

О. В. КАРАБАЕВА

ДИСКИ И ФАЙЛОВАЯ ОРГАНИЗАЦИЯ

Учебное пособие

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

О. В. КАРАВАЕВА

ДИСКИ И ФАЙЛОВАЯ ОРГАНИЗАЦИЯ

Учебное пособие

Киров
2018

УДК [004.355.085.3+004.451.53](07)

К21

Рекомендовано к изданию методическим советом
факультета автоматики и вычислительной техники ВятГУ

Допущено редакционно-издательской комиссией методического совета
ВятГУ в качестве учебного пособия для студентов направлений 09.03.01 «Ин-
форматика и вычислительная техника», а также для других направлений фа-
культета автоматики и вычислительной техники

Рецензенты:

канд. техн. наук, доцент

кафедры автоматики и телемеханики ВятГУ

В. И. Семеновых;

канд. техн. наук, доцент кафедры математических
и естественно-научных дисциплин КФ МФЮА

Т. А. Анисимова

Караваяева, О. В.

К21 Диск и файловая организация : учебное пособие / О. В. Караваяева. –
Киров : ВятГУ, 2018. – 81 с.

Пособие предназначено для студентов направлений 09.03.01 «Информатика и
вычислительная техника» всех профилей подготовки, всех форм обучения для выполнения
лабораторных работ по дисциплине «Системное программное обеспечение».

УДК [004.355.085.3+004.451.53](07)

© ВятГУ, 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	5
1. ОРГАНИЗАЦИЯ ЖЕСТКОГО ДИСКА	6
1.1. Физическая структура жесткого диска. MBR	6
1.2. Логическая структура жесткого диска	8
1.2.1. Диск со схемой разделов MBR	8
1.2.2. Загрузка ОС. MBR	11
1.2.3. Определение стандарта UEFI	13
1.2.4. Диск со схемой разделов GPT	14
1.2.5. Загрузка ОС. GPT	17
1.2.6. Сравнение MBR и GPT	18
1.2.7. Совместимость с операционными системами	19
2. ФАЙЛОВЫЕ СИСТЕМЫ. ОРГАНИЗАЦИЯ ДАННЫХ	21
2.1. Общая характеристика файловых систем	21
2.1.1. Виртуальная и реальная файловая память	22
2.1.2. Логическая и физическая организации	25
2.1.3. Процедуры доступа. Файловые справочники	25
2.2. История развития файловых систем	26
2.3. Файловая система FAT	31
2.3.2. Таблица размещения файлов FAT	36
2.3.3. Кластеры FAT	39
2.3.4. FAT32	40
2.3.5. Ошибки файловой системы FAT	42
2.3.6. Создание и именование файлов	44
2.3.7. Особенности длинных имен в ОС Windows	46
2.3.8. Создание каталогов (папок)	49
2.3.9. Сравнительные характеристики FAT разных версий и NTFS	50
2.2.10. Исследование FAT32 с помощью программы WinHEX	52
2.3. Файловая система NTFS	58
2.3.2. Метафайлы	60
2.3.3. Файлы и потоки	61
2.3.4. Каталоги	62
2.3.5. Журналирование	64

2.3.6. Сжатие	65
2.4. Сравнительные характеристики файловых систем	66
ЗАКЛЮЧЕНИЕ	72
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	74
СПИСОК СОКРАЩЕНИЙ	75

ВВЕДЕНИЕ

Каждый, кто когда-либо работал с персональным компьютером, так или иначе сталкивался с работой с файлами, хранящимися на жестком диске компьютера или на внешнем носителе. История систем управления данными во внешней памяти начинается еще с магнитных лент, но современный облик они приобрели с появлением магнитных дисков. До этого каждая прикладная программа сама решала проблемы структуризации данных во внешней памяти и именования данных.

Это затрудняло поддержание на внешнем носителе нескольких архивов долговременно хранящейся информации. Историческим шагом стал переход к использованию централизованных систем управления файлами. Система управления файлами берет на себя распределение внешней памяти, отображение имен файлов в адреса внешней памяти и обеспечение доступа к данным.

Рассмотрение принципов построения и особенностей функционирования файловых систем стоит начать с основ физической организации носителя большинства файловых систем – жесткого диска.

1. ОРГАНИЗАЦИЯ ЖЕСТКОГО ДИСКА

1.1. Физическая структура жесткого диска. MBR

Основным типом устройств, которые используются в современных вычислительных системах для хранения файлов, являются дисковые накопители. Эти устройства предназначены для считывания и записи данных на жесткие и гибкие магнитные диски. Жесткий диск состоит из одной или нескольких стеклянных или металлических пластин, каждая из которых покрыта с одной или двух сторон магнитным материалом. Таким образом, диск в общем случае состоит из пакета пластин (рисунок 1.1.).



Рисунок 1.1. Устройство жесткого диска

На каждой стороне каждой пластины размечены тонкие концентрические кольца – дорожки (англ. tracks), на которых хранятся данные. Количество дорожек зависит от типа диска. Нумерация дорожек начинается с нуля от внешнего края к центру диска. Когда диск вращается, элемент, называемый головкой, считывает двоичные данные с магнитной дорожки или записывает их на нее.

Головка может позиционироваться над заданной дорожкой и перемещаться над поверхностью диска дискретными шагами, каждый шаг соответствует сдвигу на одну дорожку. Запись на диск осуществляется благодаря способности головки изменять магнитные свойства дорожки. В некоторых дисках вдоль каждой поверхности перемещается одна головка, а в других – имеется по головке на каждую дорожку. В первом случае для поиска информации головка должна перемещаться по радиусу диска. Обычно все головки закреплены на

едином перемещающем механизме и двигаются синхронно. Поэтому, когда головка фиксируется на заданной дорожке одной поверхности, все остальные головки останавливаются над дорожками с такими же номерами. В тех же случаях, когда на каждой дорожке имеется отдельная головка, никакого перемещения головок с одной дорожки на другую не требуется, за счет этого экономится время, затрачиваемое на поиск данных.

Совокупность дорожек одного радиуса на всех поверхностях всех пластин пакета называется цилиндром (англ. cylinder). Каждая дорожка разбивается на фрагменты, называемые секторами (англ. sectors) или блоками (англ. blocks), так что все дорожки имеют равное число секторов, в которые можно максимально записать одно и то же число байт. Сектор имеет фиксированный для конкретной системы размер, выражающийся степенью двойки. Чаще всего размер сектора составляет 512 байт. Учитывая, что дорожки разного радиуса имеют одинаковое число секторов, плотность записи становится тем выше, чем ближе дорожка к центру.

Сектор – наименьшая адресуемая единица обмена данными дискового устройства с ОП. Для того чтобы контроллер мог найти на диске нужный сектор, необходимо задать ему все составляющие адреса сектора: номер цилиндра, номер поверхности и номер сектора. В связи с тем, что прикладной программе в общем случае нужен не сектор, а некоторое количество байт, не обязательно кратных размеру сектора, то типичный запрос включает чтение нескольких секторов, содержащих требуемую информацию, и одного или двух секторов, содержащих наряду с требуемыми избыточные данные (рисунок 1.2.).

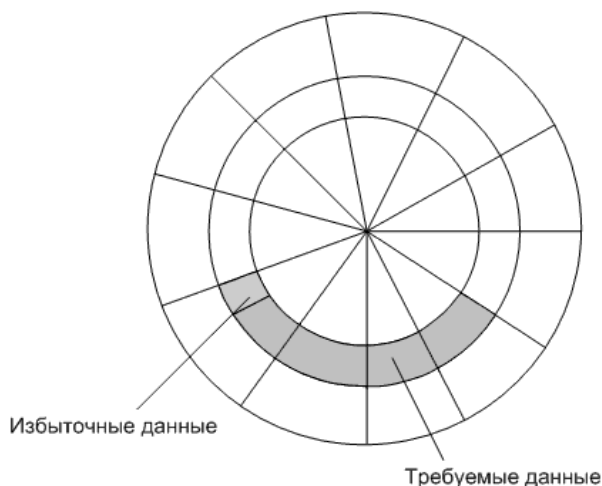


Рис. 1.2. Избыточные данные в кластере жесткого диска

Операционная система при работе с диском использует, как правило, собственную единицу дискового пространства, называемую кластером. При создании файла место на диске ему выделяется кластерами. Например, если файл имеет размер 2560 байт, а размер кластера в файловой системе определен в 1024 байта, то файлу будет выделено на диске 3 кластера.

Дорожки и секторы создаются в результате выполнения процедуры физического, или низкоуровневого форматирования диска, предшествующей использованию диска. Для определения границ блоков на диск записывается

идентификационная информация. Низкоуровневый формат диска не зависит от типа ОС, которая этот диск будет использовать.

Разметку диска под конкретный тип файловой системы выполняют процедуры высокоуровневого, или логического, форматирования. При высокоуровневом форматировании определяется размер кластера и на диск записывается информация, необходимая для работы файловой системы, в том числе информация о доступном и неиспользуемом пространстве, о границах областей, отведенных под файлы и каталоги, информация о поврежденных областях.

Жесткий диск может содержать до четырех основных разделов. Это ограничение связано с характером организации данных на жестких дисках IBM-совместимых компьютеров, но если необходимо большее количество разделов на жестком диске, то один из основных разделов может быть заменен на расширенный раздел, который в последствии может быть разбит на дополнительные логические диски.

Когда вы нажимаете кнопку питания на своём ПК, стартует процесс, который в итоге приведёт к загрузке операционной системы в память. Первая команда зависит от того, какова структура разделов на вашем жёстком диске. Есть два вида структур разделов: MBR и GPT. Структура разделов на диске определяет три вещи:

- структура данных на диске;
- код, который используется при загрузке, если раздел загрузочный;
- где начинается и заканчивается раздел.

Независимо от аппаратуры и операционной системы, все компьютеры при загрузке используют или традиционный метод BIOS-MBR, или более современный UEFI-GPT, реализованный в последних версиях ОС.

1.2. Логическая структура жесткого диска

Прежде чем использовать диск, его необходимо разбить на разделы. MBR (Главная загрузочная запись) и GPT (Таблица разделов GUID) представляют собой два различных способа хранения информации о разделах диска. Сюда входят данные о начале и конце разделов, чтобы система знала, к какому разделу принадлежит каждый сектор, и какой раздел является загрузочным.

1.2.1. Диск со схемой разделов MBR

В первом физическом секторе жесткого диска располагается главная загрузочная запись и таблица разделов (таблица 1.1.).

Главная загрузочная запись (англ. Master Boot Record, MBR) – первая часть данных на жестком диске. Она зарезервирована для программы начальной загрузки BIOS ROM Bootstrap routine, которая при загрузке с жесткого диска считывает и загружает в память первый физический сектор на активном разделе диска, называемый загрузочным сектором (англ. Boot Sector). Программа, расположенная в MBR, носит название внесистемного загрузчика (Non-System Bootstrap, NSB).

Для того, чтобы выполнить загрузку операционной системы, нужно передать управление логическому диску, запись о котором в таблице разделов помечена как активная. Таблица разделов также находится в MBR и содержит четыре записи. Каждая такая запись содержит начальную позицию и размер раздела на жестком диске, а также информацию о том, на первый сектор какого раздела необходимо передать управление (таблица 1.2.).

Можно сказать, что таблица разделов – одна из наиболее важных структур данных на жестком диске. Если эта таблица повреждена, то не только не будет загружаться ни одна из установленных на компьютере ОС, но станут недоступными данные, расположенные в диске, особенно если жесткий диск был разбит на несколько разделов.

Таблица 1.1

Структура первого физического сектора

Размер (байт)	Описание
446	Загрузочная запись (NSB)
16	Запись 1 раздела
16	Запись 2 раздела
16	Запись 3 раздела
16	Запись 4 раздела
2	Сигнатура 055AAh

Последние два байта таблицы разделов имеют значение 055AAh, то есть чередующиеся значения 0 и 1 в двоичном представлении данных. Эта сигнатура выбрана для того, чтобы проверить работоспособность всех линий передачи данных. Значение 055AAh, присвоенное последним двум байтам и имеется во всех загрузочных секторах.

Структура описания разделов

Смещение	Длина	Описание
00h	1	Признак активности раздела
01h	1	Начало раздела – головка
02h	1	Начало раздела – сектор (биты 0-5), цилиндр (биты 6,7)
03h	1	Начало раздела – цилиндр (старшие биты 8, 9 хранятся в байте номера сектора)
04h	1	Код типа раздела
05h	1	Конец раздела – головка
06h	1	Конец раздела – сектор (биты 0-5), цилиндр (биты 6,7)
07h	1	Конец раздела – цилиндр (старшие биты 8, 9 хранятся в байте номера сектора)
08h	4	Смещение первого сектора
0Ch	4	Количество секторов раздела

Многие ОС позволяют создавать, так называемый, расширенный (англ. extended) раздел, который может быть разбит на несколько логических дисков (англ. Logical disks). В этом смысле термин «первичный» можно признать не совсем удачным переводом слова «primary» – лучше было бы перевести «простейший», или «примитивный». В этом случае становится понятным и логичным термин «расширенный». Расширенный раздел содержит вторичную запись MBR (англ. Secondary MBR, SMBR), в состав которой вместо таблицы разделов входит аналогичная ей таблица логических дисков (англ. Logical Disks Table, LDT). Таблица логических дисков описывает размещение и характеристики раздела, содержащего единственный логический диск, а также может специфицировать следующую запись SMBR. Следовательно, если в расширенном разделе создано K логических дисков, то он содержит K экземпляров SMBR, связанных в список. Каждый элемент этого списка описывает соответствующий логический диск и ссылается (кроме последнего) на следующий элемент списка. Программа, расположенная в MBR, носит название внесистемного загрузчика Non-System Bootstrap. Загрузчик NSB служит для поиска с помощью таблицы разделов активного раздела, затем копирования в оперативную память компьютера систем-

ного загрузчика (англ. System Bootstrap, SB) из выбранного раздела и передачи на него управления, что позволяет осуществить загрузку ОС.

Вслед за сектором MBR размещаются собственно сами разделы (рисунок 1.3.). В процессе начальной загрузки сектора MBR, содержащего таблицу разделов, работают программные модули BIOS. Начальная загрузка считается выполненной корректно только в том случае, если таблица разделов содержит допустимую информацию.

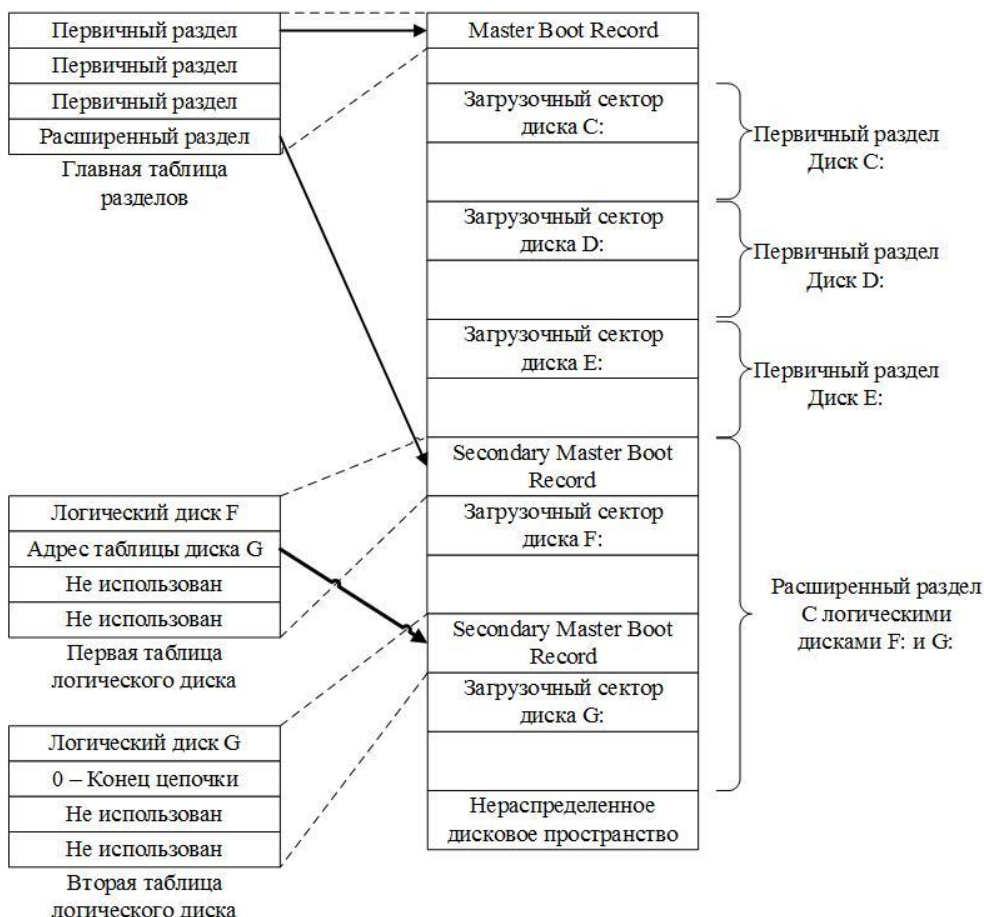


Рис. 1.3. Разбиение диска на разделы

1.2.2. Загрузка ОС. MBR

Загрузка операционной системы после включения компьютера – многоступенчатый процесс. Большинство ПК на сегодняшний день подготавливают свою аппаратную часть к работе с помощью прошивки BIOS. Во время старта BIOS инициализирует системные устройства, затем ищет загрузчик в MBR первого устройства хранения (HDD, SSD, DVD-R диск или USB-накопитель) или на первом разделе устройства (поэтому, для того чтобы выполнить загрузку с другого накопителя, нужно поменять приоритет загрузки в BIOS представленном на рисунке 1.4).

Далее BIOS передает управление загрузчику, который считывает информацию из таблицы разделов и готовится загрузить ОС. Завершает процесс – специальная сигнатура 55hAAh, которая идентифицирует главную загрузочную запись (загрузка ОС началась). Сигнатура находится в самом конце первого сектора, в котором расположен MBR.

