начальной загрузки.

Запись *8* используется для того, чтобы связать вместе все дефектные блоки и гарантировать, что они никогда не встретятся в файлах.

Запись 9 содержит информацию о защите.

Запись 10 используется для преобразования регистра для символов латинского алфавита и национальных алфавитов. Этот файл содержит необходимые инструкции.

Запись 11 представляет собой каталог, содержащий различные файлы для дисковых квот, идентификаторов объектов, точек повторного анализа и т. д.

Последние четыре записи MFT зарезервированы на будущее.

2.4 *Структура записи MFT.*

Каждая запись MFT состоит из заголовка записи, за которым следует последовательность пар (заголовок атрибута, значение).

Заголовок записи содержит:

- число, используемое системой для проверки действительности записи;
- порядковый номер, обновляемый каждый раз, когда запись используется для нового файла;
- счетчик обращений к файлу;
- действительное количество байт, используемых в записи;
- идентификатор (индекс, порядковый номер) базовой записи (используемый только для записей расширения),
- другие поля.

Следом за заголовком записи располагается **заголовок первого атрибута**, за которым идет **значение первого атрибута**, потом заголовок второго атрибута, значение второго атрибута и т. д.

В файловой системе NTFS определено **13 атрибутов**, которые могут появляться в записях MFT. Некоторые из них перечислены в **таблице.**

Заголовок атрибута идентифицирует следующий за ним атрибут, а также содержит длину и расположение поля значения вместе с разнообразными флагами и прочей информацией.

Значения атрибутов, как правило, располагаются непосредственно за заголовками, но если длина значения слишком велика, чтобы поместиться в запись MFT, она может быть помещена в отдельный блок диска. Такой атрибут **называется нерезидентным атрибутом.** Например, таким атрибутом является **атрибут данных**.

Некоторые атрибуты, такие как атрибуты имени, могут повторяться, но все атрибуты должны располагаться в записи MFT в фиксированном порядке.

Длина заголовков резидентных атрибутов 24 байт, заголовки для нерезидентных атрибутов длиннее, так как они содержат информацию о месте расположения атрибута.

Стандартное информационное поле содержит

- сведения о владельце файла,
- информацию о защите,
- временные штампы, необходимые для стандарта POSIX,
- счетчик жестких связей,
- бит «только чтение», архивный бит и т. д.
- Это поле имеет фиксированную длину, и оно всегда присутствует.

2.5 *Именование файлов*

Длина имени файла в системе NTFS ограничена **255 символами.**

Полная длина пути ограничивается **32 767 символами**.

Для имен файлов используется кодировка **Unicode**, что позволяет пользователям в странах; в которых не используется латинский алфавит писать имена файлов на своем родном языке.

Файловая система NTFS **полностью поддерживает имена, чувствительные к регистру** (таким образом, ***foo*** отличается от ***Foo*** и ***FOO)***. К сожалению, интерфейсом Win32 API не полностью поддерживается чувствительность к регистру для имен файлов и совсем не поддерживается для имен каталогов, поэтому это преимущество теряется при обращении к программам, обязанным использовать интерфейс Win32.

2.6 *Потоки данных*

Файл в системе NTFS состоит из множества атрибутов, каждый из которых представляется в виде потока байтов.

Большинство файлов имеет **несколько коротких потоков, таких как имя файла и его 64-битовый идентификатор,** плюс **один длинный (неименованный) поток с данными.**

Однако у файла **может быть и несколько длинных потоков данных**. При обращении к каждому потоку после имени файла через двоеточие указывается имя потока, например ***foo:stream1***. **У каждого потока своя длина**. Каждый поток может блокироваться независимо от остальных потоков.

Например**, программа редактирования фотографий** может использовать **неименованный поток для основного изображения**, а **именованный поток — для небольшой пиктограммы**. Эта схема проще, чем традиционный способ, при котором изображения помещаются в один и тот же файл, одно за другим. Другой пример использования потоков данных - электронная обработка текста. Эти программы часто создают две версии документа, временную для использования во время редактирования и окончательную версию, когда пользователь закончил работу. Если поместить временную версию в именованный поток, а окончательную версию в неименованный поток, обе версии автоматически оказываются в одном файле и без какой-либо дополнительной обработки пользуются одинаковыми правами доступа, временными штампами и т. д.

