Файл. Файловая система. Типы файлов. Имена файлов. Атрибуты файлов. Свойства файловой системы NTFS. Компоненты NTFS и их взаимодействие с исполнительной системой. Структуры данных NTFS. MFT. Резидентные и нерезидентные атрибуты.

Одной из основных задач ОС является предоставление удобств пользователю при работе с данными, хранящимися на дисках. Для этого ОС подменяет физическую структуру хранящихся данных некоторой удобной для пользователя логической моделью. Базовым элементом этой модели является файл, который так же, как и файловая система в целом, может характеризоваться как логической, так и физической структурой.

Файл– это именованная область внешней памяти, в которую можно записывать и из которой можно считывать данные. Файлы обычно хранятся в памяти, не зависящей от энергопитания.

Основные цели использования файлов следующие:

- Долговременное и надежное хранение информации. Долговременность достигается за счет использования запоминающих устройств, не зависящих от питания, а высокая надежность определяется средствами защиты доступа к файлам и общей организацией программного кода ОС, при которой сбои аппаратуры чаще всего не разрушают информацию, хранящуюся в файлах.
- Совместное использование информации. Файлы обеспечивают естественный и легкий способ разделения информации между приложениями и пользователями за счет наличия понятного человеку символьного имени и постоянства хранимой информации и расположения файла. Пользователь должен иметь удобные средства работы с файлами, включая каталоги-справочники, объединяющие файлы в группы, средства поиска файлов по признакам, набор команд для создания, модификации и удаления файлов. Файл может быть создан одним пользователем, а затем использоваться другим пользователем. При этом создатель файла или администратор могут определить права доступа других пользователей к файлу. Эти цели реализуются в ОС файловой системой.

Файловая система– это часть операционной системы, включающая:

- совокупность всех файлов на диске;
- наборы структур данных, используемых для управления файлами, такие, например, как каталоги файлов, дескрипторы файлов, таблицы распределения свободного и занятого пространства на диске.
- комплекс системных программных средств, реализующих различные операции над файлами, такие как создание, уничтожение, чтение, запись, именование и поиск файлов.

Файловая система позволяет программам обходиться набором достаточно простых операций для выполнения действий над некоторым абстрактным объектом, представляющим файл. При этом программистам не нужно иметь дело с деталями действительного расположения данных на диске, буферизацией данных и другими низкоуровневыми проблемами передачи данных с долговременного запоминающего устройства. Все эти функции файловая система берет на себя. Файловая система распределяет дисковую память, поддерживает именование файлов, отображает имена

файлов в соответствующие адреса во внешней памяти, обеспечивает доступ к данным, поддерживает разделение, защиту и восстановление файлов.

Таким образом, файловая система играет роль промежуточного слоя, экранирующего все сложности физической организации долговременного хранилища данных, и создающего для программ более простую логическую модель этого хранилища, а также предоставляя им набор удобных в использовании команд для манипулирования файлами.

Файловые системы поддерживают несколько функционально различных типов файлов, в число которых, как правило, входят обычные файлы, файлы-каталоги, специальные файлы и другие.

Обычные файлы, или просто *файлы*, содержат информацию произвольного характера, которую заносит в них пользователь или которая образуется в результате системных и пользовательских программ. Большинство операционных систем никак не ограничивает и не контролирует содержимое и структуру обычного файла. Содержание обычного файла определяется приложением, которое с ним работает. Все операционные системы должны уметь распознавать хотя бы один тип файлов – их собственные исполняемые файлы.

Каталоги– это особый тип файлов, которые содержат системную справочную информацию о наборе файлов, сгруппированных пользователями по какому-либо неформальному признаку. Во многих операционных системах в каталог могут входить файлы любых типов, в том числе другие каталоги, за счет чего образуется древовидная структура, удобная для поиска. Каталоги устанавливают соответствие между именами файлов и их характеристиками, используемыми файловой системой для управления файлами. В число таких характеристик входит, в частности, информация (или указатель на другую структуру, содержащую эти данные) о типе файла и расположении его на диске, правах доступа к файлу и датах его создания и модификации. Во всех остальных отношениях каталоги рассматриваются файловой системой как обычные файлы.