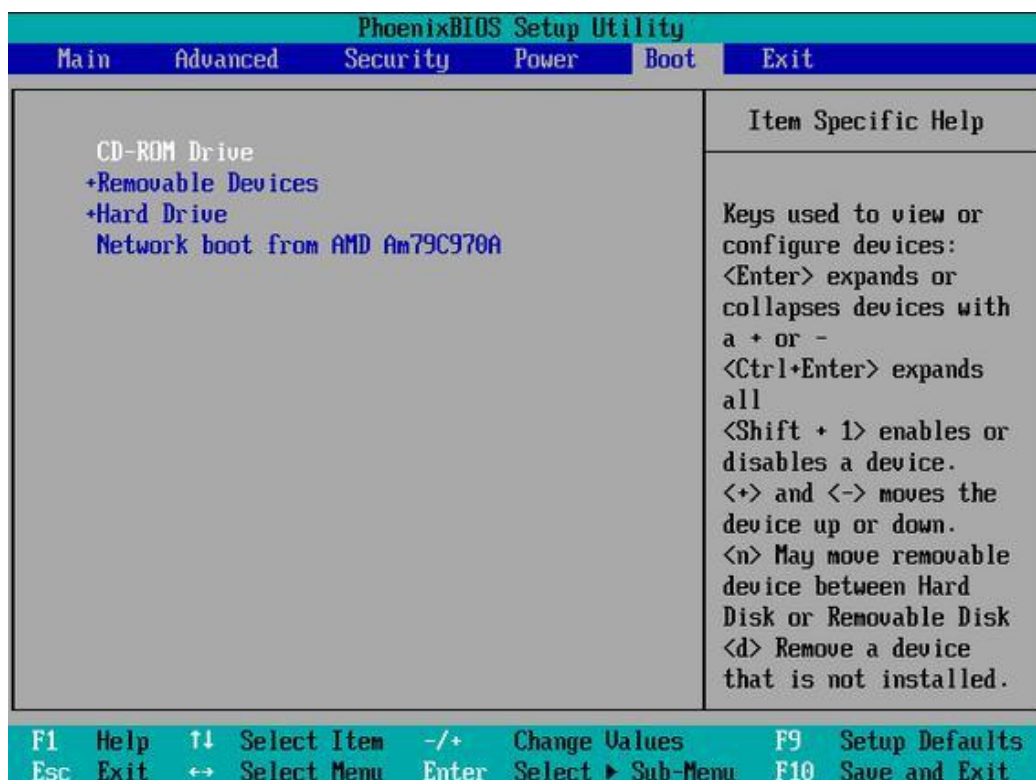


Рис. 1.4. Порядок загрузки с устройств хранения

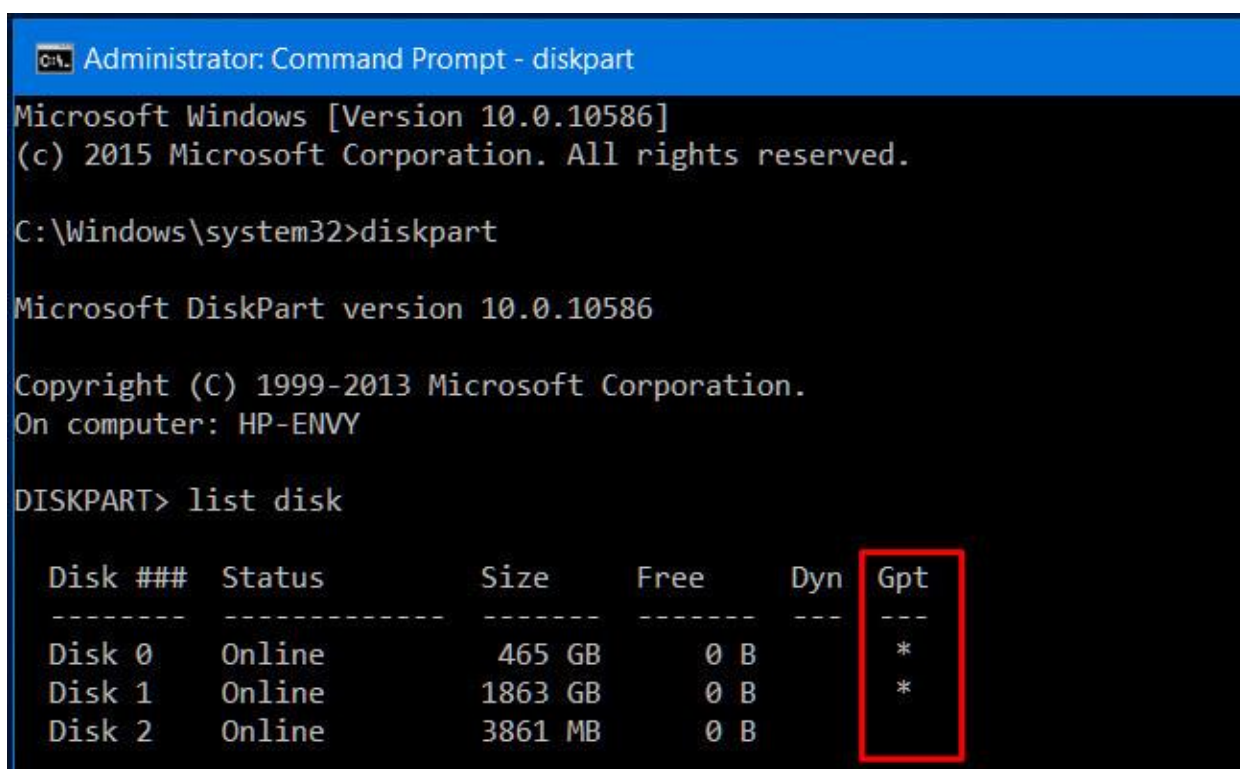
Технология MBR впервые была применена в 80-х годах еще в первых версиях DOS. По прошествии времени MBR была отшлифована и обкатана со всех сторон. Она считается простой и надежной. Но при росте вычислительных мощностей, выросла и потребность в больших объемах памяти носителей. С этим возникли сложности, так как технология MBR поддерживает работу накопителей объемом только до 2,2 ТБ. Также MBR не может поддерживать более 4 основных разделов на одном диске.

Если же необходимо создание, к примеру, 6 разделов, то потребуется превращать один из разделов в расширенный и делать из него 3 логических раздела. Для таких целей используется технология EBR – расширенная загрузочная ось. Это не совсем удобно, поэтому требовалась новая концепция, которая сможет исправить недочеты предшественницы. И она появилась в новой технологии под названием GPT.

1.2.3. Определение стандарта UEFI

Некоторое время назад существовал только BIOS в качестве API, он помогал выполнять настройки компьютерного оборудования. Но эта система была шестнадцатибитной, уже устаревшей морально. Крупнейший производитель Intel смог создать отличную и эффективную альтернативу, получившую наименование UEFI. Вместе с ней появилось огромное количество новшеств, среди которых выделялся GPT.

UEFI (UnifiedExtensibleFirmwareInterface) – это новый индустриальный стандарт, описывающий различные интерфейсы, которые система должна предоставлять до загрузки операционной системы. UEFI управляет системой с момента включения питания до окончания загрузки операционной системы. Кроме того, UEFI отвечает за интерфейс между аппаратными ресурсами и операционной системой. Другими словами, UEFI является заменой и расширением традиционной BIOS.



```
Administrator: Command Prompt - diskpart
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>diskpart

Microsoft DiskPart version 10.0.10586

Copyright (C) 1999-2013 Microsoft Corporation.
On computer: HP-ENVY

DISKPART> list disk

Disk ###  Status             Size             Free             Dyn  Gpt
-----  -
Disk 0    Online            465 GB           0 B
Disk 1    Online            1863 GB          0 B
Disk 2    Online            3861 MB          0 B
```

Рис. 1.5. Командная строка GPT

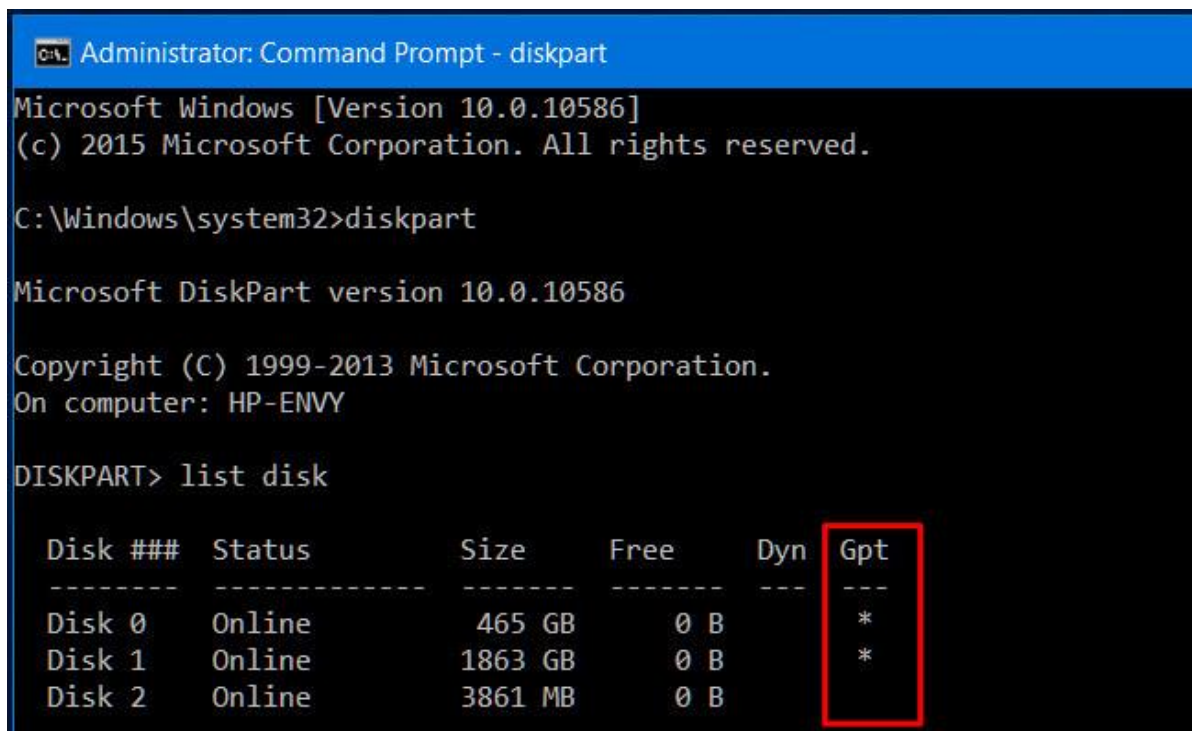
UEFI не так молод. Intel начала разработку EFI / UEFI в середине 1990 г., а крупные вендоры, такие как HP или Apple, уже довольно давно выпускают машины с EFI. Ситуация изменилась кардинально с анонсом Microsoft Windows 8, для которой UEFI стал необходим для загрузки на новых сертифицированных компьютерах.

Спецификация UEFI это объёмный документ, который описывает все интерфейсы, переменные и структуры которые производители прошивок должны реализовать и которые должны быть доступны операционной системе.

Одной из ключевых особенностей UEFI является её расширяемость. В UEFI есть внутренняя виртуальная машина, которая независима от используемой архитектуры процессора. Стандарт поддерживает специальные бинарные файлы, скомпилированные под эту виртуальную машину (EFIbinaries), которые могут быть выполнены в этой среде. Эти двоичные файлы могут быть драйверами устройств, приложениями или расширениями стандарта UEFI. UEFI в некотором смысле похожа на маленькую операционную систему, которая запускается, когда включается питание компьютера, и имеет главной задачей найти и загрузить другую операционную систему.

1.2.4. Диск со схемой разделов GPT

GPT (GUID Partition Table) – новый стандарт размещения таблиц разделов на носителе информации. Он является частью расширяемого микропрограммного интерфейса EFI (ExtensibleFirmwareInterface), разработанного компанией Intel, чтобы заменить BIOS. В процессе наработок, новый тип прошивки стал называться UnifiedExtensibleFirmwareInterface (UEFI). Одной из главных целей UEFI (рисунок 1.5.1) – стало создание нового способа загрузки ОС, который отличается от обычного загрузочного кода MBR.



```
Administrator: Command Prompt - diskpart
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>diskpart

Microsoft DiskPart version 10.0.10586

Copyright (C) 1999-2013 Microsoft Corporation.
On computer: HP-ENVY

DISKPART> list disk
```

Disk ###	Status	Size	Free	Dyn	Gpt
Disk 0	Online	465 GB	0 B		*
Disk 1	Online	1863 GB	0 B		*
Disk 2	Online	3861 MB	0 B		

Рис. 1.5.1. Микропрограммный интерфейс EFI

GPT расположена в начале жесткого диска, так же как и MBR, только не в первом, а во втором секторе. Первый сектор по-прежнему остается зарезервированным для MBR, которая также может быть в дисках GPT. Это сделано в целях защиты и для того, чтобы обеспечить совместимость со старыми ОС. GPT не ограничивает свой объем в одном секторе (512 байт).

В целом структура GPT схожа с предшественницей, за исключением некоторых особенностей:

- для таблицы разделов в Windows резервируется 16384 байт (если используется сектор в 512 байт, то доступно 32 сектора);
- GPT имеет функцию дублирования – оглавление и таблица разделов записаны в начале и в конце диска;
- количество разделов не ограничено, но технически сейчас существует ограничение в 264 раздела из-за разрядности полей;
- GPT позволяет создавать разделы диска размером до 9,4 ЗБ (это $9,4 \times 1021$ байт; объем, который вмещают в себя 940 миллионов дисков по 10 ТБ каждый). Этот факт сметает на нет проблему ограничения носителей информации в 2,2 ТБ под управлением MBR;
- GPT позволяет назначить разделам уникальный 128-битный идентификатор (GUID), имена, атрибуты. Благодаря использования стандарта кодирования символов юникод, разделы могут быть названы на любом языке и сгруппированы по папкам.

GUID Partition Table Scheme

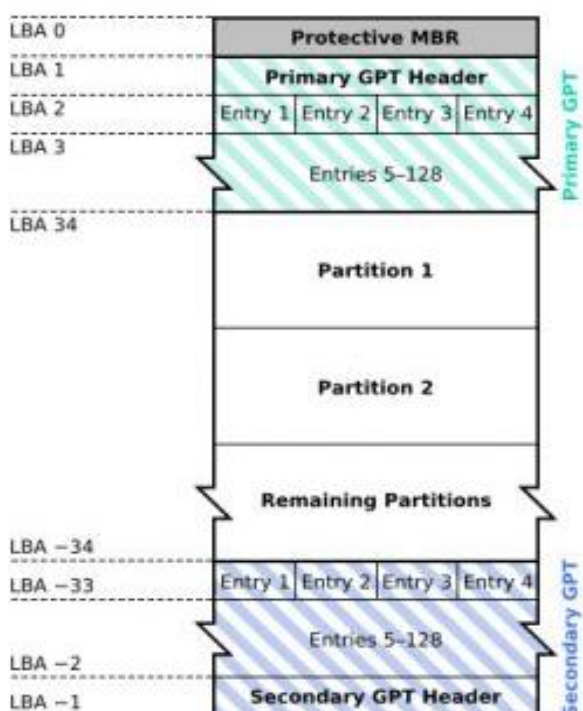


Рис. 1.6. Диаграмма, поясняющая работу GPT

Принцип организации дискового пространства для данного стандарта GPT представлен на рисунке 1.6.

Каждый LBA блок имеет размер 512 байт.

LBA0 – содержит MBR для защитных целей, так как MBR-ориентированные дисковые утилиты могут не распознать и даже переписать GPT диск. Во избежание этого указывается наличие всего одного раздела, охватывающего весь GPT диск. Системный Идентификатор (System ID) для этого раздела устанавливается в значение 0xEE, указывающее, что на данном диске применяется GPT. Вследствие этого EFI игнорирует MBR. Некоторые 32-битные ОС, не

«умеющие» читать GPT диски, распознают этот Системный Идентификатор и представляют том в качестве недоступного GPT диска. Более старые ОС представляют диск, как содержащий единственный раздел неизвестного типа без свободного места и отказываются модифицировать такой диск, пока пользователь явно не потребует и не подтвердит удаление данного раздела, таким образом, предотвращается случайное стирание содержимого GPT диска.

LBA1 – оглавление таблицы разделов указывает те логические блоки на диске, которые могут быть задействованы пользователем (the usable blocks). Оно также указывает число и размер записей данных о разделах, составляющих таблицу разделов. Например, на компьютере с установленной 64-битной ОС Windows Server 2003, зарезервировано 128 записей данных о разделах, каждая запись длиной 128 байт. То есть возможно создание 128 разделов на диске. Оглавление содержит GUID (GloballyUniqueIdentifier – Глобально Уникальный Идентификатор) диска. Он записывает свой размер и местоположение (всегда LBA 1), а также размер и местоположение вторичного (запасного) оглавления и таблицы разделов (всегда в последних секторах диска). GUID содержит контрольную сумму для себя и для таблицы разделов. Контрольные суммы проверяются процессами EFI при загрузке машины. Из-за проверок контрольных сумм невозможно применение шестнадцатеричных (hex) редакторов для изменения содержимого GPT. Любое редактирование изменит контрольные суммы, после чего EFI перезапишет первичный GPT вторичным. Если же оба GPT будут содержать неверные контрольные суммы, доступ к диску станет невозможным.

LBA2-33 – записи данных о разделах.

LBA34 – первый используемый сектор каждого жесткого диска.

Отрицательные адреса LBA говорят о том, что нумерация начинается с конца тома (диска), причем последний адресуемый блок имеет адрес -1.

Кроме того, так как GPT обеспечивает дублирование, оглавление и таблица разделов записаны как в начале, так и в конце диска.

Стандарт GPT, так же как и MBR не лишен недостатков, которые, впрочем, в целом полностью перекрываются достоинствами нового стандарта. Так, одним из главных минусов этой технологии является закрытость стандарта, из-за чего существует множество реализаций ее поддержки. Можно выделить следующие проблемы данной технологии:

- невозможно назначить имя всему диску, как отдельным разделам и его роль исполняет идентификатор GUID;

- идет привязка раздела к его номеру в таблице (сторонние загрузчики ОС предпочитают использовать номер вместо имен и GUID);
- дубликаты таблиц (PrimaryGPTHeader и SecondaryGPTHeader) строго ограничены в количестве 2 штук и имеют зафиксированные позиции. В случае повреждения носителя и наличия ошибок или некорректной работы ПО, этого может быть недостаточно для восстановления данных;
- эти 2 копии GPT (Primary и SecondaryGPTHeader) взаимодействуют друг с другом, но не позволяют удалить и перезаписать контрольную сумму в случае, если она в одной из копий не верна. Это значит, что не предусмотрена защита от плохих (битых) секторов на уровне GPT, несмотря на то, что соответствующие технологии активно используются во многих современных и не очень файловых системах.

1.2.5. Загрузка ОС. GPT

Загрузка ОС происходит совсем не так, как в BIOS. UEFI не обращается для загрузки системы к коду MBR, даже если он есть. Вместо этого используется специальный раздел на винчестере, который называется «EFISYSTEMPARTITION». В нем располагаются файлы, которые необходимо запустить для загрузки.

Загрузочные файлы хранятся в директории <EFI SYSTEM PARTITION>/EFI/<ИМЯ ВЛАДЕЛЬЦА>/. Это значит, что UEFI имеет собственный мультизагрузчик, который позволяет в разы быстрее определять и загружать нужные приложения (в BIOSMBR для этого требовались сторонние программы). Процесс загрузки UEFI происходит следующим образом:

- включение компьютера → проверка аппаратного обеспечения;
- загрузка прошивки UEFI;
- прошивка загружает диспетчер загрузки, который определяет, с каких дисков и разделов будут загружены UEFI приложения;
- запуск прошивкой UEFI приложения с файловой системой FAT32 раздела UEFISYS, как это указано в загрузочной записи менеджера загрузки микропрограммы.