Максимальная длина потока составляет 2^{64} байт. Для отслеживания местонахождения в каждом потоке используются 64-разрядные файловые указатели. Максимальный же размер потока составляет около 18,4 экзабайт

Процедура открытия файла возвращает **дескриптор файла**, который затем может использоваться для чтения этого файла или записи в файл. Для графических приложений заранее не определены указатели в файлах.

Имя потока данных располагается в заголовке атрибута.

Следом за этим заголовком располагается

□ либо список дисковых адресов, определяющий положение файла на диске

□ либо, для файлов длиной всего в несколько сотен байтов (а таких файлов довольно много), **сам файл**.

Метод помещения самого содержимого файла в запись MFT называется непосредственным файлом.

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

Для увеличения эффективности дисковые блоки файлам назначаются по возможности в виде серий последовательных блоков (сегментов файла).

Например, если первый логический блок файла помещается в блоке 20 на диске, тогда система будет стараться выделить для второго блока этого файла блок 21, для третьего - блок 22 и т. д.

Блоки в файле описываются последовательностью записей, каждая из которых описывает последовательность логически непрерывных блоков. Непрерывный файл описывается всего одной записью. К этой категории относятся файлы, записываемые за одну операцию от начала до конца.

Каждая запись начинается с заголовка, определяющего смещение первого блока в файле. Затем располагается смещение первого блока, не покрываемого первой записью.

Следом за каждым заголовком располагаются пары, в которых содержатся дисковые адреса и длины серий блоков.

Эти дисковые адреса представляют собой смещение блока от начала дискового раздела. Длина серии — это количество блоков в серии. В записи серии может содержаться любое необходимое количество пар.

Число таких серий зависит от того, насколько удачно процедура предоставления дискового пространства сумела найти место для хранения файла при его создании. Для файла, состоящего из n блоков, количество серий может быть любым от **1** до n.

Данный способ представления информации о расположении блоков файла на диске не накладывает никаких дополнительных ограничений на размер файла. Для каждой пары требуется два 64-разрядных числа, что составляет 16 байт на пару, Однако одна пара может указывать на миллион последовательных блоков. Например, 20-мегабайтный файл, состоящий из 20 сегментов по 1 млн. килобайтных блоков каждый, легко может быть описан всего одной

записью MFT, тогда как 60-килобайтный файл, состоящий из 60 изолированных блоков, не может.

Хотя при использовании простого метода представления пары требуется 2x8 байт, эти 16 байт могут быть сжаты до меньшего размера. Многие дисковые адреса содержат большое количество нулей в старших байтах. Нули могут быть опущены. В этом случае в заголовке данных будет содержаться информация о том, сколько байтов пропущено, то есть, сколько байтов фактически используется для дискового адреса. Также используются и другие методы сжатия данных. На практике пары часто занимают всего 4 байта.

Если файл окажется настолько велик, или настолько фрагментирован, что информация о блоках не поместится в одну запись MFT, используются две или более записей MFT.

Таким образом, для хранения больших фрагментированных файлов может быть использовано несколько записей таблицы MFT.

Проблема может возникнуть, если потребуется так много записей MFT, что в базовой записи не поместятся все индексы MFT.

Эта проблема решается следующим образом: **список записей MFT делается нерезидентным (то есть хранится отдельно на диске, а не в базовой записи MFT). В этом случае его размер уже ничем не ограничен.**

Помимо обычных файлов и каталогов, файловая система NTFS поддерживает жесткие связи, подобные используемых в UNIX.

2.7 *Шифрование файлов*

В операционной системе Windows проблемы несанкционированного доступа к данным решаются при помощи возможности шифрования файлов. В результате применения шифрования, даже если компьютер будет украден или перезагружен в системе MS-DOS, файлы останутся нечитаемыми.

Чтобы использовать шифрование в операционной системе Windows, нужно пометить каталог как зашифрованный, в результате чего будут зашифрованы все файлы в этом каталоге, а все новые файлы, перемещенные в этот каталог или созданные в нем, также будут зашифрованы.

Само шифрование и дешифрование выполняется не файловой системой NTFS, а специальным драйвером **EFS** (Encrypting File System — шифрующая файловая система), размещающимся **между NTFS и пользовательским процессом**. Таким образом, прикладная программа не знает о шифровании, а сама файловая система NTFS только частично вовлечена в этот процесс.

Работа системы шифрования файлов в операционной системе Windows:

Когда пользователь сообщает системе, что хочет зашифровать определенный файл, формируется случайный 128-разрядный ключ.

Ключ используется для поблочного шифрования файла с помощью симметричного алгоритма, параметром в котором используется этот ключ.