Специальные файлы– это фиктивные файлы, ассоциированные с устройствами ввода-вывода, которые используются для унификации механизма доступа к файлам и внешним устройствам. Специальные файлы позволяют пользователю выполнять операции ввода-вывода посредством обычных команд записи в файл или чтения из файла. Эти команды обрабатываются сначала программами операционной системы, а затем на некотором этапе выполнения запроса преобразуются операционной системой в команды управления соответствующим устройством.

Все типы файлов имеют символьные имена. В иерархически организованных файловых системах обычно используются три типа имен файлов: простые, составные и относительные.

Простое, или *короткое, символьное имя* идентифицирует файл в пределах одного каталога. Простые имена присваивают файлам пользователи и программисты, при этом они должны учитывать ограничения ОС как на номенклатуру символов, так и на длину символов. Например, файловая система FAT ограничивает длину имени 8+3 символами, FAT32 и NTFS – 255 символами.

В иерархических файловых системах разным файлам разрешено иметь одинаковые простые символьные имена при условии, что они принадлежат разным каталогам. Таким образом, работает схема *«много файлов – одно простое имя»*. Для однозначной идентификации файла в таких системах используется так называемое полное имя.

Полное имя представляет собой цепочку простых символьных имен всех каталогов, через которые проходит путь от корня до данного файла (*корень, root, или корневой каталог*– это каталог самого верхнего уровня в иерархической файловой системе). Таким образом, полное имя является *составным*, в котором простые имена отделены друг от друга принятым в ОС разделителем.

В древовидной файловой системе между файлом и его полным именем имеется взаимно однозначное соответствие *«один файл – одно полное имя»*. В файловых системах, имеющих сетевую структуру, файл может входить в несколько каталогов, и здесь справедливо соответствие *«один файл – много полных имен»*. В обоих случаях файл однозначно идентифицируется полным именем.

Файл может быть идентифицирован также относительным именем. *Относительное* имя файла определяется через понятие «текущий каталог». Для каждого пользователя в каждый момент времени один из каталогов файловой системы является текущим, причем этот каталог выбирается самим пользователем по команде ОС. Файловая система фиксирует имя текущего каталога, чтобы использовать его как дополнение к относительным именам для образования полного имени файла. При использовании относительных имен пользователь идентифицирует файл цепочкой имен каталогов, через которые проходит маршрут от текущего каталога до данного файла.

Понятие «файл» включает не только хранимые им данные и имя, но и атрибуты. *Атрибуты*– это информация, описывающая свойства файла. Примеры возможных атрибутов файла:

- тип файла (обычный файл, каталог, специальный файл и т.п.);
- владелец файла;
- создатель файла;
- пароль для доступа к файлу;
- информация о разрешенных операциях доступа к файлу;
- время создания, последнего доступа и последнего изменения;
- текущий размер файла;
- максимальный размер файла;
- признак «только для чтения»;
- признак «скрытый файл»;
- признак «системный файл»;
- признак «архивный файл»;
- признак «двоичный/символьный»;
- признак «временный»;
- признак блокировки;
- длина записи в файле;
- указатель на ключевое поле в записи;
- длина ключа.

Набор атрибутов файла определяется спецификой файловой системы: в файлаовых системах разного типа для характеристики файлов могут использоваться разные наборы атрибутов.

Значения атрибутов файлов могут непосредственно содержаться в каталогах, как это сделано в файловой системе FAT ОС MS-DOS (рис. 11.1, а). Другим вариантом является размещение атрибутов в специальных таблицах, а в каталогах содержатся только ссылки на эти таблицы. Такой подход реализован в файловой системе ufs ОС UNIX (рис. 11.1, б).

