**Лекция №2 файловые системы**

***Таблица размещения файлов (FAT).***

Весь логичнийдиск разбивается операционной системой на части

одинакового размера, которые называются кластерами.

Кластер может содержать несколько секторов. Для каждого кластера FAT имеет свой индивидуальный элемент, в котором хранится информация об использовании данного кластера.

Размер FAT определяется общим количеством кластеров на логическом диске. Если количество кластеров на диске <4085, то используется FAT12, если 4085 <кластеров <655525-FAT16, дальше FAT32.

Названия типов FAT происходят от размера элемента: для FAT12 каждый элемент имеет размер 12 бит (1,5 байта), FAT16 - 16 бит (2б), FAT32 - 32бит (4б). В FAT32 четыре старших двоичных разряда зарезервированы и игнорируются в процессе работы операционной системы. Значимы только семь младших шестнадцатеричных разрядов элемента. Каждому файлу, который содержится в области данных диска, соответствует цепочка элементов FAT - упорядоченный однонаправленный список. Он обеспечивает передвижение только вперед. Если нужно вернуться к предыдущему кластера, то нужно снова провести поиск с самого начала списка.

Первые 2 элемента FAT - резервные. В первом резервном элементе FAT сохраняется сигнатура. Для FAT12 это FF8h, FAT16 - FFF8h, FAT32 - FFFFFF8h.

* втором резервном элементе при форматировании диска записывается признак конца файла (FFFh, FFFFh, FFFFFFFh для Microsoft). Другие фирмы могут использовать другие значения. Кроме того, системы FAT16 и FAT32 могут использовать 2 старших разряда этого элемента в качестве флагов. Флаг ClnShutBitMask занимает в системе FAT16 двоичный разряд 15 в FAT32 - разряд 27. Если он равен единице, то логический диск чистый (clear), если нулю, то "грязный" (dirty).

Термин "грязный" означает, что работа с диском не была закончена правильно и при загрузке системы должна быть выполнена процедура восстановления диска. Другой флаг HrdErrBitMask - служит признаком присутствия сбоев при выполнении

операции ввода / вывода. В системе FAT16 он содержится в двоичном разряде 14 в FAT32 - в разряде 26.

При загрузке ОС он устанавливается в 1, а если возник сбой при записи или чтении информации, то он сбрасывается в 0.

Табл. 1.8 Значение элементов таблицы размещения файлов FAT

|  |  |  |  |
| --- | --- | --- | --- |
| FAT 12 | FAT 16 | FAT 32 | содержание |
|  |  |  |  |
| 0 | 0 | 0 | свободный кластер |
|  |  |  |  |
| FF7 h | FFF7 h | FFFFFF7 h | дефектный кластер |
|  |  |  |  |
| 003 | 0003 | 0000003 | № следующего кластера диска |
|  |  |  |  |

Общая схема использования FAT.

1. Читаем в память FAT полностью.

Обычно FAT располагается сразу после ВООТ-сектора (логический сектор с №1). Для точного определения начального сектора FAT нужно прочитать в память ВООТ-сектор и проанализировать удерживаемый ВРВ. В нем есть поля, где записано количество зарезервированных секторов, которые располагаются перед FAT и размер FAT в секторах. Кроме того, на диске может находиться несколько копий FAT. OC использует только первую копию, другие нужны только для работы утилит восстановления диска.

1. Определяем номер 1-го кластера файла, для которого необходимо определить его расположение на диске.
2. Используем номер первого кластера как индекс в FAT для определения номера следующего кластера.
3. Повторяем предыдущую процедуру до тех пор, пока изъято из FAT значение не будет соответствовать концу файла.

Процедура определения номера кластера с FAT зависит от формата таблицы размещения файлов.

16-битную FAT можем представить как массив 16-битных чисел. Для определения номера следующего кластера нужно изъять 16-битное значение с FAT, используя в качестве индекса номер предыдущего кластера. Для 12-битной FAT процедура определения следующего кластера выглядит так:

1. Умножить номер начального кластера на 3.

1. Разделить результат на 2, так как каждый элемент таблицы имеет длину 1,5

б.