1.2.6. Сравнение MBR и GPT

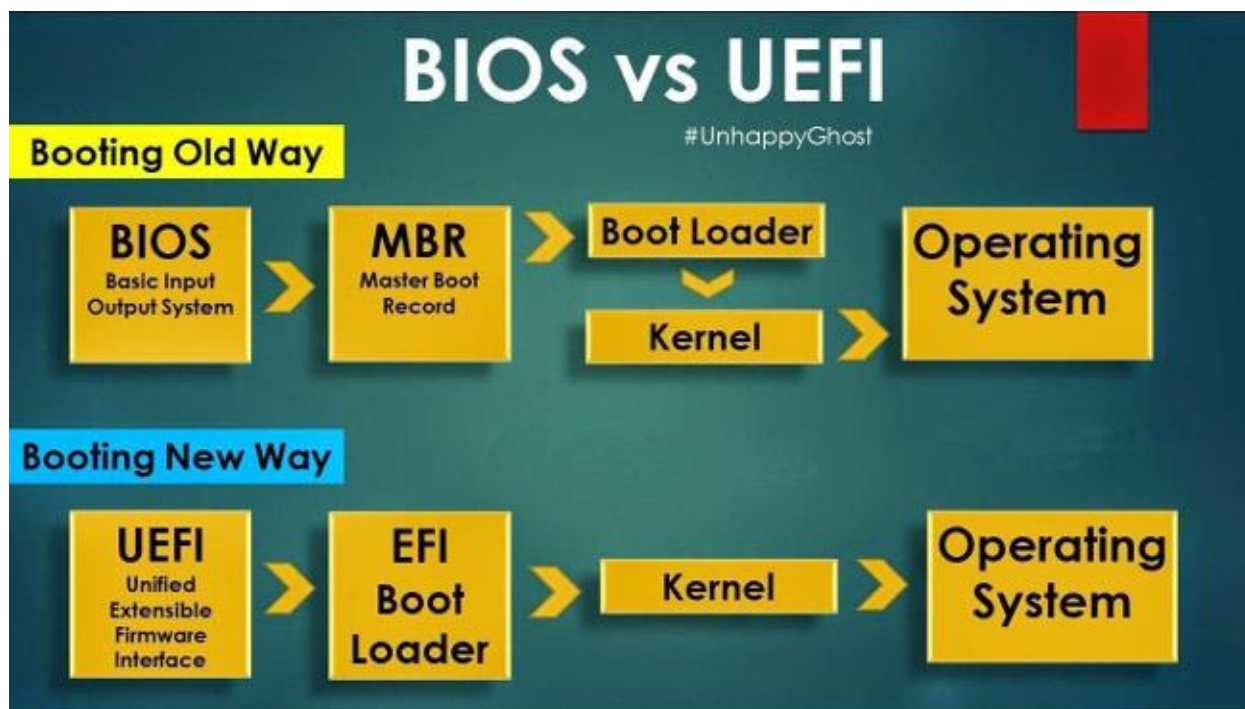


Рис. 1.7. Сравнение BIOS и UEFI

MBR – традиционная структура для управления разделами диска. Поскольку она совместима с большинством систем, то по-прежнему широко используется. Главная загрузочная запись расположена в первом секторе жёсткого диска или, проще говоря, в самом его начале. Она содержит таблицу разделов – информацию об организации логических разделов на жёстком диске.

MBR также содержит исполняемый код, который сканирует разделы на предмет активной ОС и инициализирует процедуру загрузки ОС.

Диск MBR допускает только четыре основных раздела. Если вам нужно больше, то можно назначить один из разделов расширенным разделом, и на нём можно создавать больше подразделов или логических дисков.

MBR использует 32 бита для записи длины раздела, выраженной в секторах, так что каждый раздел ограничен максимальным размером 2 ТБ.

Преимущества:

- совместима с большинством систем.

Недостатки:

- допускает только четыре раздела, с возможностью создания дополнительных подразделов на одном из основных разделов;
- ограничивает размер раздела двумя терабайтами;
- информация о разделе хранится только в одном месте – в главной загрузочной записи. Если она повреждена, то весь диск становится нечитаемым.

GPT – более новый стандарт для определения структуры разделов на диске. Для определения структуры используются глобальные уникальные идентификаторы (GUID).

Это часть стандарта UEFI, то есть систему на основе UEFI можно установить только на диск, использующий GPT, например, таково требование функции Windows 8 Secure Boot.

GPT допускает создание неограниченного количества разделов, хотя некоторые операционные системы могут ограничивать их число 128 разделами. Также в GPT практически нет ограничения на размер раздела.

Преимущества.

- допускает неограниченное количество разделов. Лимит устанавливает операционная система, например, Windows допускает не более 128 разделов;
- ограничение на максимальный размер раздела больше, чем объём любых существующих сегодня дисков. Для дисков с секторами по 512 байт поддерживается максимальный размер 9,4 ЗБ (один зеттабайт равен 1 073 741 824 терабайт)
- GPT хранит копию раздела и загрузочных данных и может восстановить данные в случае повреждения основного заголовка GPT.
- GPT хранит значения контрольной суммы по алгоритму циклического избыточного кода (CRC) для проверки целостности своих данных (используется для проверки целостности данных заголовка GPT). В случае повреждения GPT может заметить проблему и попытаться восстановить повреждённые данные из другого места на диске.

Недостатки.

- может быть несовместима со старыми операционными системами.

1.2.7. Совместимость с операционными системами

Первый сектор (сектор 0) на диске GPT содержит защитную запись MBR, в которой записано, что на диске один раздел, который распространяется на весь носитель. В случае использования старых инструментов, которые читают только диски MBR, можно увидеть один большой раздел размером с весь диск. Защитная запись сделана для того, чтобы старый инструмент ошибочно не воспринял диск как пустой и не перезаписал данные GPT новой главной загрузочной записью.

MBR защищает данные GPT от перезаписи.

Apple MacBook'и используют GPT по умолчанию, так что невозможно установить Mac OS X на систему MBR. Даже хотя Mac OS X может работать на диске MBR, но установка на него невозможна.

Большинство операционных систем на ядре Linux совместимы с GPT. При установке ОС Linux на диск в качестве загрузчика будет установлен GRUB 2.

Для операционных систем Windows загрузка из GPT возможна только на компьютерах с UEFI, работающих под 64-битными версиями Windows Vista, 7, 8, 10 и соответствующими серверными версиями. Если вы купили ноутбук с 64-битной версией Windows 8, то с большой вероятностью там GPT.

Windows 7 и более ранние системы обычно устанавливают на диски с MBR, но вы всё равно можете преобразовать разделы в GPT.

Все версии Windows 7, 8, 10 могут считывать и использовать данные из разделов GPT – но они не могут загружаться с таких дисков без UEFI.

2. ФАЙЛОВЫЕ СИСТЕМЫ. ОРГАНИЗАЦИЯ ДАННЫХ

2.1. Общая характеристика файловых систем

Файловая система – это способ хранения и организации доступа к данным на информационном носителе или его разделе. Файловая система определяет, где и каким образом на носителе будут записаны файлы, и предоставляет операционной системе доступ к этим файлам; порядок, определяющий способ организации, хранения и именования данных на носителях информации и компьютерной техники. Она определяет формат содержимого и физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имени файла (папки), максимальный возможный размер файла и раздела, набор атрибутов файла. Некоторые файловые системы предоставляют сервисные возможности, например, разграничение доступа или шифрование файлов.

Файловая система связывает носитель информации с одной стороны и API для доступа к файлам – с другой. Когда прикладная программа обращается к файлу, она не имеет никакого представления о том, каким образом расположена информация в конкретном файле, так же, как и на каком физическом типе носителя (CD, жёстком диске, магнитной ленте, блоке флеш-памяти или другом) он записан. Всё, что знает программа – это имя файла, его размер и атрибуты. Эти данные она получает от драйвера файловой системы. Именно файловая система устанавливает, где и как будет записан файл на физическом носителе (например, жёстком диске).

Файловая система является самой большой частью многих мультипрограммных систем и для ее проектирования и реализации обычно требуются наибольшие усилия.

Файл обычно определяется как набор записей данных, структурированных для того, чтобы управлять доступом к ним, их чтением и модификацией; запись данных – это просто линейный список объектов информационного характера. Примерами файлов являются программы на исходных или машинных языках, прикладные данные, такие, как счета или инвентарные списки задания пользователей, справочники файлов, а также учетные данные о работе или производительности системы. Фактически все программы и данные, к которым имеет доступ вычислительная система, т.е. все программное обеспечение ЭВМ может в некоторые моменты обработки рассматриваться как файл.

Файловая система (система управления данными) – то программное обеспечение, которое отвечает за создание, уничтожение, организацию, чтение, за-

пись, модификацию и перемещение файловой информации, а также управление доступом к файлам и за управление ресурсами, которые используются файлами. Без файловой системы мультипрограммирование в общем виде не было бы возможным. Это именно тот компонент, который обеспечивает использование многих языковых процессоров, пользовательских библиотек программ и данных, обрабатываемых в режиме «в линию», операции по организации спулинга в пакетных системах и практические интерактивные вычисления. Как мы уже упоминали раньше, весь ввод-вывод в МС обычно обслуживается посредством файловой системы.

2.1.1. Виртуальная и реальная файловая память

Обычное определение файла как набора связанных (родственных) записей данных не навязывает файлу какой-либо особой структуры (только подразделение на записи) и почти не зависит от носителя файла. Целесообразно определить единообразную бесформатную виртуальную файловую память. На самом абстрактном и неструктурированном уровне файл можно рассматривать как упорядоченную последовательность элементов с символическим именем, где элемент файла есть наименьшая адресуемая единица в виртуальном пространстве, такая, как байт или слово, i -й по порядку элемент файла с именем F адресуется парой $[F, i]$.

Цель концепции виртуальной файловой памяти – обеспечить пользователей простым единообразным линейным пространством для размещения их файлов. Это пространство может быть далее структурировано (как структурирована ЭВМ и ее реальная файловая память) любым удобным способом, чтобы отразить истинную организацию данных и их обработку, или, если известны реальные аппаратурные устройства памяти, извлечь преимущества из их физических характеристик.

Самыми основными операциями в файловой системе являются: а) отображение запросов на доступ из виртуального на реальное файловой пространство и б) передача элементов между файловой памятью и основной памятью ЭВМ. Изучение этих и других операций требует определенного знания свойств и поведения реальных устройств для хранения файлов. Мы ограничим обсуждение вспомогательным устройством (ЗУ), работающим в режиме «в линию» (on-line), которое является повторно используемым, т.е. средой памяти, с которой можно читать и на которую можно записывать много раз.

Основные устройства вспомогательной памяти, которые использованы как файловые, – это магнитная лента, диски, барабаны. При разработке систем нужно учитывать следующие показатели, причем не делается никаких различий

между устройством, на котором данные хранятся, и контроллером, который управляет выборкой устройства, считыванием, записью и другими операциями:

1. Емкость. Максимальное количество данных, которые могут храниться на устройстве.

2. Размер записи. Физическая запись есть непрерывно расположенная информация наименьшего объема, к которой можно адресоваться на устройстве. Устройство может допускать записи фиксированной или переменной длины.

3. Метод доступа. Может быть возможным прямой доступ к любой записи на устройстве или устройство может быть только с последовательным доступом. В первом случае по аппаратному адресу записи механизм считывания записи устанавливается непосредственно на запись. В случае только последовательного доступа, определенная запись достигается при явном прохождении в прямом или обратном направлении, минуя промежуточные записи; обычный способ работы состоит в том, чтобы получать доступ к записям последовательно в той же самой линейной последовательности, в которой они хранятся.

4. Сменяемость. Магнитная лента и некоторые типы дисков являются сменными, допуская существование автономной памяти. Это свойство существенно увеличивает объем виртуальной памяти, предоставляемой пользователю.

5. Скорость передачи данных. Это скорость, обычно выражаемая в битах, байтах или словах/секунду, с которой данные могут быть переданы между основной памятью и устройством. Во время передачи ввод-вывод имеет приоритет над ЦП в отношении доступа к памяти и, если они оба обращаются к одному и тому же блоку основной памяти, «крадет» циклы памяти у ЦП.

6. Задержка. После того как команда чтения или записи получена контроллером устройства, обычно требуется дополнительное время (время задержки), прежде чем головки считывания или записи переместятся к началу требуемой записи и можно будет начать передачу данных.

7. Время установки. Время на раскрутку диска.

8. Относительная стоимость. Как можно ожидать, стоимость устройства возрастает с увеличением скорости и гибкости доступа к записям.

Одна из возможных структур файловой системы, включает следующие пять модулей, связанных между собой так, как показано на рисунке 2.1.

1. Язык организации и доступа. Это та часть файловой системы, которая связана с пользователем. В этом компоненте обеспечиваются средства определения различных логических организаций в виртуальном файловом пространстве и конструкции языка для доступа к файлам. Последний включает команды и декларации для создания, разрушения, считывания, записи, модификации и управления доступом к файлам. Во многих системах эти средства охватывают и физический уровень файла так, что пользователи могут обойтись без больших частей системы и до некоторой степени прямо определять физические организации и команды ввода-вывода.

2. Процедуры доступа. Они состоят из стандартных программ управления справочниками файлов и поиска в них, открытия и закрытия файлов, отображения символических имен файлов в их действительные адреса, управления санкционированным доступом к файлам, управления внутренними буферами и генерирования соответствующих программ ввода-вывода. Как видно из этого длинного списка, процедуры доступа составляют самую большую часть большинства систем.

3. Система ввода-вывода. система ввода-вывода отвечает за поддержание очередей запросов на ввод-вывод, за планирование и запуск операций, обработку ошибок ввода-вывода, а также за обслуживание сигналов окончания ввода-вывода.

4. Управление вспомогательной памятью. Это модуль отвечает за учет доступного пространства во вспомогательной памяти и распределение и освобождение блоков вспомогательной памяти по запросам. В сложной системе можно также включить сюда средства для перемещения файлов информации по иерархии запоминающих устройств; одним из важных факторов при определении, где хранится файл в данное время, может быть его действительная или ожидаемая активность.

5. Возврат и восстановление. Система должна иметь возможность восстанавливаться при аппаратурных и программных ошибках. с этой целью последний компонент обеспечивает восстановление после ошибок файловой системы (и ОС в целом).

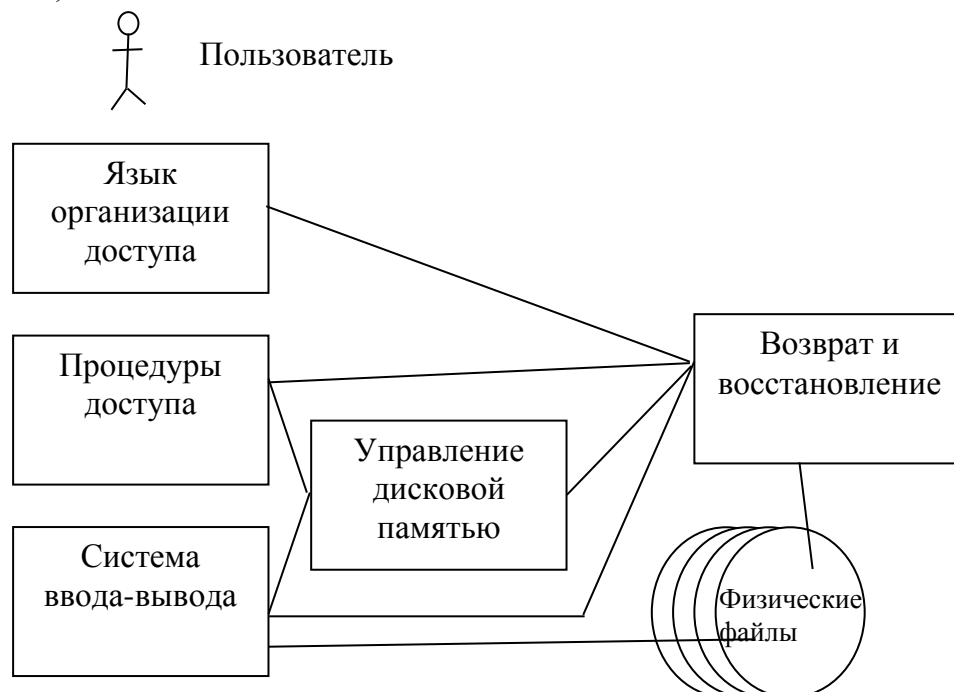


Рис. 2.1. Пример структуры файловой системы

2.1.2. Логическая и физическая организации

Несколько общих разновидностей организаций файлов рассматриваются с двух позиций: изучается логическая организация, определяемая пользователем, и соответствующая физическая организация, которая может быть эффективно реализована на устройствах вспомогательной памяти. (Мы будем также использовать термины «виртуальная» и «реальная» вместо «логическая» и «физическая» соответственно). Логический файл динамически или статически подразделяется на набор записей R_0, R_1, \dots , где запись есть непрерывный блок информации, передаваемый во время логических операций чтения или записи. аналогично, реальный файл делится на физические записи; размер записи определяется характеристиками запоминающей среды. Физическая запись может состоять из нескольких логических записей, или, наоборот, логическая запись может распространяться на несколько физических.

2.1.3. Процедуры доступа. Файловые справочники

Все системы имеют набор справочников (словарей, каталогов), которые идентифицируют и определяют местоположение всех файлов, доступных для пользовательских и системных процессов. Элемент справочника содержит как минимум имя и физический адрес файла или его дескриптора; с другой стороны, в каждом элементе может храниться весь «дескриптор» файла. Так как справочники сами являются данными, объектом поиска и модификации, то они часто рассматриваются как один файл, но специального назначения; тогда операции файловой системы можно также использовать и для управления справочником.

Наиболее общим видом практической организации справочников является древовидная структура, в которой каждая вершина дерева является справочником (файлом), а каждая ветвь – элемент справочника, который указывает или на другой справочник, или на файл данных. Если файлы данных добавляются к дереву, они будут занимать все листья. Корень дерева называется главным справочником. Если мы потребуем, чтобы имена всех ветвей (данные и справочники) с одной и той же порождающей вершиной не повторялись, тогда для каждого файла существует однозначное имя – символическое имя «пути» от корня к файлу. Это имя формируется конкатенацией идентификаторов ветвей, начиная с главного справочника и по мере прохода по дереву к желаемому файлу. (Следует заметить, что на данный файл указывает одна и только одна ветвь справочника; иначе символическое имя более не является уникальным – может существовать несколько путей от корня к файлу, но каждое имя пути все еще определяет один файл).

2.2. История развития файловых систем

В 1981 году фирма IBM представила первый персональный компьютер, который работал под управлением новой операционной системы MS-DOS, разработанной Microsoft. Этот компьютер содержал 16-разрядный процессор Intel8088 и два дисководов для гибких дисков низкой плотности. Файловая система MS-DOS, названная по её таблице размещения файлов FAT, предоставляла значительно больше возможностей, чем это требовалось для существующих тогда маленьких дисковых объёмов и управления иерархическими структурами файлов и каталогов. Эта файловая система продолжала удовлетворять потребности пользователей персональных компьютеров и в течение многих последующих лет, когда мощность аппаратных средств непрерывно возрастала. Однако поиск файлов и данных на сравнительно больших жёстких дисках стал производиться значительно менее эффективно по сравнению с гибкими дисками небольшой ёмкости первых персональных компьютеров.

Но в 1987 году возможности файловой системы FAT, разработанной за десять лет до этого фирмой Microsoft для интерпретатора Standalone Basic и позднее приспособленной для операционной системы MS-DOS, были исчерпаны. FAT предназначалась для дисков размером не более 32 мегабайт, и новые жёсткие диски большей ёмкости оказывались бесполезными для пользователей PC. Некоторые независимые поставщики предлагали свои способы решения этой проблемы, однако лишь с появлением MS-DOS 4.0 этот кризис был преодолен – на некоторое время. Значительные изменения структуры файловой системы в MS-DOS 4.0 позволили операционной системе работать с дисками ёмкостью до 128 Мбайт; с внесением в последующем небольших изменений этот предел был поднят до 2 Гбайт.

В июле 1993 года появились первые операционные системы семейства NT (NewTechnology), WindowsNT 3.1 и WindowsNTAdvancedServer 3.1, в которые была включена новая файловая система NTFS, нацеленная на эффективную работу с большими дисками, а так же на обеспечение целостности данных (защиту целостности данных при сбоях в системе).

Несмотря на появление технически более совершенных файловых систем, FAT не потеряла своей популярности из-за отсутствия поддержки более продвинутых файловых систем в популярных OS Windows 9x и из-за завышенных системных требований операционных систем семейства Windows NT. Но когда-то очень высокий потолок в 2Гбайта превратился в серьёзное препятствие. В 1997 году фирма Microsoft разработала новое расширение FAT – FAT32. FAT32 позволяет использовать диски ёмкостью до 2 Тбайт и обеспечивает бо-

лее эффективное использование жёстких дисков различного объёма (в том числе и <2 Гбайт).