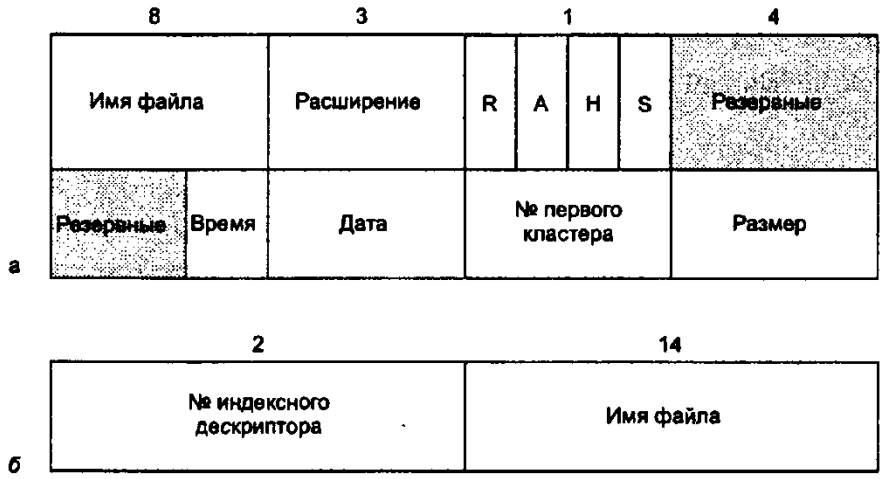

Рис 11.1. Структура каталогов: а – структура записи каталога файловой системы FAT, б- структура записи каталога файловой системы ufs.

В том и другом вариантах каталоги обеспечивают связь между именами файлов и собственно файлами. Однако подход, когда имя файла отделено от его атрибутов, делает систему более гибкой. Например, файл может быть легко включен сразу в несколько каталогов. Записи об этом файле в разных каталогах могут содержать разные простые имена, но в поле ссылки будет указан один и тот же номер индексного дескриптора (*индексный дескриптор*– это уникальный числовой идентификатор файла, предназначенный для использования операционной системой). Понятно, что файловая система FAT имеет древовидную структуру, а ufs – сетевую.

Определим некоторые термины:

- *Сектор*– аппаратно адресуемый блок носителя. Размер сектора зависит от типа носителя. Например, размер сектора для жесткого диска, как правило, равен 512 байтам, размер сектора CD-ROM – 2048 байт.
- *Кластер*– адресуемый блок, используемый многими файловыми системами. Размер кластера всегда кратен размеру сектора. Файловая система использует кластеры для более эффективного управления дисковым пространством.
- *Раздел (partition)*– набор непрерывных секторов на диске. Адрес начального сектора раздела, его размер и другие характеристики хранятся в таблице разделов или иной базе данных управления диском, которая размещается на том же диске, что и данный раздел.
- *Простой том (simple volume)*– объект, представляющий секторы одного раздела, которым драйверы файловых систем управляют как единым целым.
- *Составной том (multipartition volume)*– объект, представляющий секторы нескольких разделов, которыми драйверы файловых систем управляют как единым целым.
- *Метаданные*– данные, хранящиеся на томе и необходимые для поддержки управления файловой системой. Включают, в частности, информацию, определяющую местонахождение файлов и каталогов на томе. Как правило, они недоступны приложениям.

Дальнейшее рассмотрение работы файловых систем будем производить на примере Windows и файловой системы NTFS. Разработка NTFS велась с учетом требований, предъявляемых к файловой системе корпоративного класса:

- Чтобы свести к минимуму потери данных в случае неожиданного выхода системы из строя или ее краха, файловая система должна гарантировать целостность своих метаданных.
- Для защиты конфиденциальных данных от несанкционированного доступа файловая система должна быть построена на интегрированной модели защиты.
- Файловая система должна поддерживать защиту пользовательских данных за счет программной избыточности данных.

Рассмотрим, как эти возможности реализованы в NTFS.

Восстанавливаемость. NTFS обеспечивает восстановление файловой системы на основе концепции *атомарной транзакции (atomic transaction)*. Атомарные транзакции – это метод обработки изменений в базе данных, при котором сбои в работе системы не нарушают корректности или целостности базы данных. Суть атомарных транзакций заключается в том, что некоторые операции над базами данных, называемые транзакциями, выполняются по принципу «все или ничего». *Транзакцию* можно определить как операцию ввода-вывода, изменяющую данные файловой системы или структуру каталогов тома. Отдельные изменения на диске, составляющую транзакцию, выполняются атомарно: в ходе транзакции должны быть внесены все требуемые изменения. Если транзакция прервана аварией системы, часть изменений, уже внесенных к этому моменту, нужно отменить. Такая отмена называется откатом (roll back). После отката база данных возвращается в исходное согласованное состояние, в котором она была до начала транзакции. NTFS гарантирует, что транзакция будет либо полностью выполнена, либо отменена, если хотя бы одну из операций не удастся завершить из-за сбоя системы.