1. Прочитать 16-битное слово с FAT, используя в качестве смещения значение, полученное после деления на 2.
2. Если номер начального кластера четный, то на выбранное из FAT слово надо наложить маску 0fffh, оставив младше 12 битов. Если номер начального кластера нечетный, то выбранное из FAT значение нужно сдвинуть вправо на 4 бита, оставив 12 старших битов.
3. Полученный результат является номером следующего кластера в цепочке.

Каталоги файлов.

Каталог файлов - это массив 32 байтных элементов описателей файлов.

* точки зрения ОС все каталоги - это файлы и могут содержать произвольное количество записей. Исключение - корневой каталог в системах FAT12 и FAT16. Корневой каталог (ROOT DIRECTORY) - это главный каталог диска, с которого начинается дерево подкаталогов.

Для корневого каталога в FAT12 и FAT16 выделено в системной области логического диска специальное место фиксированного размера 18 Кб, которое рассчитано на хранение 512 элементов. В FAT32 корневой каталог - произвольного размера.

Табл. 1.9 Структура корневого каталога

|  |  |  |  |
| --- | --- | --- | --- |
| смещение | Размер | описание |  |
|  |  |
|  |  |  |  |
|  |  | Короткое имя файла или каталога, выровнено слева и |  |
| 0 | 8 | дополненное промежутками. |  |
|  |  |  |
|  |  |  |  |
|  |  | Расширение имени файла, выровнено слева и дополненное |  |
| 8 | 3 | промежутками |  |
|  |  |  |
|  |  |  |  |
| 0Bh | 1 | атрибуты файла |  |
|  |  |
|  |  |  |  |
| 1Ch | 1 | Зарезервировано для Windows NT (= 0) |  |
|  |  |
|  |  |  |  |
|  |  | Уточнение времени создания файла |  |
| 1Dh | 1 | (Содержит десятки миллисекунд 0..199) |  |
|  |  |  |
|  |  |  |  |
| 0Eh | 2 | Время создания файла или время его последней модификации. |  |
|  |  |
|  |  |  |  |
| 10h | 2 | Дата создания файла или время его последней модификации. |  |
|  |  |
|  |  |  |  |
| 12h | 2 | Дата последнего обращения к файлу для записи или чтения |  |
|  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| смещение | Размер | описание |  |
|  |  |
|  |  |  |  |
|  |  | Старше слово номера первого кластера, распределенного |  |
| 14h | 2 | файла. |  |
|  |  |  |
|  |  |  |  |
| 16h | 2 | Время выполнения последней операции записи в файл. |  |
|  |  |
|  |  |  |  |
| 18h | 2 | Дата выполнения последней операции записи в файл. |  |
|  |  |
|  |  |  |  |
| 1Ah | 2 | Младшее слово номера первого кластера файла. |  |
|  |  |
|  |  |  |  |
| 1Ch | 4 | Размер файла в байтах. |  |
|  |  |
|  |  |  |  |

Байт атрибутов является принадлежностью каждого файла. Поля от 0Dh к 14h обрабатываются только в FAT32. Биты этого байта имеют следующее значение:

* 00h файл предназначен только для чтения;
* 01h скрытый файл (его нет в списке)
* 02h системный файл. Этот бит обычно установлен в файлах, которые выступают частью ОС;
* 07h отметка диске. Для этого значения поля имени файла и расширения должны рассматриваться как одно поле длиной 11 байт. Это поле и содержит отметку диске;
* 14h подкаталог данного каталога;
* 05h архивный файл.

Формат поля времени создания файла.

Биты 04 ... 00 - число двухсекундных интервалов в двоичной форме (от 0 до 29, то есть от 0 до 58 сек.)

Биты 10 ... 05 - число минут в двоичной форме (0-59).

1. Первый байт короткого имени выполняет функции признаки занятости элемента каталога. Если он равен Е5h, то элемент каталога свободный; 00h - то элемент

каталога свободный и является началом чистой области каталога; 05h, то в этом байте

содержится ASCII символ с кодом 0E5h. В коротком имени нельзя использовать символы с кодами <20h (исключение - 05h).

1. Нельзя использовать символы с кодами 22h, 2Ah, 2Bh, 2Ch, 2Eh, 2Fh, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh, 5Bh, 5Ch, 5Dh, 7Ch.
2. Нельзя использовать символ промежутке (20h) в первом байте.