В 2000 году была представлена система шифрования данных Encrypting-FileSystem (EFS), реализующая шифрование на уровне файлов в операционных системах Microsoft Windows NT (начиная с Windows 2000 и выше), за исключением «домашних» версий. Данная система предоставляет возможность «прозрачного шифрования» данных, хранящихся на разделах с файловой системой NTFS, для защиты потенциально конфиденциальных данных от несанкционированного доступа при физическом доступе к компьютеру и дискам. EFS работает, шифруя каждый файл с помощью алгоритма симметричного шифрования, зависящего от версии операционной системы и настроек (начиная с Windows XP доступна теоретическая возможность использования сторонних библиотек для шифрования данных). При этом используется случайно сгенерированный ключ для каждого файла, называемый FileEncryptionKey (FEK), выбор симметричного шифрования на данном этапе объясняется его скоростью по отношению к асимметричному шифрованию.

exFAT (от англ. ExtendedFAT – «расширенная FAT»), иногда называется FAT64 была представлена в ноябре 2006 – проприетарная файловая система, предназначенная главным образом для флэш-накопителей. Впервые представлена фирмой Microsoft для встроенных устройств в WindowsEmbeddedCE 6.0. Размер кластера по умолчанию для файловой системы exFAT составляет от 4 КБ до 128 КБ в зависимости от размера тома.

ExtendedFileSystem (расширенная файловая система), сокращённо ext или extfs – первая файловая система, разработанная специально для ОС на ядре Linux. Представлена в апреле 1992 г. для ядра Linux.

Используемая структура метаданных была разработана Реми Кардом (англ.), на создание которой его вдохновила UnixFileSystem (UFS). Целью было преодолеть ограничения файловой системы MinixFileSystem – в новой файловой системе наибольший возможный размер раздела и файла увеличен до 2 Гб, а максимальная длина имени файла – до 255 байт.

В январе 1993 года на базе ФС ext началось создание расширяемой файловой системы ext2, в которой поддерживались разделы до 2 ТБ и три метки времени. Позже ext2 стала основой для ext3 (2000 год) и ext4 (около 2006 года).

Со временем ext2 вытеснила xiafs и ext благодаря долгосрочной жизнеспособности.

В январе 1997 года поддержка ext и xiafs была окончательно удалена из ядра (с версии 2.1.2.).

Fourthextendedfilesystem (четвёртая расширенная файловая система), сокр. ext4, или ext4fs – журналируемая ФС, используемая в ОС с ядром Linux.

Основана на ФС ext3, ранее использовавшейся по умолчанию во многих дистрибутивах GNU/Linux.

Первая экспериментальная реализация ext4 была написана Эндрю Муртоном (англ.) и выпущена 10 октября 2006 года в виде патча к ядрам Linux версий 2.6.19-rc1-mm1 и 2.6.19-rc1-git8 .

Основные изменения по сравнению с ext3:

- увеличение максимального объёма одного раздела диска до 1 эксбибайта (260 байт) при размере блока 4 кибибайт;
- увеличение размера одного файла до 16 тебибайт (244 байт);
- введение механизма пространственной (extent) записи файлов, уменьшающего фрагментацию и повышающего производительность. Суть механизма заключается в том, что новая информация добавляется в конец области диска, выделенной заранее по соседству с областью, занятой содержимым файла.

В 1990 году была создана высокоэффективная файловая система HPFS (HighPerformanceFileSystem), которая являлась частью операционной системы OS/2 версии 1.x. Эта файловая система разрабатывалась для больших жёстких дисков PC. Вслед за HPFS появилась HPFS386, которая являлась частью Microsoft LAN Manager и была рассчитана на использование преимуществ 32-разрядных процессоров.

HPFS (от англ. HighPerformanceFileSystem) – это файловая система представленная в конце 1994 года и разработанная специалистами Microsoft и IBM на основе опыта IBM по созданию файловых систем MVS, VM и виртуального метода доступа. Со стороны Microsoft проектом руководил опытный системный программист Гордон Летвиню. Впервые была использована для операционной системы OS/2 1.2, чтобы обеспечить доступ к появлявшимся в то время на рынке дискам большого размера. Кроме того, назрела необходимость расширения существующей системы имен, улучшения организации и безопасности для удовлетворения растущих потребностей рынка сетевых серверов. В файловой системе HPFS поддерживается структура каталогов FAT и добавлена сортировка файлов по именам. Имя файла может содержать до 254 двухбайтовых символов. Файл состоит из «данных» и специальных атрибутов, что создает дополнительные возможности для поддержки других типов имен файлов и повышению уровня безопасности. Кроме того, наименьший блок для хранения данных теперь равен размеру физического сектора (512 байт), что позволяет снизить потери дискового пространства. Записи в каталоге файловой системы HPFS содержат больше сведений, чем в FAT. Наряду с атрибутами файла здесь хранятся сведения о создании и внесении изменений, а также дата и время доступа. Записи в каталоге файловой системы HPFS указывают не на первый кластер файла, а на FNODE. FNODE может содержать данные файла, указатели на дан-

ные файла или другие структуры, указывающие на данные файла. HPFS старается по возможности располагать данные файла в смежных секторах. Это приводит к повышению скорости последовательной обработки файла.

HPFS делит диск на блоки по 8 МБ каждый и всегда пытается записать файл в пределах одного блока. Для каждого блока 2 КБ зарезервировано под таблицу распределения, в которой содержится информация о записанных и свободных секторах в пределах блока. Разбиение на блоки приводит к повышению производительности, так как головка диска для определения места для сохранения файла должна возвращаться не к логическому началу диска (как правило, это нулевой цилиндр), а к таблице распределения ближайшего блока.

Кроме того, файловая система HPFS содержит два уникальных объекта данных – суперблоки и запасные блоки. Суперблок располагается в логическом секторе 16 и содержит указатель на FNODE корневого каталога. В этом кроется главная опасность использования HPFS: если сектор суперблока помечен как поврежденный, это приводит к потере всех данных раздела даже на неповрежденных участках диска. Для восстановления данных их необходимо скопировать на другой диск с неповрежденным сектором 16 и воссоздать суперблок. Это очень сложная задача. Запасной блок располагается в логическом секторе 17 и содержит таблицу экстренных исправлений, а также блок резервного каталога. В файловой системе HPFS запись таблицы экстренных исправлений используется при обнаружении дефектного сектора, чтобы логически указать вместо него имеющийся неповрежденный сектор. Эта технология обработки ошибок записи известна как экстренное исправление.

Если используется технология экстренного исправления, то при обнаружении поврежденного сектора данные переносятся в другой сектор, а исходный помечается как дефектный. Эти действия выполняются открыто для любого приложения, которое выполняет дисковые операции ввода/вывода (то есть на работе приложения проблемы с жестким диском не сказываются). Сообщения об ошибке, которые появляются при обнаружении поврежденного сектора (например, «FAT"Abort, Retry, orFail?"»), в файловой системе, поддерживающей экстренные исправления, отсутствуют.

HPFS – оптимальный вариант файловой системы для использования с дисками размером 200–400 МБ.

Дополнительные накладные расходы, связанные с использованием HPFS, снижают эффективность ее применения на дисках размером меньше 200 МБ. Кроме того, производительность также снижается при использовании дисков размером больше 400 МБ. При использовании HPFS под Windows NT нельзя установить параметры безопасности.

HFS+ была представлена 19 января 1998 г. вместе с Mac OS 8.1, но впервые её представили в качестве тестовой файловой системы для так и не вышедшей OS Copland (1994–1996 гг.). Начиная с 11 ноября 2002 г., с выпуском обновления 10.2.2, Apple Inc. сделала возможным журналирование для повышения надёжности хранения информации. Оно было легко доступно с серверной версией Mac OS X, но только через интерфейс командной строки с настольных клиентов. Начиная с Mac OS X v10.3 журналирование стало включённым по умолчанию, а том с журналом получил название HFSJ. Файловая система, была разработана разработанная Apple Inc. для замены ранее использовавшейся HFS, основной файловой системы на компьютерах Macintosh. Ещё с этой файловой системой может работать плеер iPod. HFS+ можно рассматривать, как усовершенствованную версию HFS для расширения возможностей Mac OS. Во время разработки эта система называлась Sequoia. HFS+ является улучшенной версией HFS с поддержкой файлов большого размера (32-битная адресация вместо старой 16-битной) и использует кодировку UTF-16 для имён файлов и папок. HFS+ поддерживает имена длиной до 255 символов формата UTF-16 и многопоточные файлы, подобно NTFS (однако почти все программы используют только поток данных (англ. data fork) и поток с ресурсами (англ. resourcefork)). HFS+ также использует 32-битную таблицу привязки файла к месту на диске (англ. allocationmappingtable) вместо 16-битной в HFS. Старая адресация являлась серьёзным ограничением HFS, не позволявшим работать с томами объёмом более 65 536 блоков (по аналогии: сравните FAT16 и FAT-32. При объёме диска в 1 Гб размер кластера (блока) составлял 16 Кб – даже файл из 1 байта занимал все 16 Кб).

В 2014 году Доминик Джампаоло, инженер компании Apple, начал разрабатывать новую файловую систему Apple File System (APFS). AppleFileSystem оптимизирована для работы с флеш-памятью. С 2017 года используется по умолчанию в операционных системах macOS, iOS, watchOS, tvOS. APFS разработана для Flash/SSD-накопителей. Главная особенность файловой системы – шифрование файлов и чувствительных метаданных.

В 2001 году появилась ещё одна новая файловая система Lustre – распределённая файловая система массового параллелизма, используемая обычно для крупномасштабных кластерных вычислений. Название Lustre является контаминацией, образованной словами Linux и cluster. Реализованный под лицензией GNU GPL, проект предоставляет высокопроизводительную файловую систему для кластеров с десятками тысяч узлов сети и петабайтными хранилищами информации. Файловые системы Lustre используются в компьютерных кластерах, начиная от небольших кластеров рабочих групп и заканчивая масштабными географически распределёнными кластерами. Пятнадцать суперкомпьютеров из

мирового «Топ-30» используют файловые системы Lustre, в том числе самый быстрый в мире суперкомпьютер – K computer. Файловые системы Lustre могут поддерживать десятки тысяч клиентских систем, десятки петабайт (PBs) памяти для хранения данных и пропускную способность ввода-вывода в сотни гигабайт в секунду (GB/s). Благодаря высокой масштабируемости Lustre, такие области бизнеса, как провайдеры Интернет, финансовые организации, индустрия нефти и газа устанавливают файловые системы Lustre в своих центрах обработки данных.

2.3. Файловая система FAT

Большинство существующих на сегодняшний день файловых систем построены на основе таблицы размещения файлов (FileAllocationTable – FAT), которая содержит дорожки данных в каждом кластере на диске. Существует несколько типов файловой системы FAT – FAT 12, FAT 16 и FAT 32. Они отличаются количеством бит, используемых в таблице размещения файлов. Другими словами, в FAT 32 используется 32 разряда для хранения дорожки данных в каждом кластере, в FAT 16 – 16 разрядов и т.д. В настоящее время существуют следующие типы файловой системы FAT:

- FAT12, используется на дисках емкостью не более 16 Мбайт;
- FAT16, используется в разделах емкостью от 16 Мбайт до 2Гбайт;
- FAT32, используется (необязательно) в разделах от 512 Мбайт до 2 Тбайт.

На основе FAT была разработана новая файловая система exFAT (extendedFAT), используемая преимущественно для флеш-накопителей.

Файловая система FAT заполняет свободное место на диске последовательно от начала к концу. При создании нового файла или увеличении уже существующего она ищет первый свободный кластер в таблице размещения файлов. Если одни файлы были удалены, а другие изменились в размере, то появляющиеся в результате пустые кластеры будут рассеяны по диску. Если кластеры, содержащие данные файла, расположены не подряд, то файл оказывается фрагментированным. Сильно фрагментированные файлы значительно снижают эффективность работы, так как головки чтения/записи при поиске очередной записи файла должны будут перемещаться от одной области диска к другой. Желательно, чтобы кластеры, выделенные для хранения файла, шли подряд, так как это позволяет сократить время его поиска. Однако, это можно сделать только с помощью специальной программы, подобная процедура получила название дефрагментации файла.

Недостатком FAT также является то, что ее производительность зависит от количества файлов, находящихся в одном каталоге. При большом количестве файлов (около тысячи), выполнение операции считывания списка файлов

в каталоге может занять несколько минут. FAT не предусматривает хранения такой информации, как сведения о владельце или полномочия доступа к файлу.

FAT – простая файловая система, не предотвращающая порчи файлов из-за ненормального завершения работы компьютера, она является одной из самых распространенных файловых систем, и ее поддерживают большинство ОС.

Весь диск делится на сектора. Размер сектора равен 512 байт. Поскольку размер FAT-таблицы ограничен, то для дисков, размер которых превышает 32 Мбайт, обеспечить адресацию к каждому отдельному сектору не представляется возможным. В связи с этим группы секторов условно объединяются в кластеры. Кластер является наименьшей единицей адресации к данным.

Размер кластера:

- в отличие от размера сектора, не фиксирован и зависит от емкости диска;
- количество секторов в кластере равно степени двойки;
- определяется автоматически, при форматировании носителя информации;
- для хранения данных файла отводится целое число кластеров (минимум один).

Использование кластеров большой длины имеет положительные стороны:

- уменьшает фрагментирование диска;
- уменьшается размер FAT – таблицы, что увеличивает быстродействие.

Отрицательные стороны увеличения размеров кластеров: неэффективное использование пространства диска, при наличии большого числа файлов небольшой длины, поскольку любой файл (даже очень маленький) полностью оккупировывает весь кластер. Даже если файл достаточно велик и располагается в нескольких кластерах, все равно в его конце образуется некий остаток, нерационально расходующий целый кластер.

Для современных жестких дисков потери, связанные с неэффективностью файловой системы, весьма значительны и могут составлять от 25% до 40% полной емкости диска, в зависимости от среднего размера хранящихся файлов.

Для организации доступа к файлу операционная система должна иметь сведения о номерах кластеров, где размещается каждый файл. В этом ей помогает FAT-таблица, которая предназначена для размещения и поиска файлов на диске, содержит информацию о свободном пространстве на диске, о неисправных секторах, а также код формата диска.

Принцип организации файловой системы – табличный. Данные о том, в каком месте диска записан тот или иной файл, хранятся в системной области диска в специальных таблицах размещения файлов (FAT-таблицах).

Количество ячеек FAT-таблицы (рисунок 2.2) определяется количеством кластеров на диске. Каждая ячейка может содержать номер кластера, хранящего информацию.

Поскольку нарушение FAT-таблицы приводит к невозможности воспользоваться данными, записанными на диске, к ней предъявляются особые требования надежности, и она существует, как минимум, в двух экземплярах, идентичность которых регулярно контролируется средствами операционной системы.

Disk Editor							
Object	Edit	Link	View	Info	Tools	Help	
Sector 1							
9	10	11	4	5	6	<EOF>	8
<EOF>	<EOF>	<EOF>	<EOF>	<EOF>	<EOF>	<EOF>	<EOF>
<EOF>	<EOF>	<EOF>	745	<EOF>	<EOF>	<EOF>	<EOF>
61	<EOF>	35	36	37	46	<EOF>	<EOF>
<EOF>	<EOF>	<EOF>	<EOF>	<EOF>	<EOF>	50	<EOF>
<EOF>	<EOF>	51	52	53	54	58	56
57	<EOF>	59	60	63	888	<EOF>	64
65	69	67	68	<EOF>	70	71	<EOF>
73	74	75	76	77	<EOF>	79	80
81	82	83	84	85	86	87	88
89	90	91	92	93	94	100	96
<EOF>	98	99	<EOF>	101	102	103	104
105	106	107	108	109	110	111	112
113	<EOF>	<EOF>	116	117	118	119	120
121	122	123	124	125	126	127	128
129	130	131	132	133	134	135	136
<EOF>	<EOF>	139	140	141	142	143	144
<EOF>	146	147	148	149	150	151	152
153	154	155	156	157	158	159	160
FAT (1st Copy) C:\IO.DOS							Sector 1 Cluster 2, hex 2

Рис. 2.2. FAT таблица

2.3.1. Каталоги

Для каждого файла и каталога на диске имеется один элемент в определенном каталоге. Каталог – это база данных, содержащая информацию о записанных на диске файлах. Каждая запись в ней имеет длину 32 байт, и между записями не должно быть никаких разделителей. В каталоге сохраняется практически вся информация о файле, которой располагает операционная система.

Существует два основных типа каталогов: корневой каталог и подкаталог. Различаются они максимальным количеством хранящихся файлов. На каждом логическом диске в фиксированном месте, сразу же за копиями FAT, располагается корневой каталог. Размеры корневых каталогов варьируются в зависимости от размера диска, но каждый конкретный корневой каталог имеет фиксированное максимальное число файлов. Длина корневого каталога фиксируется при создании логического диска и не может быть изменена в процессе работы. В отличие от корневого каталога, подкаталог может хранить произвольное количество файлов и расширяться по мере необходимости.

Один элемент корневого каталога выделяется для метки диска. Для каждого каталога есть один элемент в его родительском каталоге. Каждый каталог за исключением корневого, содержит по одному элементу для специальных имен «.» и «..». Эти элементы указывают начало цепочки в FAT соответственно для самого каталога и для его родительского каталога. Такая система описания каталогов обеспечивает возможность сокращенного написания пути к данному файлу, когда он проходит через родительский каталог.

Одно из преимуществ FAT 32 заключается в том, что корневой каталог может располагаться в любом месте диска и содержать неограниченное количество записей.

Все каталоги имеют одинаковую структуру. Записи в этой базе данных сохраняют важную информацию о файлах, которая связана с информацией, хранящейся в FAT, посредством одного из полей записи – номера первого занимаемого файлом кластера на диске. Если бы все файлы на диске не превышали размеров одного кластера, потребности в FAT вообще бы не возникло. В FAT содержится информация о файле, отсутствующая в каталоге, – номера кластеров, в которых расположен весь файл.

Чтобы отследить расположение всего файла на диске, обратитесь к каталогу и выясните номер первого кластера и длину файла. Затем, используя таблицу размещения файлов, просмотрите цепочку кластеров, занимаемых файлом, пока не дойдете до конца файла.

Каждый элемент каталога имеет длину 32 байта и структуру представленную в таблице 2.1.

Таблица 2.1

Структура дескриптора файла или каталога

Hex	Dec	Длина, (байт)	FAT
00h	0	8	Имя файла, каталога или тома
08h	8	3	Расширение имени файла
0Bh	11	1	Байт атрибутов
0Ch	12	10	Резервное поле
16h	22	2	Код времени обновления файла
18h	24	2	Код даты изменения файла
1Ah	26	2	Номер первого кластера файла
1Ch	28	4	Размер файла

Поле «Имя файла, каталога или тома»:если имя содержит меньше 8 символов, то справа оно дополняется пустыми позициями.

Код 00 в первом байте поля имени показывает, что этот элемент каталога ранее не использовался. Так как каталог заполняется последовательно, это

означает, что и следующие за ним элементы не использовались. Это позволяет избежать лишнего поиска в каталоге. При удалении файла в первом байте соответствующего элемента каталога записывается код E5, все остальные байты элемента не изменяются. Сохраняемая в каталоге и FAT информация после удаления файла позволяет выполнять его восстановление, если занимаемое им ранее дисковое пространство не было выделено другому файлу.

Код 2E (символ «.») в первом байте показывает, что элемент описывает сам каталог. Если и второй байт содержит код 2E, то элемент описывает родительский каталог («..»).

Поле «Расширение имени файла»: если этот элемент описывает файл, то поле может быть пустым. В противном случае это поле используется, когда в элементе корневого каталога указывается метка тома.

Поле «Байт атрибутов»: каждый бит этого поля задает определенный атрибут, указанный в таблице 2.2.