Защита. Файлы и каталоги в NTFS защищены от доступа пользователей, не имеющих соответствующих прав. Открытый файл реализуется в виде объекта Windows «файл» с дескриптором защиты, хранящимся на диске как часть файла. Прежде чем процесс сможет открыть описатель объекта «файл» система защиты Windows должна убедиться, что у этого процесса имеются соответствующие полномочия.

Избыточность данных и отказоустойчивость. Избыточность данных для пользовательских файлов реализуется через многоуровневую модель драйверов, которая поддерживает отказоустойчивые диски. При записи данных диск NTFS взаимодействует с диспетчером томов, а тот – с драйвером жесткого диска. Диспетчер томов может зеркалировать данные одного диска на другом и таким образом позволяет при необходимости использовать данные с избыточной копии (RAID-1). Диспетчеры томов также могут записывать данные в чередующиеся области (stripes) на три и более дисков, используя один диск для хранения информации о четности (RAID-5).

Кроме того, NTFS обладает рядом дополнительных возможностей.

Множественные потоки данных.В NTFS каждая единица информации, сопоставленная с файлом (имя и владелец файла, метка времени и т.д.), реализована в виде атрибута файла (атрибута объекта NTFS). Каждый атрибут состоит из одного потока данных, т.е. из простой последовательности байтов. Это позволяет легко добавить к файлу новые атрибуты (и соответственно новые потоки). Поскольку данные являются всего лишь одним из атрибутов файла и поскольку можно добавлять новые атрибуты, файлы и каталоги NTFS могут содержать несколько потоков данных.

В любом файле NTFS по умолчанию имеется один безымянный поток данных. Приложения могут создавать дополнительные, именованные потоки данных и обращаться к ним по именам. Чтобы не изменять Windows-функции API ввода-вывода, которым имя файла передается как строковый аргумент, имена потоков данных задаются через двоеточие после имени файла, например:

myfile.dat:stream2

Каждый поток имеет свой выделенный размер (объем зарезервированного для него дискового пространства), реальный размер (использованное число байтов)и длину действительных данных (инициализированная часть потока). Кроме того, каждому потоку предоставляется отдельная файловая блокировка, позволяющая блокировать диапазоны байтов и поддерживать параллельный доступ.

Имена на основе Unocode. Как и Windows в целом, NTFS полностью поддерживает Unocode, используя 16-битную кодировку символов для хранения имен файлов, каталогов и томов. Длина имени каждого каталога или файла в пути может достигать 255 символов.

Динамическое переназначение плохих кластеров. Обычно, если программа пытается считать данные из плохого сектора диска, операция чтения заканчивается неудачей, а данные соответствующего кластера становятся недоступными. Но если диск отформатирован как отказоустойчивый том NTFS, специальный драйвер Windows динамически считывает «хорошую» копию данных, хранившихся в плохом секторе, и посылает предупреждение NTFS предупреждение о плохом кластере. NTFS выделяет новый кластер, заменяющий тот, в котором находится плохой сектор, и копирует данные в этот кластер. Плохой кластер помечается как аварийный и больше не используется.

Сжатие и разреженные файлы. NTFS поддерживает сжатие файловых данных. NTFS выполняет процедуры сжатия и декомпрессии прозрачно, незаметно для пользователей и приложений. Каталог также может быть сжат. Это влечет за собой сжатие файлов, которые будут впоследствии созданы в этом каталоге.

Второй тип сжатия известен как разреженные файлы (sparse files). Если файл помечен как разреженный, NTFS не выделяет на томе место для тех частей файла, которые определены приложением как пустые. При чтении приложением пустых областей разреженного файла NTFS просто возвращает буферы, заполненные нулевыми значениями.