Каждый новый шифруемый файл получает новый случайный 128-разрядный ключ, так что никакие два файла не используют один и тот же ключ шифрования, что увеличивает защиту данных в случае, если какой-либо из ключей окажется скомпрометированным.

Применяемый в настоящий момент алгоритм шифрования представляет собой вариант стандартного алгоритма **DES** (Data Encryption Standard — стандарт шифрования данных), но архитектура EFS поддерживает добавление новых алгоритмов в будущем. Независимое шифрование каждого блока файла необходимо для сохранения возможности произвольного доступа к блокам файла.

Чтобы файл мог быть впоследствии расшифрован, ключ файла должен где-то храниться. Если бы ключ хранился на диске в открытом виде, тогда злоумышленник, укравший файлы, мог бы легко найти его и воспользоваться им для расшифровки украденных файлов. В этом случае сама идея шифрования файлов оказалась бы бессмысленной. Поэтому ключи файлов сами должны храниться на диске в зашифрованном виде. Для этого используется шифрование с открытым ключом.

После того как файл зашифрован, система с помощью информации в системном реестре ищет расположение открытого ключа пользователя. Открытый ключ можно без каких-либо опасений хранить прямо в реестре, так как по

открытому ключу невозможно определить закрытый ключ, необходимый для расшифровки файлов. Затем случайный 128-разрядный ключ файла шифруется открытым ключом, а результат сохраняется на диске вместе с файло.

Чтобы расшифровать файл, с диска считывается зашифрованный случайный 128-разрядный ключ файла. Однако для его расшифровки необходим закрытый ключ. В идеале этот ключ должен храниться на смарт-карте, вне компьютера, и вставляться в считывающее устройство только тогда, когда требуется расшифровать файл. Хотя операционная система Windows поддерживает смарт-карты, она не позволяет хранить на них закрытые ключи.

Вместо этого, когда пользователь в первый раз зашифровывает файл с помощью системы EFS, операционная система Windows формирует пару ключей (закрытый ключ, открытый ключ) и сохраняет закрытый ключ, зашифрованный при помощи симметричного алгоритма шифрования, на диске. Ключ для этого симметричного алгоритма формируется либо из пароля пользователя для регистрации в системе, либо из ключа, хранящегося на смарт-карте, если регистрация при помощи смарт-карты разрешена.

Таким образом, система EFS может расшифровать закрытый ключ во время регистрации пользователя в системе и хранить его в своем виртуальном адресном пространстве во время работы, чтобы иметь возможность расшифровывать 128-разрядные ключи файлов без дополнительного обращения к диску. Когда компьютер выключается, закрытый ключ стирается из виртуального адресного пространства системы EFS, так что никто, даже украв компьютер, не получит доступа к закрытому ключу.

Сложности возникают тогда, когда нескольким пользователям требуется доступ к одному и тому же зашифрованному файлу. В настоящий момент совместное использование зашифрованных файлов несколькими пользователями не поддерживается. Однако в будущем архитектура EFS может поддержать совместное использование, если ключ шифрования файла будет зашифровываться несколько раз, отдельно для каждого авторизованного пользователя,

его закрытым ключом. Все зашифрованные версии ключа могут добавляться к файлу.

Потенциальная потребность в совместном использовании зашифрованных файлов является одной из причин, по которой применяется такая двухуровневая система ключей. Если бы все файлы зашифровывались ключами владельцев файлов, то совместное использование зашифрованных файлов было бы невозможным. Эта проблема может быть решена, если для зашифровки каждого файла использовать отдельный ключ.

Схема с использованием случайных ключей для шифрования файлов, но с шифрованием самих ключей при помощи симметричного алгоритма шифрования не будет работать. Проблема в том, что наличие симметричного ключа, хранящегося на диске в открытом виде, разрушит всю систему защиты — сформировать ключ дешифрации по ключу шифрования слишком легко. Таким образом, медленное шифрование с открытым ключом требуется для шифрования ключей файлов.

Поскольку ключи шифрования все равно являются открытыми, хранение их в открытом виде не представляет опасности.

Вторая причина использования двухуровневой системы ключей заключается в производительности. Использование криптографии с открытым ключом для шифрования файлов было бы слишком медленным. Для повышения эффективности шифрование с открытым ключом применяется лишь для зашифровки коротких 128-разрядных ключей файлов, тогда как для шифрования самих файлов используется симметричный алгоритм.