Признак того, что свободный элемент каталога используется для хранения участки долгого имени файла - заполнение единицами разрядов с 0 по 3 байта атрибутов.

Длинное имя можно создавать начиная с ОС Windows 95. Для его хранения используют свободные элементы каталога, смежные с основным элементом - описателем файла.

Оно записывается в Unicode, где каждой национальной азбуке соответствует свой набор кодов. К свободным элементов каталога длинное имя записывается в разрезанном на кусочки виде. В одном элементе каталога можно хранить фрагмент до 13 символов Unicode. Неиспользованная часть последнего фрагмента заполняется кодами FFFFh.

Табл. 1.10 Структура элемента каталога с длинным именем файла

|  |  |  |
| --- | --- | --- |
| смещение | Размер | описание |
|  |  |  |
| 0h | 1 | номер фрагмента |
|  |  |  |
| 01h | 10 | Первая часть фрагмента имени файла |
|  |  |  |
| 0Bh | 1 | атрибуты файла |
|  |  |  |
| 0Ch | 1 | байт флагов |
|  |  |  |
| 0Dh | 1 | Контрольная сумма короткого имени |
|  |  |  |
| 0Eh | 12 | Вторая часть фрагмента имени |
| 1Ah | 2 | Номер первого кластера (должен быть 0) |
| 1Ch | 4 | Третья часть фрагмента имени |

Длинное имя записывается в каталог первым. Фрагменты расположены в обратном порядке. Далее содержится описатель файла, который сохраняет сокращенный вариант этого имени.

начало каталога

Последняя часть длинного имени

....................................

Первая часть длинного имени

описатель файла

....................................

конец каталога

Рис. 1.3 Описатель файла

***Файловая система NTFS.***

Для NTFS вся информация томе представляет собой файл или частью файла. Каждый разделен на томе NTFS сектор принадлежит некоторому файлу.

Диск NTFS условно делится на две части. Рассмотрим общую структуру NTFS. Первые 12% диска отводятся под так называемую MFT зону - пространство, в которое растет метафайл MFT.

Запись каких-либо данных в эту область невозможна. MFT-зона всегда держится пустой - это делается для того, чтобы главный, служебный файл (MFT) не фрагментувався в своем росте. Другие 88% диска является обычным пространством для хранения файлов.

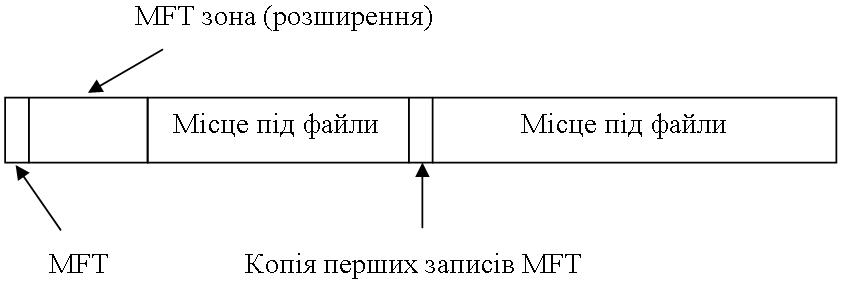


Рис. 1.4 Общая структура NTFS

Свободное место диска, включая все физически свободное место - незаполненные промежутки MFT-зоны туда тоже включаются. Механизм использования MFT-зоны таков: когда файлы уже нельзя записывать в обычное пространство, MFT-зона просто сокращается (в текущих версиях операционных систем ровно в два раза), освобождая таким образом место для записи файлов.

При увольнении места в обычной области MFT зона может снова расшириться. При этом не исключена ситуация, когда в этой зоне остались и обычные файлы: никакой аномалии тут нет. Метафайл MFT все-таки может фрагментироваться, хоть это и было бы нежелательно.

Эта основанная на атрибутах файловая система поддерживает объектно-ориентированные приложения, обрабатывая все файлы как объекты, которые имеют атрибуты, определяемые пользователем и системой.

Функциональный состав файловой системы NTFS.