Протоколирование изменений. Приложениям многих типов нужно отслеживать изменения файлов и каталогов тома. Очевидный способ мониторинга изменений тома – его сканирование с записью состояния файлов и каталогов и анализ отличий при следующем сканировании. Альтернативный подход заключается в том, чтобы зарегистрироваться на получение уведомлений об изменении содержимого каталогов. NTFS предусматривает третий подход, сочетающий достоинства обоих подходов. Приложение может настроить журнал изменений, и тогда NTFS будет регистрировать информацию об изменениях файлов и каталогов во внутреннем файле -*журнале изменений (change journal)*.

Шифрование. Корпоративные пользователи часто хранят на своих компьютерах конфиденциальную информацию .NTFS поддерживает механизм Encrypting File System (EFS), с помощью которого пользователи могут шифровать конфиденциальные данные. EFS, как и механизм сжатия, полностью прозрачен для приложения. Это означает, что данные автоматически расшифровываются при чтении их приложением, работающим под

учетной записью пользователя, который имеет права на просмотр этих данных, и автоматически шифруются при изменении их авторизированным приложением.

Доступ к зашифрованным файлам можно получить только с помощью закрытого ключа из криптографической пары EFS (которая состоит из открытого и закрытого ключей), а закрытые ключи защищены паролем учетной записи. Таким образом. без пароля учетной записи доступ к зашифрованным EFS файлам нельзя получить никакими средствами.

Рассмотрим, как драйвер NTFS взаимодействует с другими частями исполнительной системы (рис. 11.2).

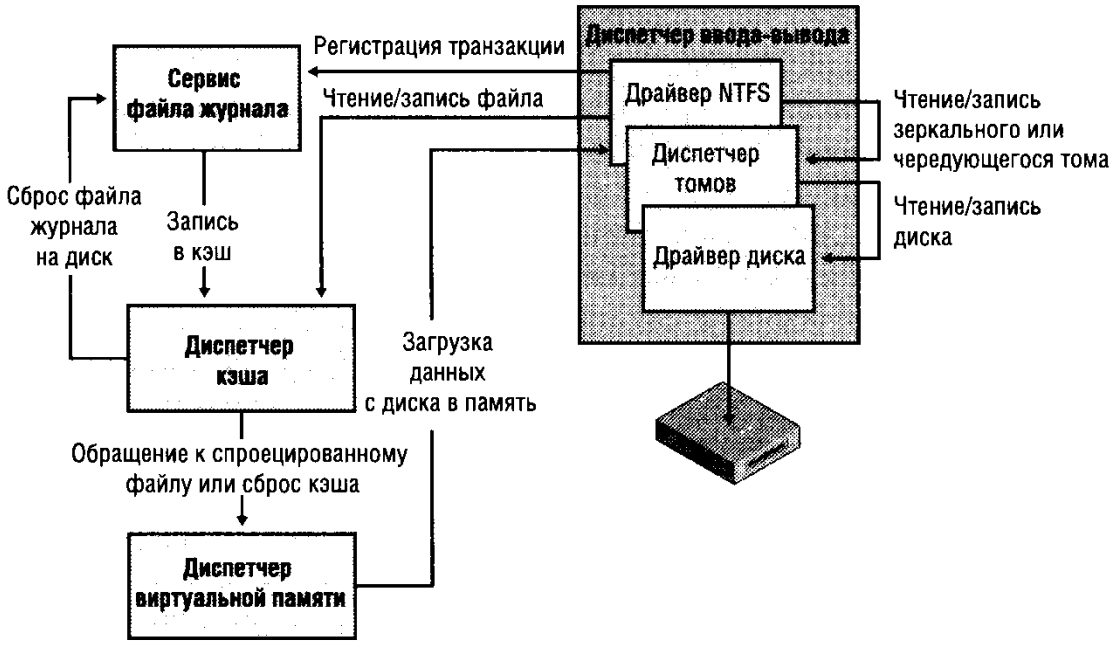

Рис. 11.2. NTFSи связанные с ней компоненты.