Таблица 2.2

Назначение битов байта атрибутов

7 6 5 4 3 2 1 0	ЗНАЧЕНИЕ	НАЗНАЧЕНИЕ
. 1	1h	защищенный
. 1 .	2h	скрытый
. 1 . .	4h	системный
. . . . 1 . . .	8h	метка тома
. . . 1	10h	каталог
. . 1	20h	архивный
. 0	40h	не используется
0	80h	не используется

Поле «Код времени обновления файла»: содержимое этого поля рассматривается как целое число без знака, полученное по следующей формуле:

$$\text{часы} * 2048 + \text{минуты} * 32 + \text{секунды} \setminus 2.$$

Для выполнения обратного преобразования следует разделить содержимое поля на 2048, частное от деления даст нам часы. Деление остатка на 32 даст нам минуты, а полученный остаток при умножении его на 2 дает секунды.

Поле «Код даты изменения файла»: содержимое его рассматривается как целое число без знака, полученное по следующей формуле:

$$(\text{год}-1980) * 512 + \text{месяц} * 32 + \text{день}.$$

По этой формуле календарь поддерживается с 1980г. по 2108г. (из-за ограниченности ширины поля даты).

Поле «Номер первого кластера файла»: содержимое этого поля одновременно служит указателем к первому кластеру файла в поле данных и к первому

элементу в цепочке FAT. Для файлов, которым не выделено место на диске и для метки тома это поле содержит код 0000(h).

Для полноценного обслуживания файловой системы, операционная система должна уметь выполнять следующие действия:

- создание файлов;
- создание каталогов (папок);
- переименование файлов и каталогов (папок);
- копирование и перемещение файлов между различными дисками компьютера, а так же между каталогами одного диска;
- удаление файлов и каталогов;
- навигация по структуре файлов и папок для доступа к необходимым файлам;
- управление атрибутами файла.

2.3.2. Таблица размещения файлов FAT

Таблица размещения файлов (FAT) содержит номера кластеров, в которых расположены файлы на диске. Каждому кластеру в FAT соответствует одно число. Секторы, не содержащие пользовательских данных (файлов), не отражены в FAT. К таким секторам относятся загрузочные секторы, таблицы размещения файлов, а в FAT16 ещё и секторы корневого каталога.

FAT – это электронная таблица, управляющая распределением дискового пространства. Каждая ячейка этой таблицы связана с определенным кластером на диске. Число, содержащееся в этой ячейке, сообщает о том, использован ли данный кластер под какой-либо файл и, если использован, где находится следующий кластер этого файла.

Каждая ячейка FAT хранит шестнадцатиричное значение длиной 12, 16 или 32 бита. Шестнадцатирядные FAT более удобны в работе, так как значительно легче редактировать поля размером в два байта, чем в полтора. Чтобы самостоятельно отредактировать FAT, нужно выполнить некоторые математические преобразования для получения номера кластера. К счастью, многие программы позволяют отредактировать FAT автоматически. В большинстве этих программ номера кластеров представлены в десятичном виде, наиболее удобном для пользователей. На рисунке 2.2.1 приведены данные о каталоге и FAT (файл не фрагментирован)

Записи о нефрагментированном файле в каталоге и FAT		

Каталог		

Имя	Начальный кластер	Размер

Usconst.txt	1000	4

FAT 16		

Номер кластера	Значение	Назначение

00002	Первый доступный кластер	
00999	0 0	Кластер доступен
1 1 00		
01000	1001	Используется; ссылка на следующий кластер
01001	1002	Используется; ссылка на следующий кластер
01002	1003	Используется; ссылка на следующий кластер
01003	FFFFh	Конец файла
01004	0	Кластер доступен
01005	0	Кластер доступен
65526	0	Последний доступный кластер

Рис. 2.2.1. Файл Usconst в каталоге FAT

В данном примере запись каталога указывает начальный кластер (1000), в котором размещается файл. В FAT кластеры с ненулевыми значениями используются, а специальное значение указывает дальнейшее расположение файла. В рассматриваемом примере в кластере 1000 указывается кластер 1001, в 1001–1002, в 1002–1003, а в 1003 записано значение FFFFh, т.е. на этом кластере файл заканчивается.

Рассмотрим пример с фрагментированным файлом. Пусть файл Usconst.txt записан, начиная с кластера номер 1000. А файл Pledge.txt начинается с кластера 1002. Таким образом, файл Usconst.txt становится фрагментированным. Описанная ситуация показана на рисунке 2.3.

Записи о фрагментированном файле в каталоге и FAT

Каталог

Имя	Начальный кластер	Размер
Pledge.txt	1002	2
Usconst.txt	1000	4

FAT 16

Номер кластера	Значение	Назначение
00002	0	Первый доступный кластер
00999	0	Кластер доступен
01000	1001	Используется; ссылка на следующий кластер
01001	1004	Используется; ссылка на следующий кластер
01002	1003	Используется; ссылка на следующий кластер
01003	FFFFh	Конец файла
01004	1005	Используется; ссылка на следующий кластер
01005	FFFFh	Конец файла
65526	0	Последний доступный кластер

Рис. 2.3. Фрагментация файла на диске FAT

В данном примере в файл Usconst.txt «внедряется» файл Pledge.txt, что приводит к непоследовательному расположению файлов на диске, т.е. фрагментации. В операционных системах DOS и Windows есть программы дефрагментации, которые перемещают файлы для их последовательного размещения на диске.

Первые две записи FAT зарезервированы и содержат информацию о самой FAT, все остальные указывают на соответствующие кластеры диска. Большинство записей FAT состоит из ссылок на кластеры, в которых содержатся части определенного файла, а некоторые содержат специальные шестнадцатиричные значения:

- 0000h – кластер не используется;
- FFF7h – как минимум один секторов в кластере поврежден и не может быть использован для хранения данных;
- FFF8h-FFFFh – кластер содержит конец файла.

Обычно на диске создается на две копии FAT. Каждая копия занимает несколько последовательных секторов на диске, и вторая копия записывается непосредственно после первой. К сожалению, операционная система использует вторую копию FAT только в том случае, когда невозможно прочитать секторы, содержащие первую копию. Таким образом, если первая копия FAT пропадет (весьма распространенная ситуация), операционная система не будет использовать вторую копию. Кроме того, каждый раз, когда операционная система обновляет первую копию FAT, большие участки первой копии автоматически копируются во вторую. Если же первая копия повреждена, то и вторая окажется поврежденной: после обновления FAT вторая копия отражает все изме-

нения в первой копии, включая и ошибки. Обе копии FAT редко отличаются одна от другой, по крайней мере в течение продолжительного срока: при обновлении первая копия FAT автоматически копируется во вторую. Учитывая все это, можно сказать, что применение второй копии FAT ограничивается только операциями по восстановлению дефектных данных. Но даже в такой ситуации использовать вторую копию FAT можно только в том случае, когда проблема решается немедленно, не дожидаясь очередного обновления FAT.

2.3.3. Кластеры FAT

Кластер является наименьшей ячейкой на диске, которой может оперировать система при чтении или записи файла на диск. Кластер соответствует одному или (чаще всего) нескольким секторам. Это позволяет уменьшить размер FAT и ускорить работу операционной системы, так как ей приходится оперировать меньшим числом распределяемых ячеек. В то же время с увеличением размера кластера на диске растет и размер неиспользуемого дискового пространства, так как его распределение происходит с дискретностью в один кластер.

Довольно странным является то обстоятельство, что некоторые дискеты высокой плотности имеют меньший размер кластера, чем дискеты низкой плотности. Увеличивается размер FAT, увеличивается количество записей, которые должна обрабатывать операционная система, и замедляется работа самой системы. Меньший размер кластера позволяет уменьшить размер неиспользуемого дискового пространства. Все пространство между концом файла и концом последнего занимаемого кластера не используется, и в результате, чем больше размер кластера, тем больше потери дискового пространства. Кроме того, диски высокой плотности работают быстрее, чем диски низкой плотности.

Для жестких дисков размер кластера может варьироваться в зависимости от размера раздела диска. На рисунке 2.4. приведены размеры кластеров в зависимости от размера логического диска.

Стандартные размеры кластеров

Размер диска, Мбайт	Размер кластера	Тип FAT
Меньше 16	8 секторов (4096 байт)	12-разрядная
16-128	4 сектора (2048 байт)	16-разрядная
128-256	8 секторов (4096 байт)	16-разрядная
256-512	16 секторов (8192 байт)	16-разрядная
512-1024	32 сектора (16384 байт)	16-разрядная
1024-2048 и более	64 сектора (32768 байт)	16-разрядная

Рис. 2.4. Стандартные размеры кластеров

Использование кластеров больших размеров ощутимо сказывается на работе системы. Например, на диске емкостью 2 Гбайт, содержащем 5000 файлов,

со средней потерей дискового пространства в полкластера на один файл суммарные потери дискового пространства составят около 78 Мбайт $[5000 \cdot (0,5 \cdot 32)]$.

Размер кластера и структура FAT определяют максимально возможный размер раздела. Поскольку FAT использует записи размером 16 байт для ссылки на кластер в разделе, максимально возможное число кластеров может равняться 65536 (2¹⁶). Максимальный размер кластера – 32 Кбайт, следовательно, максимально возможный размер раздела – 2047,6875 Мбайт.

2.3.4. FAT32

Когда разрабатывалась FAT, жесткие диски размером 2 Гбайт можно было встретить разве что в научно-фантастических романах. Современные размеры дисков во много раз больше.

Для устранения такого ограничения Microsoft предложила новую файловую систему с расширенными возможностями, называемую FAT32. Эта файловая система работает как стандартная FAT, но имеет отличия в организации хранения файлов. Система FAT32 была впервые реализована в Windows 95 OEMServiceRelease 2 (OSR2). Она встроена также и во все последующие версии Windows.

Основное преимущество FAT32 – это возможность использования 32-разрядных записей, вместо 16-разрядных, что приводит к увеличению числа кластеров в разделе до 268435456 (вместо 65536, или 2¹⁶). Это значение эквивалентно 2²⁸, а не 2³², поскольку четыре бита из 32 зарезервированы для других целей.

При использовании FAT32 размер раздела может достигать 2 Тбайт (1 Тбайт равен 1024 Мбайт). Новая файловая система может иметь 4294967296 (2³²) кластеров размером 512 байт, а размер единичного файла может составлять 4 Гбайт.

Существует еще одно отличие FAT32 от ее предшественниц – положение корневого каталога: он не занимает фиксированного места на диске, как в FAT16. Корневой каталог в FAT32 может располагаться в любом месте раздела и иметь любой размер. Устранение ограничений записей корневого каталога обеспечивает динамическое изменение размера раздела FAT32. Однако Microsoft не реализовала это замечательное свойство в операционных системах Windows 9x, чем и воспользовались независимые разработчики, такие как фирма PowerQuest, создавшая программу PartitionMagic.

Поскольку раздел FAT32 имеет больше кластеров, чем раздел FAT16, размер кластера уменьшается. Использование меньшего кластера снижает потери дискового пространства. Например, раздел размером 2 Гбайт с 5000 файлов в FAT32 использует кластер размером 4 Кбайт, вместо 32 Кбайт в FAT 16. Такое уменьшение размера кластера снижает потери дискового пространства с 78 до 10 Мбайт.

Для сравнения FAT 16 и FAT 32 необходимо посмотреть, как в этих файловых системах организовано хранение данных. Номера кластеров в FAT 16 хранятся в виде 16-разрядных записей (0000h-FFFFh). Максимальное значение FFFFh соответствует десятичному 65536, но несколько значений зарезервированы для специальных целей. Реальное число кластеров в FAT16 лежит в диапазоне 0002h-FFF6h, или 2-65526. Таким образом, для хранения файлов используется 65524 кластера. Типичная запись о файле в FAT 16 представлена на рисунке 2.5.

В FAT 32 количество кластеров лежит в диапазоне 00000000h-FFFFFFFFh, или 0-4294967295. Как и в FAT 16 верхние и нижние кластеры зарезервированы для специальных целей и их номера лежат в диапазоне 00000002h-FFFFFFFF6h, или 2-4294967286. Таким образом, для хранения файлов можно использовать 4 294 967 284 кластера (рис. 2.6).

Записи файлов в файловой системе FAT 16		

Каталог		

Имя	Начальный кластер	Размер

Usconst.txt	1000	4

FAT 16		

Номер кластера	Значение	Назначение

00002	0	Первый доступный кластер
00999	0	Кластер доступен
01000	1001	Используется; ссылка на следующий кластер
01001	1002	Используется; ссылка на следующий кластер
01002	1003	Используется; ссылка на следующий кластер
01003	FFFFh	Конец файла
01004	0	Кластер доступен
65526	0	Последний доступный кластер

Рис. 2.5. Записи в файловой системе FAT16

Накопитель на жестких дисках разбит на большее количество кластеров, каждый из которых становится меньше, что снижает потери дискового пространства. Пример записей о файле в FAT 32 приведены на рисунке 2.6.

Записи файлов в системе FAT 32		

Каталог		

Имя	Начальный кластер	Размер

Usconst.txt	1000	8

FAT 32		

Номер кластера	Значение	Назначение

0000000002	0	Первый доступный кластер
0000000999	0	Кластер доступен
0000001000	1001	Используется; ссылка на следующий кластер
0000001001	1002	Используется; ссылка на следующий кластер
0000001002	1003	Используется; ссылка на следующий кластер
0000001003	1004	Используется; ссылка на следующий кластер
0000001004	1005	Используется; ссылка на следующий кластер
0000001005	1006	Используется; ссылка на следующий кластер
0000001006	1007	Используется; ссылка на следующий кластер
0000001007	FFFFFFFFh	Конец файла
0000001008	0	Кластер доступен
4294967286	0	Последний доступный кластер

Рис. 2.6. Записи в файловой системе FAT32

Уменьшение размера кластера приводит к увеличению записей в FAT. Раздел размером 2 Гбайт с FAT32 использует 524288 записей, в то время как аналогичный раздел с FAT16 использует 65536 записей. Следовательно, таблица FAT16 имеет размер 128 Кбайт (65536 записей * 16 бит = 1048576 бит, или 131072 байт, или 128 Кбайт), а таблица FAT32 – 2 Мбайт.

Размер FAT существенно влияет на производительность файловой системы. В Windows 9x модуль VCACHE пытается загрузить FAT в оперативную память для повышения производительности системы. Выбор кластера размером 4 Кбайт на дисках емкостью до 8 Гбайт обеспечивает компромисс между производительностью и размером FAT в оперативной памяти.

Несмотря на то что размер FAT в файловой системе FAT32 практически в двадцать раз больше, чем в FAT16, появляется незначительный (менее 15%) прирост производительности FAT32 в современных операционных системах. Это отчасти достигается использованием в персональных компьютерах самых современных накопителей на жестких дисках.

2.3.5. Ошибки файловой системы FAT

Ошибки в файловой системе появляются скорее из-за программных, нежели из-за аппаратных сбоев (например, при неверном завершении работы Windows). Наиболее часто встречаются следующие ошибки.

Потерянные кластеры – это наиболее распространенная ошибка файловой системы, при которой кластеры в FAT помечаются как используемые, хотя на самом деле таковыми не являются. Эти потерянные кластеры появляются при неверном завершении работы приложения или крахе системы. Программы восстановления диска могут обнаружить эти кластеры и восстановить их (рисунок 2.7.).

Появляются цепочки кластеров, не имеющие записей в каталоге. Чаще всего это происходит при «зависании» программы при операции открытия файла. Программы восстановления диска просматривают диск и создают копию FAT в оперативной памяти.

Потерянные кластеры в файловой структуре		

Каталог		

Имя	Начальный кластер	Размер

(Нет записей)	0	0

FAT 16		

Номер кластера	Значение	Назначение

00002	0	Первый доступный кластер
00999	0	Кластер доступен
01000	1001	Используется; ссылка на следующий кластер
01001	1002	Используется; ссылка на следующий кластер
01002	1003	Используется; ссылка на следующий кластер
01003	FFFFh	Конец файла
01004	0	Кластер доступен
65526	0	Последний доступный кластер

Рис. 2.7. Потерянные кластеры в файловой структуре

Затем эта копия сравнивается с «настоящей» FAT и таким образом выявляются потерянные кластеры, т.е. не принадлежащие ни одному из существующих файлов. Практически все программы восстановления могут сохранять информацию из потерянных кластеров в файл, а затем обнулять их.

Например, программа Chkdsk из цепочек потерянных кластеров создает файлы с именами FILE0001.CHK, FILE0002.CHK и т.д. Программа Chkdsk преобразует потерянные кластеры в файлы так, как показано на рисунке 2.8.

Потерянные кластеры найдены

Каталог

Имя Начальный кластер Размер

FILE0001.CHK 1000 4

FAT 16

Номер кластера Значение Назначение

00002 0 Первый доступный кластер
00999 0 Кластер доступен
01000 1001 Используется; ссылка на следующий кластер
01001 1002 Используется; ссылка на следующий кластер
01002 1003 Используется; ссылка на следующий кластер
01003 FFFFh Конец файла
01004 0 Кластер доступен
65526 0 Последний доступный кластер

Рис. 2.8. Восстановление потерянных кластеров

Как видно из приведенного примера, оригинальное имя файла не восстанавливается. Однако его можно восстановить, просмотрев содержимое файлов, которые созданы программой восстановления диска.

2.3.6. Создание и именование файлов

Файл – это поименованная область внешней памяти произвольной длины.

Поскольку из этого определения вытекает, что файл может иметь нулевую длину, то фактически создание файла состоит в присвоении ему имени и регистрации его в файловой системе (создание элемента каталога) – это одна из функций операционной системы. Даже когда мы создаем файл, работая в какой-то прикладной программе, в общем случае для этой операции привлекаются средства операционной системы.

При создании файла файловая система ОС выделяет один элемент в таблице каталога, куда записываются свойства файла, определяет количество кластеров (секторов), необходимых для размещения файла. Затем файловая система в FAT таблице находит первый свободный кластер и записывает его номер в элемент каталога, а в ячейку FAT таблицы, соответствующую первому свободному кластеру записывает номер следующего свободного кластера (сектора) и т.д. пока не будет определен последний, необходимый для размещения кластер,

в который ОС поставит метку FFFFh (последний) и только после этого ОС разместит данные в области данных по выбранным секторам.

Каждый элемент каталога имеет длину 32 байта и состоит из восьми полей (таблица 2.1.). Для Fat32 элемент каталога отличается длиной номера начального кластера, поэтому структура его элементов имеет несколько другой вид и представлена в таблице 2.3.

Первый байт поля имени используется для обозначения трех специальных случаев:

1) значение 00H в первом байте показывает, что этот элемент каталога никогда не был использован. Так как каталог заполняется последовательно, это означает, что и следующие за ним элементы не были использованы;

2) при удалении файла DOS записывает E5H в первом байте соответствующего элемента каталога. Все остальные баты элемента остаются без изменений. Элемент становится свободным, но после него могут быть другие активные элементы. При удалении файла DOS освобождает и все элементы соответствующей цепочки в FAT таблице. Но сам файл продолжает существовать на диске, что позволяет соответствующим программам восстановить удаленный файл, если элемент каталог или кластеры файла в области данных не используется другим файлом;

3) значение 2EH (символ «.») в первом байте показывает, что этот элемент служит для описания самого каталога. Если во второй байт содержит 2EH, элемент описывает родительский каталог («..»).

Таблица 2.3

Элементы каталога FAT32

Смещение	Размер	Значение
0	8	Имя файла в кодах ASCII
8	3	Расширение имени файла в кодах ASCII
11	1	Атрибут
12	8	Зарезервировано
20	2	Номер начального кластера (старшие разряды)
22	2	Время создания или последней модификации
24	2	Дата создания или последней модификации
26	2	Номер начального кластера (младшие разряды)
28	4	Фактическая длина файла в байтах

Для файлов, которым не выделено места, и для метки тома, поле номера кластера содержит 0000H. А байты атрибута определяют каждый файл как системный, скрытый и т.д. Значение битов байта атрибута приведены в таблице 2.2.