Главная файловая таблица. Каждый файл на томе NTFS представлен записью в специальном файле, который называется главной файловой таблице (MFA - master file table). NTFS резервирует первые 16 записей таблицы для специальной информации. Первая запись этой таблицы описывает непосредственно главную файловую таблицу; за ней следует зеркальный запись (mirror record) MFT. Если первая запись MFT разрушен, то NTFS читает вторую запись для отыскания зеркального файла MFT, первая запись которого идентичен первой записи MFT.

Местоположение сегментов данных MFT и зеркального файла MFT записаны в секторе начальной загрузки. Дубликат сектора начальной загрузки находится в логическом центре диска.

Третья запись MFT - файл регистрации (log file) используется для восстановления файлов. Семнадцатая и последующие записи главной файловой таблицы используются собственно файлами и каталогами (также рассматриваются как файлы NTFS) на томе. На рисунке 1.5 показана упрощенная структура MFT.

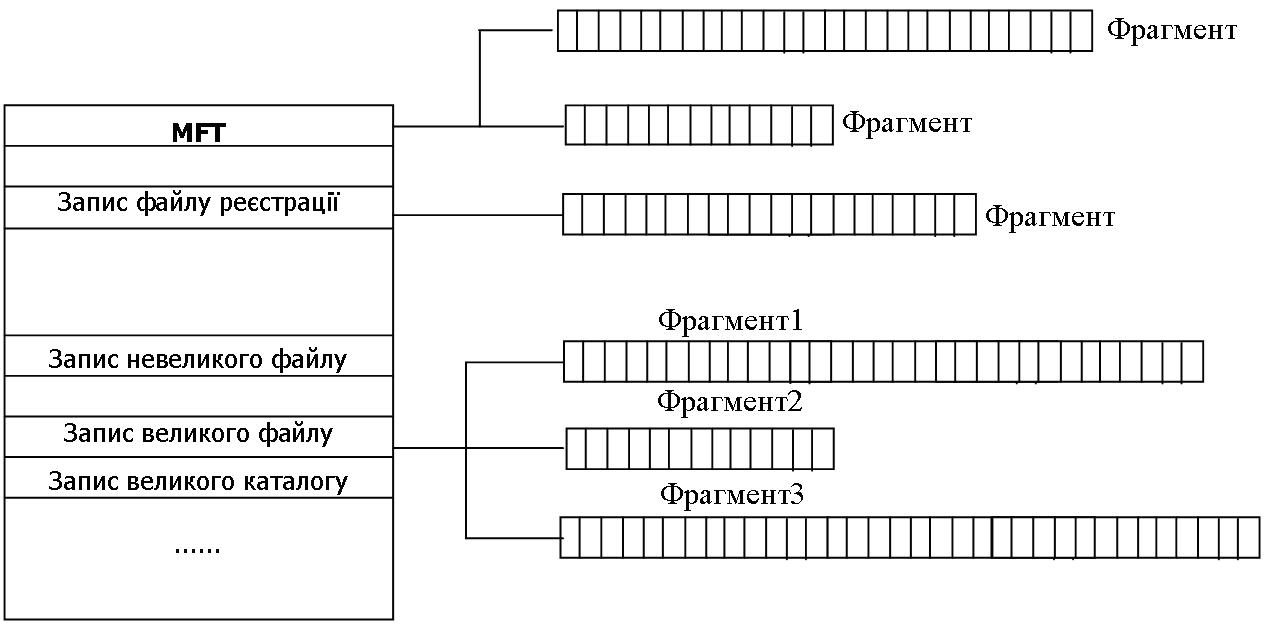


Рис. 1.5 Организация главной файловой таблицы

Главная файловая таблица отводит определенное количество пространства для каждой записи файла. Атрибуты файла записываются в распределенный пространство MFT.

Небольшие файлы и каталоги (обычно до 1500 байт или меньше) могут полностью содержаться внутри записи главной файловой таблицы. Такой подход обеспечивает очень быстрый доступ к файлам.



Рис. 1.6 Распределенный пространство MFT

Запись MFT для небольшого файла или каталога.