Сервис файла журнала (log file service, LFS) является частью NTFS и предоставляет функции для поддержки журнала изменений на диске. Файл журнала LFS используется при восстановлении тома NTFS в случае аварии системы. *Диспетчер кэша*– компонент исполнительной системы, предоставляющий общесистемные сервисы кэширования для NTFS и драйверов других файловых систем. Когда программа пытается обратиться к какой-либо части файла, не загруженной в кэш (промах кэша), диспетчер памяти вызывает NTFS для обращения к драйверу диска и загрузки нужных файловых данных. Диспетчер кэша оптимизирует дисковый ввод-вывод, вызывая диспетчер памяти из потоков подсистемы отложенной записи.

Приложение создает файлы и обращается к ним так же, как и к любым другим объектам Windows – через описатели (дескрипторы) объектов. На рисунке 11.3 показаны структуры данных, связывающие описатель файла со структурой файловой системы на диске.

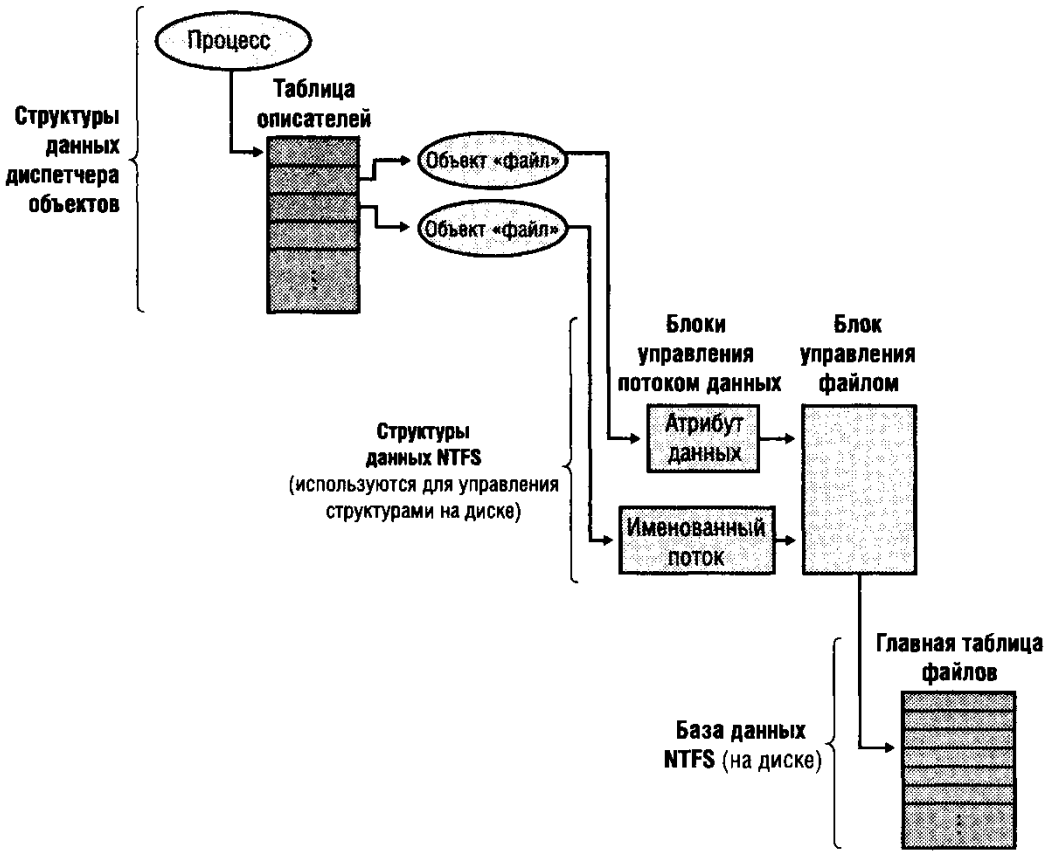

Рис. 11.3. Структуры данных NTFS.