По способам именования файлов различают «короткое» и «длинное» имя. До появления операционной системы Windows 95 общепринятым способом именования файлов было соглашение 8.3. Согласно этому соглашению, приня-

тому в MS-DOS, имя файла состоит из двух частей: собственно имени и расширения имени.

На имя файла отводится 8 символов, а на его расширение – 3 символа. Имя от расширения отделяется точкой. Как имя, так и расширение могут включать только алфавитно-цифровые символы латинского алфавита и символ подчеркивания, имя не может начинаться с цифры.

Соглашение 8.3 не является стандартом, и потому в ряде случаев отклонения от правильной формы записи допускаются как операционной системой, так и ее приложениями. Сегодня имена файлов, записанные в соответствии с соглашением 8.3, считаются «короткими».

Основным недостатком «коротких» имен является их низкая содержательность. С появлением операционной системы Windows 95 было введено понятие «длинного» имени оно может содержать до 256 символов.

«Длинное» имя может содержать любые символы, кроме девяти специальных: \ / | : * ? " < > . В имени разрешается использовать пробелы и несколько точек. Расширением имени считаются все символы, идущие после последней точки.

2.3.7. Особенности длинных имен в ОС Windows

Требования совместимости, которым должна удовлетворять система Windows, означают, что невозможно просто изменить существующий формат хранения данных на диске, который применяем в FAT файловой системе.

VFAT файловая система поддерживает как длинные, так и короткие имена файла. 32-байтный элемент каталога идентичен тому формату, который поддерживают версии MS-DOS.

Метод работы с длинными именами файлов строится на использовании байта атрибута элемента для короткого имени файла. Установка младших четырех битов этого байта (значение 0Fh) задает элементу каталога атрибуты «только для чтения», «скрытый», «системный» и «метка тома». Добавление атрибута «метка тома» дает не имеющее смысл сочетание и защищает элемент каталога от изменения.

Для того чтобы исключить возможность, что какая-нибудь утилита обслуживания диска уничтожит данные, разработчики ввели в API функцию «исключительного закрепления метки тома» (exclusivevolumelock), которую приложение должно вызвать прежде, чем Windows позволит ему осуществить прямую запись на диск (при помощи MS-DOS прерываний Int 13h и Int 26h). Windows использует для формирования длинного имени несколько последовательных коротких имен – элементов каталога (рисунок 2.9.), защищая каждое из 32-байтных имен при помощи атрибута 0Fh.

В пределах одного кластера, в котором хранится информация о содержимом каталога, элемент с длинным именем файла располагается в соответствии с форматом, который показан на рисунке 3.5. Длинное имя файла не может существовать без связанного с ним элемента с коротким именем. Если имеет место такая ситуация, значит, нарушена целостность данных на диске.

Последняя из частей элемента, содержащего длинное имя файла
.
.
.
... добавочные части элемента, содержащего длинное имя файла
Вторая часть элемента, содержащего длинное имя файла
Первая часть элемента, содержащего длинное имя файла
Связанный с этим именем элемент, содержащий короткое имя файла

Рис. 2.9. Формирование длинного имени в каталоге из элементов коротких имен

Каждый 32-байтный элемент, описывающий длинное имя файла, содержит порядковый номер, защитный байт атрибута, значение типа и контрольную сумму (рисунок 2.10.).

Порядковый номер помогает Windows узнать о непоследовательном или некорректном изменении структуры каталога. Поле типа идентифицирует элемент как LONG_NAME_COMP (элемент длинного имени) либо как LONG_CLASS (32-байтный элемент, который содержит информацию о классе для данного файла). Если элемент представляет собой часть имени, то большая часть 32 байт используется для хранения символов имени. Если это элемент, описывающий класс, то в нем хранится информация о классе. Система хранит длинные имена файлов в виде символов таблицы Unicode, т.е. каждый символ имени файла требует 16 бит. Контрольная сумма вычисляется по связанному с данным файлом короткому имени. Если короткое имя изменится вне среды Windows, то Windows поймет, что элементы длинного имени больше не имеют смысла.

Порядковый номер	СИМВОЛЫ ИМЕНИ ФАЙЛА										Атрибуты	Тип	Контрольная сумма	Имя (продолжение)
	Имя (продолжение)										0	Имя (продолжение)		

Рис. 2.10. Формат элемента каталога, в котором хранится длинное имя файла

Основная проблема при формировании короткого имени, связанного с длинным, состоит в создании уникального короткого имени, которое не совпадает с каким-нибудь уже существующим коротким именем.

Использование «длинных» имен файлов в операционных системах Windows имеет ряд особенностей.

1. Если «длинное» имя файла включает пробелы, то в служебных операциях его надо заключать в кавычки. Рекомендуется не использовать пробелы, а заменять их символами подчеркивания.

2. В корневой папке диска (на верхнем уровне иерархической файловой структуры) нежелательно хранить файлы с длинными именами – в отличие от прочих папок в ней ограничено количество единиц хранения, причем, чем длиннее имена, тем меньше файлов можно разместить в корневой папке.

3. Кроме ограничения на длину имени файла (256 символов) существует гораздо более жесткое ограничение на длину полного имени файла (в него входит путь доступа к файлу, начиная от вершины иерархической структуры). Полное имя не может быть длиннее 260 символов.

4. Разрешается использовать символы любых алфавитов.

5. Прописные и строчные буквы не различаются операционной системой. Для нее имена Письмо.txt и письмо.txt соответствуют одному и тому же файлу. Однако символы разных регистров исправно отображаются операционной си-

стемой, и, если для наглядности надо использовать прописные буквы, это можно делать.

6. Расширение имени файла рассказывает операционной системе, исполняющей программе или пользователю, к какому типу относятся данные, содержащиеся в файле, и о формате, в котором они записаны.

2.3.8. Создание каталогов (папок)

Каталог (папка) – специальная структура, содержащая информацию о группе файлов, каталогов, хранимых совместно на одном носителе. Все современные операционные системы позволяют создавать каталоги.

Правила присвоения имени каталогу ничем не отличаются от правил присвоения имени файлу, хотя негласно для каталогов не принято задавать расширения имен.

Каждому подкаталогу, как и файлу, в корневом каталоге соответствует 32-байтный элемент каталога, содержащий имя, его атрибуты (архивный, скрытый, системный и «только для чтения»), дату и время создания (или внесения в него последних изменений), а также прочую информацию. Для файловых систем FAT положение корневого каталога на разделе и его размер жестко зафиксированы.

Особенности ОС Windows. До появления операционной системы Windows 95 при описании иерархической файловой структуры использовался введенный выше термин каталог. С появлением этой системы был введен новый термин – папка.

В том, что касается обслуживания файловой структуры носителя данных, эти термины равнозначны: каждому каталогу файлов на диске соответствует одноименная папка операционной системы. Основное отличие понятий папка и каталог проявляется не в организации хранения файлов, а в организации хранения объектов иной природы. Так, например, в Windows существуют специальные папки, представляющие собой удобные логические структуры, которым не соответствует ни один каталог диска (Панель управления, Принтеры, Назначенные задания).

Основной (корневой) каталог – это начало древовидной структуры диска. Как и все остальные каталоги, создаваемые на диске, он содержит информацию о файлах, которые ему принадлежат. Наиболее существенная часть этой информации – имя файла и указатель к началу соответствующей цепочки элементов FAT. После формирования диска корневой каталог либо пуст, либо содержит системные файлы. Пользователь строит файловую структуру диска, добавляя к основному каталогу файлы или новые каталоги. С точки зрения файловой организации каталоги тоже являются файлами. Исключением является корне-

вой каталог, который находится на определенном месте на диске, имеет фиксированную длину и который нельзя удалить.

Корневой каталог диска FAT32 не имеет фиксированного расположения, как на дисках с FAT12 и FAT16. На дисках с FAT32, корневой каталог – обычная цепочка кластеров. Элемент в структуре BPBFAT32 содержит номер первого кластера в корневом каталоге, что позволяет корневому каталогу расти так, как это может быть необходимо в процессе работы.

Для каждого файла на диске имеется один элемент в определенном каталоге. Один элемент корневого каталога выделяется для метки диска. Для каждого каталога имеется элемент в его родительском каталоге. Кроме того, каждый каталог, за исключением корневого, содержит по одному элементу со специальными именами «.» и «..». Эти элементы указывает начало цепочки в FAT для текущего каталога и для его родительского каталога.

Когда программа отправляет запрос к операционной системе с требованием предоставить ей содержимое какого-либо файла, ОС просматривает его запись в каталоге, чтобы найти первый кластер этого файла. Затем она обращается к элементу FAT для данного кластера, чтобы найти следующий кластер в цепочке. Повторяя этот процесс, пока не обнаружит последний кластер файла, таким образом ОС определяет, какие кластеры принадлежат данному файлу. Таким образом, система может предоставить программе любую часть запрошенного файла.

2.3.9. Сравнительные характеристики FAT разных версий и NTFS

	FAT12	FAT16	FAT32	NTFS
Разработчик	Microsoft			
Полное название	File Allocation Table (Таблица размещения файлов)			new technology file system
	12 битная версия	16 битная версия	32 битная версия	
Представлена	1980 (Microsoft Disk BASIC)	Ноябрь 1987 (MS-DOS 3.31)	Август 1996 (Windows 95 OSR2)	Июль 1993 года (Windows NT 3.1)
Структуры				
Содержание директории	Таблица			
Размещение файлов	Линейный список			Bitmap
Сбойные блоки	Тегирование кластера			\$badclus
Ограничения				

Максимальный размер файла	32Мб	2Гб	4Гб	2 ⁶⁴ байт (16 ЭиБ) минус 1 КиБ
Максимальное количество кластеров	4084	65 524	258 435 445(2 ²⁸ -12)	4 294 967 295(2 ³² -1)
Максимальная длина имени файла	8.3 или 255 символов при использовании LFN			255 16-битовых слов в кодировке UTF-16
Максимальный размер тома	32 Мб	2Гб 4Гб (64Кб на кластер, но поддерживается не везде)	2Тб 8Тб(2Кб на сектор)	2 ⁶⁴ - 1 кластер
Возможности				
Сохранение даты	Создание, модификации, доступа			Создание, изменение, изменения согласно POSIX, доступ
Диапазон дат	1 января 1980 — 31 декабря 2107			1 января 1980 года — 28 мая 2056 года
Дополнительные данные	Изначально не поддерживаются			ADS (Alternate Data Streams)
Атрибуты файлов	Только для чтения, скрытый, системный, метки тома, подкаталог, архивный			Только для чтения, скрытый, системный, требует архивирования, не проиндексирован, недоступен, временный, архивный, зашифрованный
Разграничение прав доступа	Нет			ACL
Безопасность	Нет		Нет	Да (начиная с NT5.0 встроена возможность физически шифровать данные)
Сжатие	Нет		Нет	Да
Устойчивость к сбоям	Средняя		Плохая (средства оптимизации по скорости привели к появлению слабых по надежности мест	Полная — автоматическое восстановление системы при любых сбоях (не считая физические ошибки записи, когда пишется одно, а на самом деле записывается другое)
Экономичность	Минимальная (огромные размеры кластеров на больших дисках)		Улучшена за счет уменьшения размеров кластеров	Максимальна. Очень эффективная и разнообразная система хранения данных

<p>Быстродействие</p>	<p>Высокое для малого числа файлов, но быстро уменьшается с появлением большого количества файлов в каталогах. результат — для слабо заполненных дисков — максимальное, для заполненных — плохое</p>	<p>Полностью аналогично FAT, но на дисках большого размера (десятки гигабайт) начинаются серьезные проблемы с общей организацией данных</p>	<p>Система не очень эффективна для малых и простых разделов (до 1 Гбайт), но работа с огромными массивами данных и внушительными каталогами организована так, нельзя более эффективно и очень сильно превосходит по скорости другие системы</p>
------------------------------	--	---	---

2.2.10. Исследование FAT32 с помощью программы WinHEX

Для анализа структуры и данных FAT32 дисков часто используется программа WinHEX, главное окно которой представлено на рисунке 2.11.

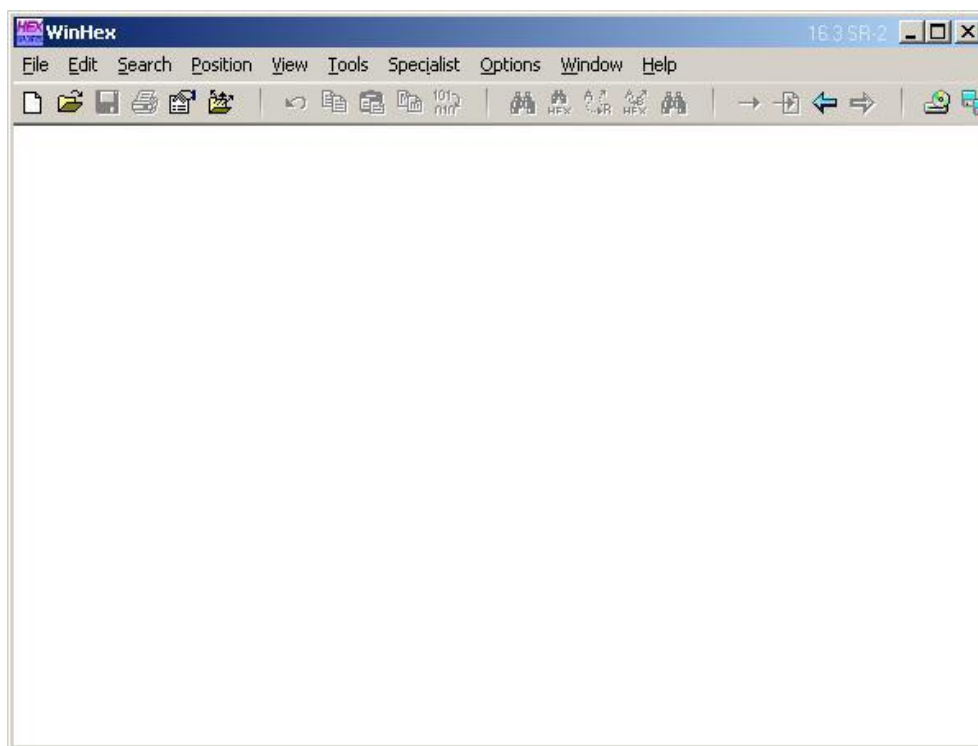


Рис. 2.11. Интерфейс главного окна программы WinHEX и окно выбора диска

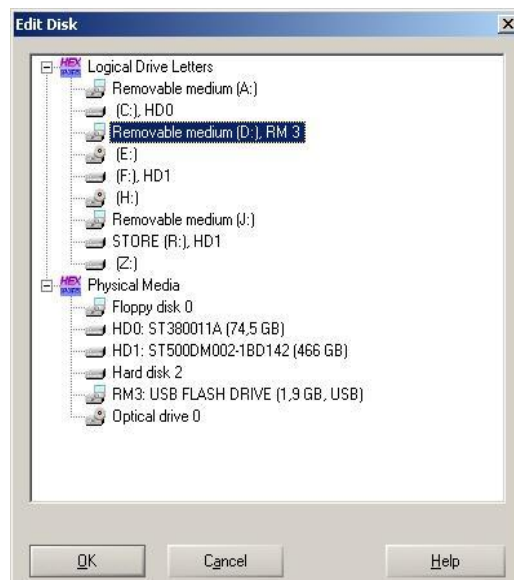


Рис. 2.12. Структура файловой системы

Открываем предназначенный для исследования диск (рисунок 2.11.) структура которого представлена на рисунке 2.12.

Нулевой сектор, называемый также загрузочным (Boot), содержит таблицу с параметрами диска и начальный загрузчик ОС. Первые 3 байта сектора (рисунок 2.13.) содержат команду перехода JMP на начало загрузчика: либо байт 0E9h и 1 байт короткого смещения, после которых следует команда NOP (код 90h), либо байт 0EBh и два байта длинного смещения, которое используется, если загрузчик расположен в зарезервированной области.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	EB	3C	90	4D	53	44	4F	53	35	2E	30	00	02	40	02	00	л<тMSDOS5.0 @
00000010	02	00	02	00	00	F8	EF	00	3F	00	FF	00	80	1F	00	00	шп ? я Ъ
00000020	80	B0	3B	00	00	00	29	23	18	C7	50	4E	4F	20	4E	41	Ъ°;)# ЗРНО N
00000030	4D	45	20	20	20	20	46	41	54	31	36	20	20	20	33	C9	ME FAT16 3
00000040	8E	D1	BC	F0	7B	8E	D9	B8	00	20	8E	C0	FC	BD	00	7C	тСjр{тЩё тАЬS
00000050	38	4E	24	7D	24	8B	C1	99	E8	3C	01	72	1C	83	EB	3A	8N\$}§<В™и< r фл
00000060	66	A1	1C	7C	26	66	3B	07	26	8A	57	FC	75	06	80	CA	f\$ &f; &ЬWьu E
00000070	02	88	56	02	80	C3	10	73	EB	33	C9	8A	46	10	98	F7	eV ЪГ слЗЙЬF C
00000080	66	16	03	46	1C	13	56	1E	03	46	0E	13	D1	8B	76	11	f F V F C<v
00000090	60	89	46	FC	89	56	FE	B8	20	00	F7	E6	8B	5E	0B	03	`%Fь%Vюё чж<^
000000A0	C3	48	F7	F3	01	46	FC	11	4E	FE	61	BF	00	00	E8	E6	ГНчу Fь Ньюi и
000000B0	00	72	39	26	38	2D	74	17	60	B1	0B	BE	A1	7D	F3	A6	r9&8-t `± s\$}y

(Root directory)	16,0 KB						480
Boot sector	1,0 KB						0
FAT 1	120 KB						2
FAT 2	120 KB						241
Free space	1,9 GB						
Idle space							

Рис. 2.13. Команда перехода JMP на начало загрузчика

Далее расположено поле из 8 байт, содержащее текстовый идентификатор версии ОС (рисунок 2.14.).