* + NTFS поиск файла производится только для непосредственного его использования. Записи каталога помещены внутри главной файловой таблицы так же, как записи файла. Вместо данных, каталоги содержат индексную информацию. Небольшие записи каталогов находятся полностью внутри структуры MFT. Большие каталоги организованы в B-tree, имея записи с указателями на внешние кластеры.

Первые 16 файлов NTFS (метафайлы) носят служебный характер. Каждый из них отвечает за какой-либо аспект работы системы. Преимущество модульного подхода заключается в поразительной гибкости - например, на fat и физическое повреждение в самой области fat фатально для функционирования всего диска, а NTFS может сместить, даже фрагментировать по диску, все свои служебные области, обойдя любые неисправности поверхности - кроме первые 16 элементов MFT.

Метафайлы находятся в корневом каталоге NTFS диска - они начинаются

* символа имени "$", хотя получить какую-либо информацию о них стандартными средствами сложно. Для этих файлов указанный вполне реальный размер - можно узнать, например, сколько операционная система тратит на каталогизацию всего диска, посмотрев размер файла $ MFT. В следующей таблице приведены те метафайлы, используемых на данный момент метафайлы и их назначение.

|  |  |  |
| --- | --- | --- |
|  | Табл. 1.11 Метафайлы |  |
|  |  |  |
| Название | описание |  |
| $ MFT | соответственно сам MFT |  |
| $ MFTmirr | копия первые 16 записей MFT, размещенная посередине |  |
| диска |  |
|  |  |
| $ logfile | файл поддержки журналирования (см. ниже) |  |
| $ volume | служебная информация - метка тома, версия файловой |  |
| системы, т |  |
|  |  |
| $ attrdef | список стандартных атрибутов файлов на томе |  |
| $. | корневой каталог |  |
| $ bitmap | карта свободного места том |  |
| $ boot | загрузочный сектор (если раздел |  |
| загрузочный) |  |
|  |  |
|  | файл, в котором записаны права пользователей на |  |
| $ quota | использование дискового пространства (начал |  |
|  | работать только в nt5) |  |
|  | файл - таблица соответствия заглавных и прописных |  |
|  | букв в именах файлов на текущем томе. нужен в |  |
| $ upcase | основном потому, что в NTFS имена файлов записываются в |  |
|  | unicode, что составляет 65 000 различных символов, искать |  |
|  | большие и малые эквиваленты которых очень нетривиально. |  |

Атрибуты файла NTFS.NTFS просматривает каждый файл (или каталог) как набор атрибутов файла. Такие элементы, как имя файла, зашита информация о файле и даже данные - все это атрибуты файла. Каждый атрибут идентифицирован кодом типа атрибута, но необязательно именем атрибута.

Если атрибуты файла могут находится внутри записи файла MFT, они называются резидентными (resident) атрибутами. Информация типа имени файла и отметки времени всегда включается в запись файла MFT. Если файл слишком велик, чтобы содержать все атрибуты в записи файла MFT, часть атрибутов является нерезидентной (nonresident). Нерезидентные атрибуты занимают один или несколько пробегов (run) дискового пространства в другом месте назад (пробег дискового пространства - непрерывная линейная область на диске). Вообще, все атрибуты могут быть вызваны как поток байтов независимо от того, являются ли они резидентными или нерезидентными.