NTFS получает адрес файла на диске из объекта «файл» по нескольким указателям. Как видно из рисунка, объект «файл», предоставляющий один вызов системного сервиса для открытия файла, указывает на *блок управления потоком данных* (stream control block, SCB) для атрибута, который вызывающая программа пытается считать или записать. В нашем случае процесс открыл как безымянный атрибут данных, так и именованный поток (альтернативный атрибут данных) файла. SCB предоставляет отдельные атрибуты файла и содержит информацию о том, как искать конкретные атрибуты внутри файла. Все SCB файла указывают на общую структуру данных *–блок управления файлом* (file control block, FCB).FCB содержит указатель на запись файла в *главной таблице файлов* (master files table,MFT).

Рассмотрим структуру тома NTFS. В NTFS все данные, хранящиеся на томе, содержатся в файлах. Это относится и к структурам данных, используемым для поиска и выборки файлов, к начальному загрузочному коду и битовой карте, в которой регистрируется состояние пространства всего тома (метаданные NTFS). Главная таблица файлов (MFT) занимает центральное место в структуре тома. MFT реализована как массив записей о файлах. Размер каждой записи фиксирован и равен 1 Кб. Логически MFT содержит по одной строке на каждый файл тома, включая строку для самой MFT. Кроме MFT, на каждом томе NTFS имеется набор файлов метаданных с информацией, необходимой для реализации структуры файловой системы. Остальные файлы NTFS-тома являются обычными файлами и каталогами.

Обычно каждая запись MFT соответствует отдельному файлу. Но если у файла много атрибутов или он сильно фрагментирован, для него может понадобиться более одной записи. Тогда первая запись MFT, хранящая адреса других записей, называется *базовой* (base file record).

Файл на томе NTFS идентифицируется 64-битным значением, которое называется файловой ссылкой (file reference). Файловая ссылка состоит из номера файла и номера последовательности. Номер файла равен позиции его записи в MFT минус 1. Номер последовательности в файловой ссылке увеличивается каждый раз на 1 при каждом повторном использовании позиции записи в MFT, что позволяет NTFS проверять внутреннюю целостность файловой системы. Структура файловой ссылки показана на рис. 11.4.

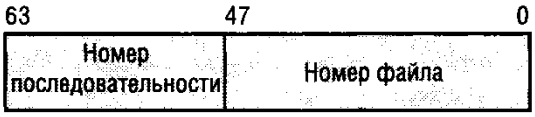

Рис. 11.4. Файловая ссылка.

Как уже было сказано, NTFS рассматривает файл не просто как хранилище текстовых или двоичных данных, а как совокупность пар атрибутов и их значений, одна из которых содержит данные файла. Другие атрибуты включают имя файла, стандартную информацию (метки времени создания, модификации, атрибуты «только для чтения», «архивный» и др.), и, возможно, дополнительные именованные атрибуты данных. Каждый атрибут файла хранится в файле как отдельный поток байтов. Строго говоря, NTFS читает и записывает не файлы, а потоки атрибутов. NTFS поддерживает следующие операции над атрибутами: создание, удаление, чтение и запись.

Если файл невелик, то все его атрибуты и их значения (например, файловые данные) умещаются в одной записи файла. Когда значение атрибута хранится непосредственно в MFT, атрибут называется *резидентным*. Некоторые атрибуты всегда резидентны – по ним NTFS находит нерезидентные атрибуты. Так, атрибуты «стандартная информация» и «корень индекса» всегда резидентные.

Каждый атрибут начинается со стандартного заголовка, в котором содержится информация об атрибуте, используемая NTFS для базового управления атрибутами. В случае резидентных атрибутов заголовок также содержит смещение значения атрибута от начала заголовка и длину этого значения (рис. 11.5).

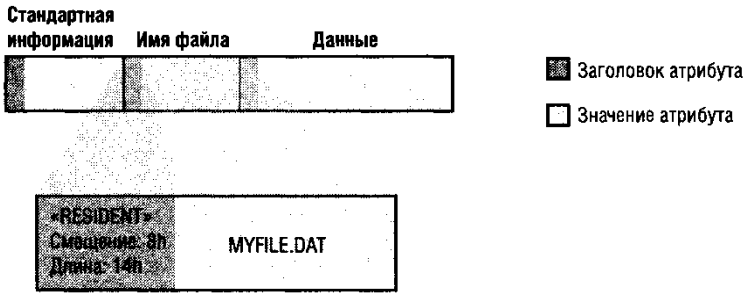

Рис. 11.5. Заголовок и значение резидентного атрибута.