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	EB	3C	90	4D	53	44	4F	53	35	2E	30	00	02	40	02	00	л<MSDOS.0 @
00000010	02	00	02	00	00	F8	EF	00	3F	00	FF	00	80	1F	00	00	шп ? я Ъ
00000020	80	B0	3B	00	00	00	29	23	18	C7	50	4E	4F	20	4E	41	Ъ°;)# ЭРНО NA
00000030	4D	45	20	20	20	20	46	41	54	31	36	20	20	20	33	C9	ME FAT16 ЗЙ
00000040	8E	D1	BC	F0	7B	8E	D9	B8	00	20	8E	C0	FC	BD	00	7C	ЪCjp(ЪЩё ЪАьS
00000050	38	4E	24	7D	24	8B	C1	99	E8	3C	01	72	1C	83	EB	3A	8N\$)S<В"и< г фл:
00000060	66	A1	1C	7C	26	66	3B	07	26	8A	57	FC	75	06	80	CA	фŸ &f; &ЪWъu ЪK
00000070	02	88	56	02	80	C3	10	73	EB	33	C9	8A	46	10	98	F7	ев ЪГ слЗЙЪF Пч
00000080	66	16	03	46	1C	13	56	1E	03	46	0E	13	D1	8B	76	11	f F V F C<v
00000090	60	89	46	FC	89	56	FE	B8	20	00	F7	E6	8B	5E	0B	03	`%Fъ%Vюё чж<^
000000A0	C3	48	F7	F3	01	46	FC	11	4E	FE	61	BF	00	00	E8	E6	ГНчу Fъ NюaI иж
000000B0	00	72	39	26	38	2D	74	17	60	B1	0B	BE	A1	7D	F3	A6	г9&8-t `± sŸ}y;
000000C0	61	74	32	4E	74	09	83	C7	20	3B	FB	72	E6	EB	DC	A0	at2Nt фЗ ;ыгжль
000000D0	FB	7D	B4	7D	8B	F0	AC	98	40	74	0C	48	74	13	B4	0E	ы}г}<p-П@t Нт г
000000E0	BB	07	00	CD	10	BB	EF	A0	FD	7D	EB	E6	A0	FC	7D	EB	» Н лп э)лж ь)л
000000F0	E1	CD	16	CD	19	26	8B	55	1A	52	B0	01	BB	00	00	E8	бН Н &<U R° » и
00000100	3B	00	72	E8	5B	8A	56	24	BE	0B	7C	8B	FC	C7	46	F0	; ги{ЪV\$S <ьЭFr
00000110	3D	7D	C7	46	F4	29	7D	8C	D9	89	4E	F2	89	4E	F6	C6	=)ЭFф)}НЩ%Nт%NцЖ
00000120	06	96	7D	CB	EA	03	00	00	20	0F	B6	C8	66	8B	46	F8	-)лк ИИф<Fш
00000130	66	03	46	1C	66	8B	D0	66	C1	EA	10	EB	5E	0F	B6	C8	f F f: PфBк л^ ИИ
00000140	4A	4A	8A	46	0D	32	B4	F7	E2	03	46	FC	13	56	FE	EB	JJЪF 2дчв Fъ Vюл
00000150	4A	52	50	06	53	6A	01	6A	10	91	8B	46	18	96	92	33	JRP Sj j `<F -'З
00000160	D2	F7	F6	91	F7	F6	42	87	CA	F7	76	1A	8A	F2	8A	E8	Тчц`чцBфKчv ЪТЬи

Рисунок 2.14. Текстовый идентификатор версии ОС

Параметрическая информация (таблица 2.4) в разделе диска находится в первом секторе и используется для работы с диском.

Таблица 2.4

Параметрическая информация о диске

Смещение	Длина	Содержание
00h	3 байта	Команда перехода на программу начально загрузки
03h	8 байт	Идентификатор изготовителя
0Bh	1 слово	Размер сектора в байтах
0Dh	1 байт	Размер кластера в секторах
0Eh	1 слово	Количество зарезервированы секторов в начале
10h	1 байт	Количество копий FAT
11h	1 слово	Количество элементов основного каталога
13h	1 слово	Общее количество секторов диска
15h	1 байт	Идентификатор формата
16h	1 слово	Размер FAT в секторах
18h	1 слово	Количество секторов на дорожке
1Ah	1 слово	Количество поверхностей (головок)
1Ch	4 байта	Количество скрытых секторов
27h	4 байта	Серийный номер тома
2Bh	11 байт	Метка тома

Информация о количестве байт в одном секторе находится в таблице со смещение 0Bh (рисунок 2.15.). Смещение 0Bh, длина – два байта. Мы видим там значения 00h 02h. Байты следует читать с конца. 02h 00h = 0200h. Переводя в десятичное – 512. В опциях рекомендуется включить View -> Show -> Data Interpreter. А в опциях Options -> Data Interpreter включить 8 bit (unsigned), 16 bit (unsigned), 32 bit (unsigned), Assembler opcodes, Digit Grouping. Тогда в плавающем окошке будет отображаться информация уже переведенная в 10 систему счисления. Ставим курсор на первое число, и, поскольку, необходимо перевести 2 байта – смотрим значение в поле 16 bit (для длины 1 байт смотрим 8 bit, для длины 4, dword, смотрим 32).

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	EB	3C	90	4D	53	44	4F	53	35	2E	30	00	02	40	02	00	л<М
00000010	02	00	02	00	00	F8	EF	00	3F	00	F1	00	80	15	00	00	
00000020	80	B0	3B	00	00	00	29	23	18	C7	50	4E	4E				
00000030	4D	45	20	20	20	20	46	41	54	31	36	20	20				
00000040	8E	D1	BC	F0	7B	8E	D9	B8	00	20	8E	C0	FC				
00000050	38	4E	24	7D	24	8B	C1	99	E8	3C	01	72	1C				

Data Interpreter

8 Bit (+/-): 0

16 Bit (+/-): 512

32 Bit (+/-): 37 749 2 4

ASM: ADD

Рисунок 2.15. Текстовый идентификатор версии ОС

Сигнатура конца загрузочного раздела – последовательность байт 55Aah – находится внизу (рисунок 2.16.). Показывает, что загрузочный сектор закончился. Эта сигнатура присутствует во всех файловых системах, поэтому можно легко идентифицировать конец загрузочного сектора.

BB	UU	UU	6U	66	6A	UU	EB
20	20	20	20	0D	0A	52	65
6B	73	20	6F	72	20	6F	74
61	2E	FF	0D	0A	44	69	73
0D	0A	50	72	65	73	73	20
74	6F	20	72	65	73	74	61
00	00	00	AC	CB	D8	55	AA
00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00

Рисунок 2.16. Сигнатура конца загрузочного раздела

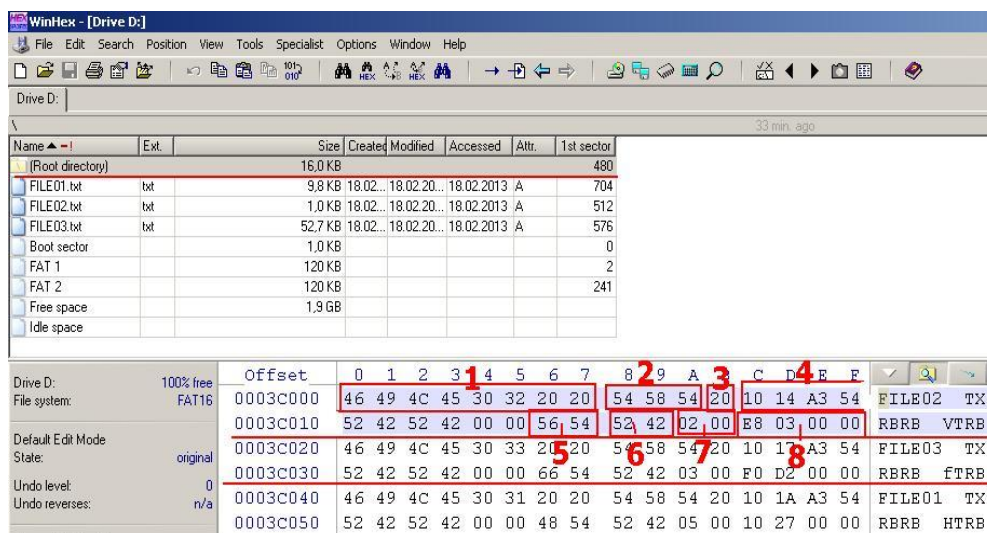


Рисунок 2.17. Структура элемента каталога, описывающего файл FILE02.TXT

Исследуем структуру файла, хранящегося на диске FAT32. Для этого создадим на диске файл FILE02.TXT (рисунок 2.17.)

1. Имя файла, закодированное ASCII-кодами (см. справа) – FILE02
2. Расширение файла – TXT
3. Атрибуты файла: 20h = 00100000b, переводим в атрибуты: выставлен бит с номером 5, считая от конца: это значит, что файл “Архивный”.
4. 54A3h = 101010010100011b, делим на битовые группы: 1010b 100101b 00011b = 10 37 03 в переводе на десятичные числа, что означает время 10:36:06 (секунды считаются парами); Далее идёт дата: 4252h = 100001001010010b, делим на битовые группы: 100001b 0010b 10010b = 33 2 18, переводим в дату: 18.02.2013 (1980г.+33). Получили дату создания файла.
5. Время и дата закодированы аналогично;
6. Дата последнего изменения, создания файла;
7. 0002h – первый кластер файла;
8. 03E8h – размер файла в байтах = 1000 байт.

Первый символ имени может иметь специальное значение:

- 00h – элемент каталога никогда не используется;
- 05h – первый символ имени имеет код E5h;
- 2Eh – псевдоним каталога (точка);
- E5h – элемент удален.

Подробнее рассмотрим то, как хранятся данные файла. Сначала узнаем, какие кластера занимает этот файл. Открываем таблицу FAT и ищем 2-ой кластер там. Пропускаем 6 байт (каждый кластер – 2 байта) и видим там FFFFh, что

означает конец цепочки. Значит файл уместился там полностью. Теперь ищем файл на диске. Размер кластера у нас 32 Кб (это мы вычислили ранее). Так же берём во внимание тот факт, что корневой каталог у нас может включать в себя максимум 512 элементов, на запись каждого из которых (включая атрибуты и даты-время) уходит ровно 32 байта, вы видите это из скриншота. $32 * 512 = 16384$. Прибавим это значение к смещению адреса начала корневого каталога 0003C000h, исходя из рисунка 2.17. и получим: 0040000h. Вот с этого смещения и начнётся область данных диска. Открываем это смещение (рисунок 2.18.).

0003FFFF0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000400000	30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35	0123456789012345
000400010	36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31	6789012345678901
000400020	32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37	2345678901234567
000400030	38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33	8901234567890123
000400040	34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39	4567890123456789
000400050	30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35	0123456789012345
000400060	36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31	6789012345678901
000400070	32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37	2345678901234567
000400080	38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33	8901234567890123
000400090	34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39	4567890123456789
0004000A0	30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35	0123456789012345
0004000B0	36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31	6789012345678901
0004000C0	32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37	2345678901234567
0004000D0	38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33	8901234567890123
0004000E0	34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39	4567890123456789
0004000F0	30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35	0123456789012345
000400100	36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31	6789012345678901
000400110	32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37	2345678901234567
000400120	38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33	8901234567890123

Рисунок 2.18. Данные файла

Теперь удалим файл вручную. Поставим в имени файла первым символом код E5h. Нетрудно заметить, что после сохранения изменений (Ctrl+S) файл исчезнет из содержимого каталога в проводнике. Обновим содержимое в WinHEX – Tools->DiskTools-

>TakeNewVolumeSnapshot(получить новый снимок) и увидим, что файл помечен удаленным (рисунок 2.19.).

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0003C000	E5	49	4C	45	30	32	20	20	54	58	54	20	10	14	A3	54
0003C010	52	42	52	42	00	00	56	54	52	42	02	00	E8	03	00	00
0003C020	46	49	4C	45	30	33	20	20	54	58	54	20	10	17	A3	54
0003C030	52	42	52	42	00	00	66	54	52	42	03	00	F0	D2	00	00
0003C040	46	49	4C	45	30	31	20	20	54	58	54	20	10	1A	A3	54
0003C050	52	42	52	42	00	00	48	54	52	42	05	00	10	27	00	00
0003C060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Рисунок 2.19. Удаление файла

Восстановить такой файл – проще простого. Данные остаются незатронутыми. Важно понимать, что удалив файл таким образом, получаются «потерянные кластеры», FAT кластер помечен, как «занятый», а никакому файлу он не принадлежит. При штатном удалении файла – система пробегает по всем кластерам в таблицах FAT и присваивает соответствующим элементам значения 0000h. Нетрудно в этом убедиться (рисунок 2.20).

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000400	F8	FF	FF	FF	00	00	04	00	FF	FF	FF	FF	00	00	00	00
00000410	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000420	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000430	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000440	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000450	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000460	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Рисунок 2.20. Штатное удаление файла

WinHEX позволяет изучить структуру FAT диска и файлов, которые на нем хранятся. Анализируя диск с помощью данной программы можно подробно изучить структуру FAT диска. Это может быть полезно для низкоуровневого программирования и решения задач связанных с защитой программ от копирования.

Name ▲ - !	Ext.	Size	Created	Modified	Accessed	Attr.	1st sector
(Root directory)		16,0 KB					480
FILE01.txt	txt	9,8 KB	18.02...	18.02.2013 10:34:36	18.02.2013	A	704
FILE03.txt	txt	52,7 KB	18.02...	18.02.2013 10:34:16	18.02.2013	A	576
FILE02.txt	txt	1,0 KB	18.02...	18.02.2013 10:33:20	18.02.2013	A	512
Boot sector		1,0 KB					0
FAT 1		120 KB					2
FAT 2		120 KB					241
Free space		1,9 GB					
Idle space							

2.3. Файловая система NTFS

NTFS (аббревиатура от англ. newtechnologyfilesystem – «файловая система новой технологии») – стандартная файловая система для семейства операционных систем Windows NT фирмы Microsoft.

NTFS поддерживает хранение метаданных. С целью улучшения производительности, надёжности и эффективности использования дискового пространства для хранения информации о файлах в NTFS используются специализированные структуры данных. Информация о файлах хранится в главной файловой таблице – Master File Table (MFT). NTFS поддерживает разграничение доступа к данным для различных пользователей и групп пользователей (списки кон-

троля доступа – англ. accesscontrollists, ACL), а также позволяет назначать (ограничения на максимальный объём дискового пространства, занимаемый файлами тех или иных пользователей). Для повышения надёжности файловой системы в NTFS используется система журналирования USN. Для NTFS размер кластера по умолчанию составляет от 512 байт до 64 КиБ в зависимости от размера тома и версии ОС.

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.

Диск NTFS условно делится на две части (рисунок 2.21.). Первые 12% диска отводятся под так называемую MFT зону – пространство, в которое растёт метафайл MFT (об этом ниже). Запись каких-либо данных в эту область невозможна. MFT-зона всегда держится пустой – это делается для того, чтобы самый главный, служебный файл (MFT) не фрагментировался при своем росте. Остальные 88% диска представляют собой обычное пространство для хранения файлов.

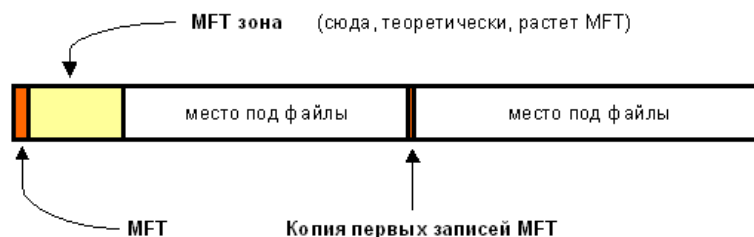


Рисунок 2.21. Структура NTFS

Свободное место диска включает в себя всё физически свободное место включая незаполненные куски MFT-зоны. 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 освобождении места в обычной области MFT зона может снова расширится. При этом не исключена ситуация, когда в этой зоне остались и обычные файлы. В итоге метафайл MFT все-таки может фрагментироваться.

2.3.1. MFT и его структура

Файловая система NTFS представляет собой структуру файлов: каждый элемент системы представляет собой файл – даже служебная информация. Самый главный файл на NTFS называется MFT, или Master File Table – общая

таблица файлов. Именно он размещается в MFT зоне и представляет собой централизованный каталог всех остальных файлов диска, и, себя самого. MFT поделен на записи фиксированного размера (обычно 1 Кбайт), и каждая запись соответствует какому либо файлу (в общем смысле этого слова). Первые 16 файлов носят служебный характер и недоступны операционной системе – они называются метафайлами, причем самый первый метафайл – сам MFT. Эти первые 16 элементов MFT – единственная часть диска, имеющая фиксированное положение. Вторая копия первых трех записей, для надежности – они очень важны – хранится ровно посередине диска. Остальной MFT-файл может располагаться, как и любой другой файл, в произвольных местах диска – восстановить его положение можно с помощью его самого, взяв его основу – первый элемент MFT.

2.3.2. Метафайлы

Первые 16 файлов NTFS (метафайлы) носят служебный характер. Каждый из них отвечает за какой-либо аспект работы системы. Преимущество настолько модульного подхода заключается в поразительной гибкости – например, на FAT-е 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 первых 16 элементов MFT.

Метафайлы находятся в корневом каталоге NTFS диска – они начинаются с символа имени «\$», хотя получить какую-либо информацию о них стандартными средствами сложно. Для этих файлов указан вполне реальный размер – можно узнать, например, сколько операционная система тратит на каталогизацию всего диска, посмотрев размер файла \$MFT. В таблице 2.5. приведены используемые в данный момент метафайлы и их назначение.

Таблица 2.5

Метафайлы	
\$MFT	сам MFT
\$MFTmirr	копия первых 16 записей MFT, размещенная посередине диска
\$LogFile	файл поддержки журналирования (см. ниже)
\$Volume	служебная информация – метка тома, версия файловой системы, т. д.

\$AttrDef	список стандартных атрибутов файлов на томе
\$.	корневой каталог
\$Bitmap	карта свободного места тома
\$Boot	загрузочный сектор (если раздел загрузочный)
\$Quota	файл, в котором записаны права пользователей на использование дискового пространства (начал работать лишь в NT5)
\$Upcase	файл – таблица соответствия заглавных и прописных букв в имен файлов на текущем томе. Нужен в основном потому, что в NTFS имена файлов записываются в Unicode, что составляет 65 тысяч различных символов, искать большие и малые эквиваленты которых очень нетривиально.

2.3.3. Файлы и потоки

Так как система оперирует только файлами, это накладывает следующие особенности работы NTFS дисков:

- Прежде всего, обязательный элемент – запись в MFT, так как все файлы диска упоминаются в MFT. В этом месте хранится вся информация о файле, за исключением собственно данных: имя файла, размер, положение на диске отдельных фрагментов, и т. д. Если для информации не хватает одной записи MFT, то используются несколько, причем не обязательно подряд.
- Опциональный элемент – потоки данных файла. Если файл не имеет никаких данных, то на него не расходуется свободное место на жестком диске. Если файл имеет не очень большой размер, то данные файла хранятся прямо в MFT, в оставшемся от основных данных месте в пределах одной записи MFT. Файлы, занимающие сотни байт, обычно не имеют своего «физического» воплощения в основной файловой области – все данные такого файла хранятся в одном месте – в MFT.

Довольно интересно обстоит дело и с данными файла. Каждый файл на NTFS, в общем-то, имеет несколько абстрактное строение – у него нет как таковых данных, а есть потоки (streams). Один из потоков и носит привычный нам смысл – данные файла. Но большинство атрибутов файла – тоже потоки! Таким образом, получается, что базовая сущность у файла только одна – номер в MFT, а всё остальное опционально. Данная абстракция может использоваться для создания довольно удобных вещей – например, файлу можно «прилепить» еще один поток, записав в него любые данные – например, информацию об авторе и содержании файла, как это сделано в Windows 2000 (самая правая закладка в свойствах файла, просматриваемых из проводника). Интересно, что эти дополнительные потоки не видны стандартными средствами: наблюдаемый размер файла – это лишь размер основного потока, который содержит традиционные данные. К примеру может иметься файл нулевой длины, при стирании которого освободится 1 Гбайт свободного места – просто потому, что какая-нибудь программа или технология добавила ему дополнительный поток (альтернативные данные) большого размера. Но на самом деле в текущий момент потоки практически не используются, так что опасаться подобных ситуаций не следует, хотя гипотетически они возможны. Следует учитывать, что файл на NTFS – это более глубокое и глобальное понятие, чем может показаться при простом просмотре каталога диска.

Имена файлов могут содержать любые символы, включая полный набор национальных алфавитов, так как данные представлены в Unicode – 16-битном представлении, которое дает 65535 разных символов. Максимальная длина имени файла – 255 символов.