* таблице представлен список всех атрибутов файла, в настоящее время определенных для NTFS. Этот список можно расширить, то есть другие атрибуты файла в будущем могут быть определены в случае необходимости.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Табл. 1.12 Атрибуты файла NTFS | |  |
|  |  |  |  |
| Тип атрибута |  | описание |  |
| Attribute List |  | Перечисляет все остальные атрибуты (только в больших |  |
| (Список атрибутов) |  | файлах) |  |
|  |  | Атрибут повторяется для длинных и для |  |
|  |  | коротких имен файлов. Длинное имя файла может |  |
|  |  | содержать до 255 символов Unicode. Короткое имя - |  |
| Filename |  | доступно, восемь плюс три символа, без |  |
| (Имя файла) |  | учета регистра. Дополнительные имена, или |  |
|  |  | жесткие связи (hard links), используются |  |
|  |  | POSIX и могут быть также включены как |  |
|  |  | дополнительные атрибуты имени файла |  |
| Security Descriptor |  | Фиксирует информацию о том, кто может обращаться |  |
| (Дескриптор безопасности) |  | к файлу, кто является его владельцем и т.д. |  |
| Data |  | Содержит данные файла |  |
| (Данные) |  |  |  |
| Index Root |  | Используется при работе с каталогами |  |
| (Корень индексов) |  |  |  |
| Index Allocation |  | Используется при работе с каталогами |  |
| (Индексное размещения) |  |  |  |
| Volume Information |  | Используется только в системном файле |  |
| (Информация назад) |  | поэтому и включает в частности версию и имя том |  |
| Bitmap |  | Предоставляет информацию об использовании записей |  |
| (Битовый массив) |  | в MFT или каталоге |  |
| Extended Attribute |  | Используется файловыми серверами, |  |
| Information |  | связанные с системами OS / 2 Этот тип атрибута не |  |
| (информация |  | используется Windows NT |  |
| расширенного атрибута) |  |  |  |
| Extended Attributes |  | Используется файловыми серверами, |  |
|  | связанные с системами OS / 2 Этот тип атрибута не |  |
| (Расширенные атрибуты) |  |  |
|  | используется Windows NT |  |
|  |  |  |

NTFS поддерживает имена файлов до 255 символов. Имена файлов NTFS используют набор символов Unicode с 16 битами; однако вопрос доступа решен. NTFS автоматически генерирует поддерживаемое MS-DOS имя (восемь плюс три символа) для каждого файла.

Таким образом, файлы NTFS могут использоваться по сети. Это особенно важно для файловых серверов организации, которая использует персональные компьютеры с двумя или всеми тремя данным операционными системами.

Как и любая другая система, NTFS делит все полезное место на кластеры - блоки данных, используемые единовременно. NTFS поддерживает почти любые размеры

кластеров - от 512 байт до 64 Кбайт, неким стандартом же считается кластер размером 4 Кбайт.

Никаких аномалий кластерной структуры NTFS не имеет. Поиск свободного места в NTFS проводится в том случае, если файл нужно создать с нуля или скопировать на диск. Поиск места под физические данные файла зависит от того, как хранится информация о занятых участка диска. NTFS имеет битовую карту свободного места, одному кластеру соответствует 1 бит.

Для поиска файла с данным именем в линейном каталоге, таком, например, как в FAT, операционной системе приходится просматривать все элементы каталога, пока она не найдет нужный. Поскольку внутренняя структура каталога является бинарным деревом, то имена файлов располагаются так, чтобы поиск файла осуществлялся более быстрым способом - с помощью получения двухзначных ответов на вопросы о положении файла.

Вопрос, на который бинарное дерево способно дать ответ, таков: в какой группе, относительно данного элемента, находится нужное имя - выше или ниже?

Мы начинаем с такого вопроса к среднему элемента, и каждый ответ сужает зону поиска в среднем в два раза, как показано на рисунке 1.7. Файлы, например, просто отсортированы по алфавиту, и ответ на вопрос осуществляется очевидным способом - сравнением начальных букв.

Область поиска, суженная в два раза, начинает исследоваться аналогичным образом, начиная опять же со среднего элемента.