Когда значение атрибута хранится непосредственно в MFT, обращение к нему занимает значительно меньше времени. Вместо того, чтобы искать файл в таблице, а затем считывать цепочку кластеров для поиска файловых данных, NTFS обращается к диску только один раз и немедленно считывает данные. Таким образом, небольшие файлы могут полностью храниться в MFT.

Если некоторый атрибут, например, файловые данные, слишком велик и не может уместиться в записи MFT, NTFSвыделяет для него отдельные кластеры за пределами MFT. Эта область называется*группой*(run) или*экстентом*(extent). Если размер значения атрибута впоследствии расширяется, NTFS выделяет для новых данных еще одну группу. Атрибуты, значения которых хранятся в группах, называются*нерезидентными*.

В случае нерезидентного атрибута (им может быть атрибут данных большого файла) в его заголовке содержится информация, нужная NTFS для поиска значения атрибута на диске. На рисунке 11.6 показан нерезидентный атрибут данных, хранящийся в двух группах.

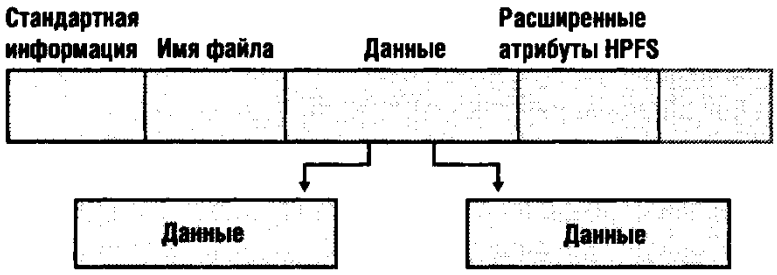

Рис. 11.6. Запись MFT для большого файла с двумя группами данных

NTFSадресуется к конкретным местам на диске, используя*логические номера кластеров*(logicalclusternumbers,LCN). Для этого все кластеры на томе нумеруются по порядку. Для преобразования LCN в физический адрес на диске NTFS умножает LCN на кластерный множитель (размер кластера на томе) и получает байтовое смещение от начала тома, воспринимаемое интерфейсом драйвера диска. На данные внутри файла NTFS ссылается по*виртуальным номерам кластеров*(virtual cluster numbers, VCN), нумеруя кластеры, которые принадлежат конкретному файлу.VCNне обязательно должны быть физически непрерывными. Поэтому для каждой группы в записиMFTсодержится сопоставление VCN-LCN для всех групп, что позволяет NTFS легко находить выделенные под них области на диске (рис. 11.7).

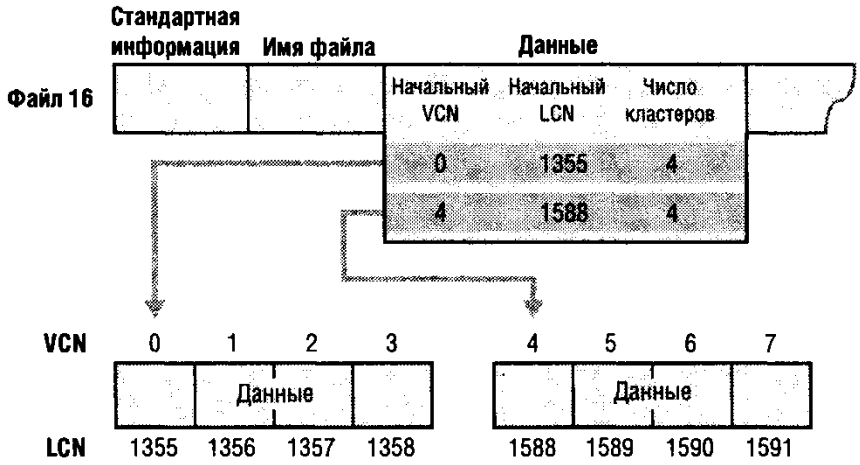

Рис. 11.7. Сопоставление VCN-LCN для нерезидентного атрибута данных