3 *Организация дисковой системы Windows.*

3.1 *Типы дисков*

Можно использовать диски двух разных типов: **базовые** и **динамические.**

Дисковая система Windows поддерживает работу с любыми комбинациями дисков двух категорий (**базовые диски (Basic disks), динамические диски (Dynamic disks).**

3.2 *Базовый формат дисков*

Базовый формат основан на использовании **таблиц разделов**(partitions).

Диск, обладающий базовым форматом, называют базовым **диском (basic disk).**

Таким образом, базовый диск - это физический диск, содержащий

первичные разделы (primary partitions),

расширенные разделы (extended partitions)

и логические диски (logical drives).

В операционной системе Windows базовый формат является форматом по умолчанию

3.3 *Динамический формат дисков*

Динамический формат основан на понятии динамического диска **(dynamic disk)** и динамического тома **(dynamic volumes)**.

Динамический диск (dynamic disk) - это физический диск, который содержит только **динамические тома.**

Том(volumes) - это участок дискового пространства, который может включать в себя дисковое пространство, принадлежащие одному или нескольким разным физическим дискам.

Можно разделить физический жесткий диск на несколько томов, или создать том, включающий в себя несколько физических жестких дисков.

Каждый том форматируется в соответствии со стандартом определенной файловой системы, и ему можно поставить в соответствие латинскую букву, обозначающую логический диск.

Том, расположенный на динамическом диске (дисках), может принадлежать к одной из следующих разновидностей:

Simple (простой). При создании такого тома используется дисковое пространство, принадлежащее одному физическому диску. Это может быть одна область или несколько связанных между собой областей. Простой том можно расширить за счет дискового пространства, принадлежащего тому же самому диску, или за счет дискового пространства, принадлежащего другим физическим дискам, однако в последнем случае простой том становится простирающимся (spanned) томом.

Spanned (составной). В состав такого тома входит дисковое пространство, расположенное на разных физических дисках (допускается использовать до 32 дисков). Простирающийся том может быть расширен за счет дискового пространства, принадлежащего дополнительным дискам, однако его нельзя сделать отраженным (mirrored).

Mirrored (зеркальный) или RAID-1(disk mirroring). Данные, хранящиеся на таком томе, автоматически дублируются на двух разных физических дисках. Таким образом повышается уровень отказоустойчивости тома. Все данные, записываемые на отраженный том, автоматически копируются на два разных физических диска. Если один из дисков выходит из строя, пользователь сможет обратиться к данным, расположенным на другом диске. Отраженный том нельзя расширить.

Striped (чередующийся) или RAID-0. Данные, хранящиеся на таком томе, равномерно распределяются между несколькими физическими дисками таким образом, чтобы повысить производительность. Таким образом, операционная система может выполнять одновременно несколько операций обращения к тому, например, читать данные с одного из дисков и одновременно с этим записывать данные на другой или выполнять одновременно несколько операций записи. Полосатый том нельзя расширить или сделать отраженным (mirrored), кроме того, он не обеспечивает отказоустойчивости.

RAID-5 или Striped with parity (чередующийся с четностью). Данные, хранящиеся на таком томе, равномерно распределены между несколькими физическими дисками, однако в отличие от полосатого тома том RAID-

5 предусматривает хранение в дисковом массиве специальной служебной информации, благодаря которой, в случае если один из дисков массива выходит из строя, хранившиеся на нем данные можно восстановить. Эту информацию называют информацией о четности (parity information). Том RAID-5 нельзя сделать отраженным (mirrored) или расширить.

Динамический формат обладает более широкими возможностями и поддерживает множество дополнительных механизмов.

Использование динамического формата на системах, оснащенных единственным жестким диском, редко когда бывает оправданным. Преобразовав единственный базовый диск компьютера в динамический, всего лишь получают возможность создать на нем более трех первичных и одного логического разделов или четырех первичных разделов (такие ограничения свойственны базовым дискам). **Нельзя воспользоваться всеми остальными преимуществами динамического формата, если компьютер оснащен только одним жестким диском.**

Для хранения пользовательских данных рекомендуется использовать динамические диски, так как именно динамический формат позволяет с легкостью увеличивать размер томов.

Динамическая долговременная память также будет предпочтительней, в случае, если необходимо использовать многодисковые системы хранения данных, такие как составные, чередующиеся и зеркальные тома, или тома RAID-5.

Формат диска (базовый или динамический) никак не связан с форматом файловой системы. Как разделы базовых дисков, так и тома динамических дисков могут содержать на себе любую из поддерживаемых Windows файловых систем.