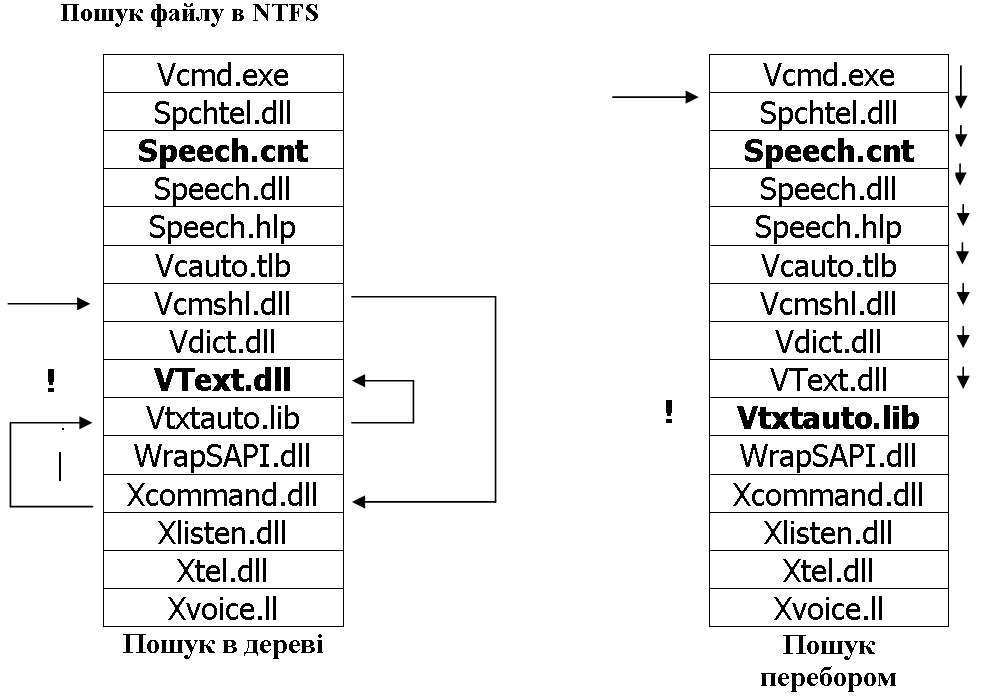


Рис. 1.7 Поиск файла в NTFS

NTFS - отказоустойчивая система, которая вполне может привести себя в корректное состояние при практически любых реальных сбоях. Любая из современных файловых систем основана на таком понятии, как транзакция - действие, которое выполняется полностью и корректно или не выполняется вообще. В NTFS просто не бывает промежуточных (ошибочных или некорректных) состояний - квант изменения данных не может быть разделен на "до" и "после" сбоя, принося разрушения и непонятность - он или есть и как следствие определенный результат, или его отменен.

Файлы NTFS имеют один довольно полезный атрибут - "сжатый". Дело в том, что NTFS имеет встроенную поддержку сжатия дисков - то, для чего раньше приходилось использовать stacker или doublespace. Любой файл или каталог в индивидуальном порядке может хранится на диске в сжатом виде - этот процесс совершенно прозрачен для приложений.

Сжатие файлов имеет очень высокую скорость и только одна большая отрицательное свойство - огромная виртуальная фрагментация сжатых файлов, которая, правда, никому особо не мешает. Сжатие осуществляется блоками по 16 кластеров и использует так называемые "виртуальные кластеры" - опять же предельно гибкое решение, позволяющее добиться интересных эффектов - например, половина файла может быть сжата, а половина - нет. Это достигается благодаря тому, что хранение информации о компрессирования определенных фрагментов очень похоже на обычную фрагментацию файлов: например, типичный запись физической раскладки для реального, несжатого, файла:

* кластеры файла с 1 по 43-й хранятся в кластерах диска начиная с 400-м;
* кластеры файла с 44 по 52-й хранятся в кластерах диска начиная с 8530-м.

Физическая раскладка типичного сжатого файла:

* кластеры файла с 1 по 9-й хранятся в кластерах диска начиная с 400-м;
* кластеры файла с 10 по 16-й нигде не хранятся;
* кластеры файла с 17 по 18-й хранятся в кластерах диска начиная с 409-м;
* кластеры файла с 19 по 36-й нигде не хранятся.

Сжатый файл имеет "виртуальные кластеры", как показано на рисунке, реальной информации в которых нет. Как только система видит такие виртуальные кластеры, она сразу понимает, что данные предыдущего блока, кратного 16-ти, должны быть не сжатые, а данные, вышедшие раз заполнят виртуальные кластеры - вот, по сути, и весь алгоритм.

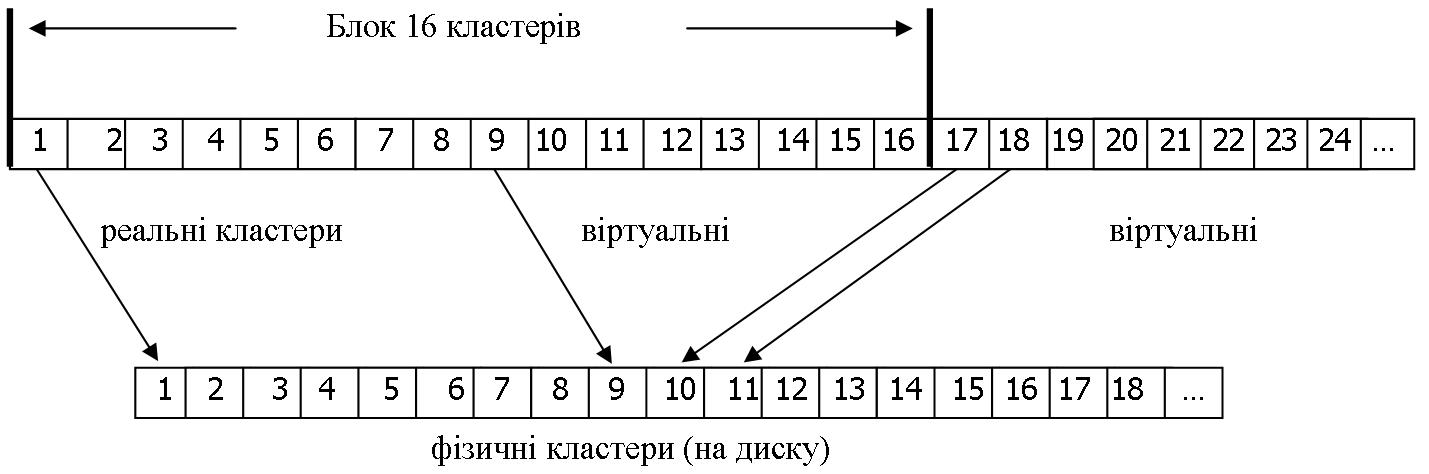


Рис. 1.8 Сжатие файла в NTFS

NTFS содержит множество средств разграничения прав объектов - есть мнение, что это самая файловая система из всех ныне существующих.

Недостатки и преимущества FAT и NTFS.

FAT - преимущества:

* Для эффективной работы требуется немного оперативной памяти;
* Быстрая работа с малыми и средними каталогами;
* Диск выполняет в среднем меньшее количество движений головок (по сравнению с NTFS)
* Эффективная работа на медленных дисках.
* Катастрофическая потеря быстродействия с увеличением фрагментации, особенно для больших дисков (только FAT32)
* Сложности с произвольным доступом к большим (скажем, 10% и более от размера диска) файлов;
* Очень медленная работа с каталогами, содержащими большое количество файлов.

NTFS - преимущества:

* Фрагментация файлов не имеет практически никаких последствий для самой файловой системы;
* Работа фрагментированной системы ухудшается только с точки зрения доступа к самим данным файлов;
* Сложность структуры каталогов и число файлов в одном каталоге не оказывает особых препятствий быстродействию;
* Быстрый доступ к произвольному фрагменту файла (например, редактирование больших wav файлов);
* Очень быстрый доступ к маленьким файлам (несколько сотен байт) - весь файл находится в том же месте, где и системные данные (запись MFT).

NTFS - недостатки:

* Существенные требования к памяти системы (64 Мбайт - абсолютный минимум)
* Медленные диски и контроллеры без Bus Mastering сильно снижают быстродействие NTFS;
* Работа с каталогами средних размеров затруднена тем, что они почти всегда фрагментированы;
* Диск, долго работает в заполненном на 80% - 90% состоянии, будет показывать очень низкое быстродействие.

Организация данных на лазерном диске.

Сектора тома организованы в виде логических секторов. Обычно размер логического сектора составляет 2048 байтов, однако стандарт допускает также использование секторов, имеющих размер 512 или 1024 байта. Логический сектор может состоять из нескольких расположенных последовательно физических секторов. Логические сектора не должны перекрываться. Идентифицировать логический сектор можно с уникальным номером (Logical Sector Number). Нумерация логических секторов начинается с нуля. Информация в каждом томе диска представлена ​​в виде набора логических секторов. Такой набор секторов называется пространством назад (Volume Space). Пространство том организован в виде одномерного (линейного) массива байтов. Байты в пространстве том нумеруются последовательно, начиная с единицы. Первый байт пространства том соответствует первому байту первого входящего в состав их логической сектора.