2.3.4. Каталоги

Каталог в NTFS представляет собой специфический файл, хранящий ссылки на другие файлы и каталоги, создавая иерархическое строение данных на диске. Файл каталога поделен на блоки, каждый из которых содержит имя файла, базовые атрибуты и ссылку на элемент MFT, который уже предоставляет полную информацию об элементе каталога. Внутренняя структура каталога представляет собой бинарное дерево. Это означает, что для поиска файла с данным именем в линейном каталоге, таком, например, как у FAT-а, операционной системе приходится просматривать все элементы каталога, пока она не найдет нужный. Бинарное же дерево располагает именами файлов таким образом, чтобы поиск файла осуществлялся более быстрым способом – с помощью получения двузначных ответов на вопросы о положении файла. Вопрос, на который бинарное дерево способно дать ответ, таков: в какой группе, относительно данного элемента, находится искомое имя – выше или ниже? Такой вопрос задается среднему элементу, и каждый ответ сужает зону поиска в среднем в два

раза. Файлы 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 (рисунок 2.21.).

Следовательно для поиска одного (рисунок 2.22.) файла среди 1000, например, FAT придется осуществить в среднем 500 сравнений (наиболее вероятно, что файл будет найден на середине поиска), а системе на основе дерева – всего около двенадцати ($2^{10} = 1024$). Экономия времени поиска налицо. Не стоит, однако думать, что в традиционных системах (FAT) всё так запущено: во-первых, поддержание списка файлов в виде бинарного дерева довольно трудоемко, а во-вторых – даже FAT в исполнении современных систем использует сходную оптимизацию поиска. Хочется также развеять распространенное заблуждение о том, что добавлять файл в каталог в виде дерева труднее, чем в линейный каталог: это достаточно сравнимые по времени операции – дело в том, что для того, чтобы добавить файл в каталог, нужно сначала убедиться, что файла с таким именем там еще нет – в линейной системе будут трудности с поиском файла, описанные выше, которые компенсируют саму простоту добавления файла в каталог.

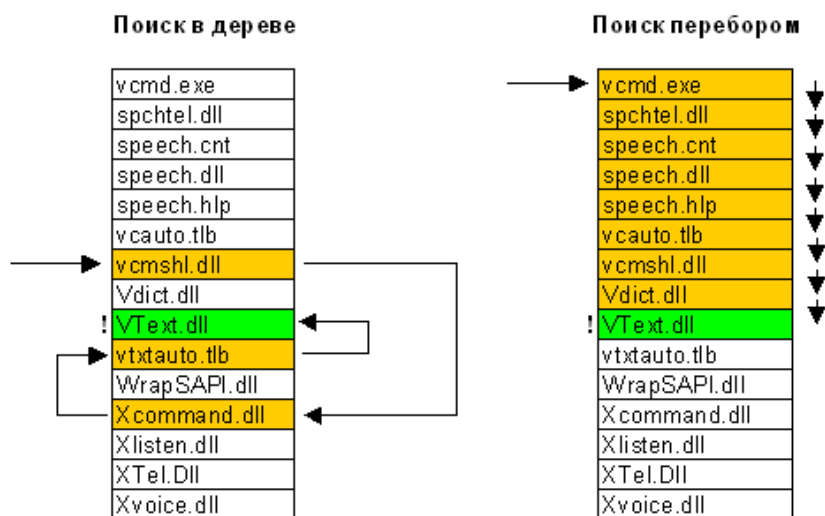


Рисунок 2.22. Поиск в каталоге NTFS

Какую информацию можно получить, просто прочитав файл каталога? Ровно то, что выдает команда `dir`. Для выполнения простейшей навигации по диску не нужно лазить в MFT за каждым файлом, надо лишь читать самую общую информацию о файлах из файлов каталогов. Главный каталог диска – корневой – ничем не отличается от обычных каталогов, кроме специальной ссылки на него из начала метафайла MFT.

2.3.5. Журналирование

NTFS – отказоустойчивая система, которая вполне может привести себя в корректное состояние при практически любых реальных сбоях. Любая современная файловая система основана на таком понятии, как транзакция – действие, совершаемое целиком и корректно или не совершаемое вообще. У NTFS просто не бывает промежуточных (ошибочных или некорректных) состояний – квант изменения данных не может быть поделен на до и после сбоя, принося разрушения и путаницу – он либо совершен, либо отменен.

Пример 1: осуществляется запись данных на диск. Вдруг выясняется, что в то место, куда мы только что решили записать очередную порцию данных, писать не удалось – физическое повреждение поверхности. Поведение NTFS в этом случае довольно логично: транзакция записи откатывается целиком – система осознает, что запись не произведена. Место помечается как сбойное, а данные записываются в другое место – начинается новая транзакция.

Пример 2: более сложный случай – идет запись данных на диск. При отключении питания и система перезагружается. На какой фазе остановилась запись и какие данные записались? На помощь приходит другой механизм системы – журнал транзакций. Дело в том, что система, осознав свое желание писать на диск, пометила в метафайле \$LogFile это свое состояние. При перезагрузке это файл изучается на предмет наличия незавершенных транзакций, которые были прерваны аварией и результат которых непредсказуем – все эти транзакции отменяются: место, в которое осуществлялась запись, помечается снова как свободное, индексы и элементы MFT приводятся в состояние, в котором они были до сбоя, и система в целом остается стабильна. Если ошибка происходит при записи в журнал, то транзакция либо еще и не начиналась (идет только попытка записать намерения её произвести), либо уже закончилась – то есть идет попытка записать, что транзакция на самом деле уже выполнена. В последнем случае при следующей загрузке система сама вполне разберется, что на самом деле всё и так записано корректно, и не обратит внимания на «незаконченную» транзакцию.

Следует понимать, что журналирование – не абсолютная панацея, а лишь средство существенно сократить число ошибок и сбоев системы. Вряд ли рядовой пользователь NTFS хоть когда-нибудь заметит ошибку системы или вынужден будет запускать chkdsk – опыт показывает, что NTFS восстанавливается в полностью корректное состояние даже при сбоях в очень загруженные дисковой активностью моменты. Можно даже оптимизировать диск и в самый разгар этого процесса нажав reset – вероятность потерь данных даже в этом случае будет очень низка. Важно понимать, однако, что система восстановления NTFS

гарантирует корректность файловой системы, а не данных. Если производилась запись на диск и случилась аварийная ситуация – данные могут и не записаться.

2.3.6. Сжатие

Файлы NTFS имеют один довольно полезный атрибут – «сжатый». NTFS имеет встроенную поддержку сжатия дисков – то, для чего раньше приходилось использовать Stacker или DoubleSpace. Любой файл или каталог в индивидуальном порядке может храниться на диске в сжатом виде – этот процесс совершенно прозрачен для приложений. Сжатие файлов имеет очень высокую скорость и только одно большое 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 компрессированности определенных фрагментов очень похоже на обычную фрагментацию файлов: например, типичная запись физической раскладки для реального, несжатого, файла:

- кластеры файла с 1 по 43-й хранятся в кластерах диска начиная с четырехсотого;
- кластеры файла с 44 по 52-й хранятся в кластерах диска начиная с 8530;

Физическая раскладка типичного сжатого файла:

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

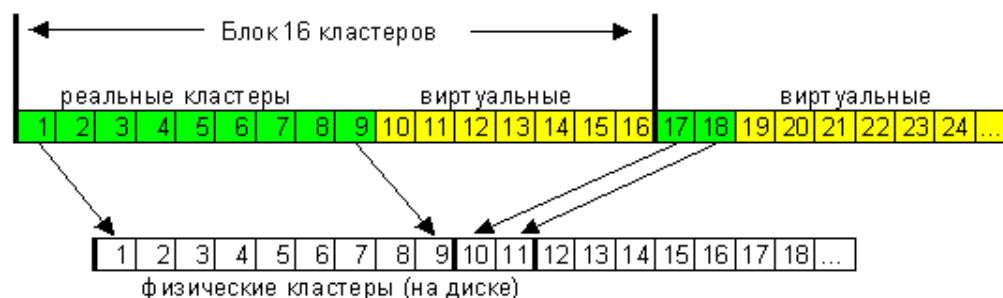


Рисунок 2.23. Сжатие файла

Видно, что сжатый файл имеет «виртуальные» кластеры, реальной информации в которых нет. Как только система видит такие виртуальные кластеры, она тут же понимает, что данные предыдущего блока, кратного шестнадцати, должны быть разжаты, а получившиеся данные как раз заполняют виртуальные кластеры – вот, по сути, и весь алгоритм.

2.4. Сравнительные характеристики файловых систем

FAT32 (File Allocation Table) – файловая система, созданная компанией Microsoft на замену умирающей FAT16. На данный момент самая распространенная система. Большая часть карточек памяти и флэшек поставляется в FAT32. Кроме того, при форматировании носителя в бытовом устройстве, например фотоаппарате, он будет работать именно в этой файловой системе. В этом заключается ее главный плюс – совместимость. Вставив такой накопитель в бытовой DVD-плеер, медиапроигрыватель или фотопринтер, можно быть уверенным, что все файлы доступны и могут нормально прочитаться.

Самым главным недостатком данной системы является ограничение на размер файла в 4 Гбайта. Поэтому записать большой файл (например, резервную копию системного диска или переписанное с камеры видео) не получится – система выдаст ошибку «Недостаточно дискового пространства».

Диск использующий FAT32 хранит электронную таблицу размещения файлов, которая находится практически в самом начале диска и имеет следующую структуру:

- загрузочные секторы главного и дополнительного разделов;
- загрузочный сектор логического диска;
- корневой каталог;
- область данных;
- цилиндр для выполнения диагностических операций чтения/записи.

NTFS – файловая система, берущая начало с Windows NT. При установке NTFS, диск разделяется на две неравные части: первая отводится под MFT (Master File Table – общая таблица файлов), называется MFT – зоной и занимает порядка 12% от общего размера диска, вторую часть занимают данные. MFT – это основа NTFS. Каждая запись в MFT соответствует какому-либо файлу и занимает около 1 Kb. По своей сути это каталог всех файлов находящихся на диске. Любой элемент данных в NTFS рассматривается как файл, даже MFT. Первые 16 файлов (метафайлы) в MFT содержат служебную информацию, они имеют фиксированное положение и они недоступны даже операционной системе. Существует копия первых трех записей, это сделано для надежности, в случае утери информации в MFT – файле, всегда можно восстановить информа-

цию. Все остальные файлы в MFT – зоне могут располагаться произвольно. Надо заметить, что в MFT – зоне теоретически кроме служебных файлов ничего не находится. Но бывают случаи, когда места на той части диска, что отведена для пользователя не остается, тогда MFT – зона уменьшается. Соответственно появляется место во второй половине (резервная копия первых трех записей делит диск пополам) диска для записи данных. Когда же в этой зоне освобождается достаточное количество свободного места, MFT – зона опять расширяется. В MFT – зону попадают обычные файлы и она начинает фрагментироваться.

NTFS практически не имеет ограничения на размеры диска (во всяком случае при нынешних технологиях производства жестких дисков). Размер кластера может варьироваться от 512 b до 64 Kb, хотя обычный его размер равен 4 Kb.

Каталог в NTFS представляет собой специфический файл, хранящий ссылки на другие файлы и каталоги, создавая иерархическое строение данных на диске. Файл каталога поделен на блоки, каждый из которых содержит имя файла, базовые атрибуты и ссылку на элемент MFT, который уже предоставляет полную информацию об элементе каталога. Внутренняя структура каталога представляет собой бинарное дерево. Это означает, что для поиска файла с данным именем в линейном каталоге, таком, например, как у FAT-а, операционной системе приходится просматривать все элементы каталога, пока она не найдет нужный. Бинарное же дерево располагает имена файлов таким образом, чтобы поиск файла осуществлялся более быстрым способом – с помощью получения двузначных ответов на вопросы о положении файла.

Файлы в NTFS представлены потоками. То есть, любая единица информации представляет собой несколько потоков. Один поток – это сами данные, он является основным. Другие потоки – атрибуты файла. К любому файлу можно прикрепить любой другой файл. Это позволяет к потокам одних данных прикрепить совершенно новый поток и записать туда новые данные. Информация по объему файла берется по объему основного потока. Пустые или мало-размерные файлы на диске отображены только в метафайлах. Сделано это в целях экономии дискового пространства.

Файлы NTFS имеют такой атрибут как сжатый. Любой файл или даже каталог может быть сжат. Сама операция сжатия происходит незаметно, так как скорость ее довольно высока, используется так называемое виртуальное сжатие т.е. одна часть файла может быть сжата, а другая нет. Сжатие осуществляется блоками. Каждый блок равен 16 кластерам.

Каждая файловая система выполняет элементарные операции с файлами – доступ, удаление, создание, перемещение и т.д. Скорость работы этих операций зависит от принципов организации хранения данных об отдельных файлах и от устройства структур каталогов. Все это влияет на скорость осуществления любых

операций с файлом, в том числе – на скорость любой операции доступа к файлу, особенно – в каталогах с большим числом файлов (тысячи).

FAT16 и FAT32 имеют очень компактные каталоги, размер каждой записи которых предельно мал. Более того, из-за сложившейся исторически системы хранения длинных имен файлов (более 11 символов), в каталогах систем FAT используется не очень эффективная и на первый взгляд неудачная, но зато очень экономная структура хранения этих самих длинных имен файлов. Работа с каталогами FAT производится достаточно быстро, так как в подавляющем числе случаев каталог (файл данных каталога) не фрагментирован и находится на диске в одном месте.

Единственная проблема, которая может существенно понизить скорость работы каталогов FAT – большое количество файлов в одном каталоге (порядка тысячи или более). Система хранения данных – линейный массив – не позволяет организовать эффективный поиск файлов в таком каталоге, и для нахождения данного файла приходится перебирать большой объем данных (в среднем – половину файла каталога).

NTFS использует гораздо более эффективный способ адресации – бинарное дерево. Эта организация позволяет эффективно работать с каталогами любого размера – каталогам NTFS не страшно увеличение количества файлов в одном каталоге и до десятков тысяч.

Стоит заметить, однако, что сам каталог NTFS представляет собой гораздо менее компактную структуру, нежели каталог FAT – это связано с гораздо большим (в несколько раз) размером одной записи каталога. Данное обстоятельство приводит к тому, что каталоги на томе NTFS в подавляющем числе случаев сильно фрагментированы. Размер типичного каталога на FAT-е укладывается в один кластер, тогда как сотня файлов (и даже меньше) в каталоге на NTFS уже приводит к размеру файла каталога, превышающему типичный размер одного кластера. Это, в свою очередь, почти гарантирует фрагментацию файла каталога, что, к сожалению, довольно часто сводит на нет все преимущества гораздо более эффективной организации самих данных.

Таким образом, структура каталогов на NTFS теоретически гораздо эффективнее, но при размере каталога в несколько сотен файлов это практически не имеет значения. Фрагментация каталогов NTFS, однако, уверенно наступает уже при таком размере каталога. Для малых и средних каталогов NTFS, как это не печально, имеет на практике меньшее быстродействие.

Преимущества каталогов NTFS становятся реальными и неоспоримыми только в том случае, если в одном каталоге присутствуют тысячи файлов – в этом случае быстродействие компенсирует фрагментированность самого каталога и трудности с физическим обращением к данным (в первый раз – далее каталог кэ-

шируется). Напряженная работа с каталогами, содержащими порядка тысячи и более файлов, проходит на NTFS буквально в несколько раз быстрее, а иногда выигрыш в скорости по сравнению с FAT и FAT32 достигает десятков раз.

Следующая частая операция – поиск данных файла. Операция выясняющая, в каких областях диска хранится тот или иной фрагмент файла – процесс, который имеет принципиально разное воплощение в различных файловых системах. Имейте в виду, что это лишь поиск информации о местоположении файла – доступ к самим данным, фрагментированы они или нет, здесь уже не рассматривается, так как этот процесс совершенно одинаков для всех систем. Речь идет о тех «лишних» действиях, которые приходится выполнять системе перед доступом к реальным данным файлов. Это влияет на скорость навигации по файлу (доступ к произвольному фрагменту файла). Любая работа с большими файлами данных и документов, если их размер – несколько мегабайт и более. Этот параметр показывает, насколько сильно сама файловая система страдает от фрагментации файлов.

FAT32, из-за большой области самой таблицы размещения будет испытывать огромные трудности, если фрагменты файла разбросаны по всему диску. Дело в том, что FAT (FileAllocationTable, таблица размещения файлов) представляет собой мини-образ диска, куда включен каждый его кластер. Для доступа к фрагменту файла в системе FAT16 и FAT32 приходится обращаться к соответствующей частичке FAT. Если файл, к примеру, расположен в трех фрагментах – в начале диска, в середине, и в конце – то в системе FAT нам придется обратиться к фрагменту FAT также в его начале, в середине и в конце. В системе FAT16, где максимальный размер области FAT составляет 128 Кбайт, это не составит проблемы – вся область FAT просто хранится в памяти, или же считывается с диска целиком за один проход и буферизируется. FAT32 же, напротив, имеет типичный размер области FAT порядка сотен килобайт, а на больших дисках – даже несколько мегабайт. Если файл расположен в разных частях диска – это вынуждает систему совершать движения головок винчестера столько раз, сколько групп фрагментов в разных областях имеет файл, а это очень и очень сильно замедляет процесс поиска фрагментов файла.

NTFS способна обеспечить быстрый поиск фрагментов, поскольку вся информация хранится в нескольких очень компактных записях (типичный размер – несколько килобайт). Если файл очень сильно фрагментирован (содержит большое число фрагментов) – NTFS придется использовать много записей, что часто заставит хранить их в разных местах. Лишние движения головок при поиске этих данных, в таком случае, приведут к сильному замедлению процесса поиска данных о местоположении файла.

FAT16 никогда не заставит систему делать лишние дисковые операции для данной цели. Затем идет NTFS – эта система также не требует чтения лишней информации, по крайней мере, до того момента, пока файл имеет разумное

число фрагментов. FAT32 испытывает огромные трудности, вплоть до чтения лишних сотен килобайт из области FAT, если файл разбросан разным областям диска. Работа с внушительными по размеру файлами на FAT32 в любом случае сопряжена с огромными трудностями – понять, в каком месте на диске расположен тот или иной фрагмент файла, можно лишь изучив всю последовательность кластеров файла с самого начала, обрабатывая за один раз один кластер (через каждые 4 Кбайт файла в типичной системе). Стоит отметить, что если файл фрагментирован, но лежит компактной кучей фрагментов – FAT32 всё же не испытывает больших трудностей, так как физический доступ к области FAT будет также компактен и буферизован.

Поиск свободного места на диске. Операция производится 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 участках диска. Это влияет на скорость создания файлов, особенно больших. Сохранение или создание в реальном времени больших мультимедийных файлов (.wav, к примеру), копирование больших объемов информации, т. д. Этот параметр показывает, насколько быстро система сможет найти место для записи на диск новых данных, и какие операции ей придется для этого проделать.

Для определения того, свободен ли данный кластер или нет, системы на основе FAT должны просмотреть одну запись FAT, соответствующую этому кластеру. Размер одной записи FAT16 составляет 16 бит, одной записи FAT32 – 32 бита. 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), операционной системе 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.

Таким образом, NTFS имеет наиболее эффективную систему нахождения свободного места. Стоит отметить, что действовать «в лоб» на FAT16 или FAT32 очень медленно, поэтому для нахождения свободного места в этих системах применяются различные методы оптимизации, в результате чего и там достигается приемлемая скорость. (Одно можно сказать наверняка – поиск свободного места при работе в DOS на FAT32 – катастрофический по скорости процесс, поскольку никакая оптимизация невозможна без поддержки хоть сколь-нибудь серьезной операционной системы).

Достоинства FAT:

- для эффективной работы требуется немного оперативной памяти;
- быстрая работа с малыми и средними каталогами;

- диск совершает в среднем меньшее количество движений головок (в сравнении с NTFS);
- эффективная работа на медленных дисках.

Недостатки FAT:

- 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) файлам;
- очень медленная работа с каталогами, содержащими большое количество файлов.

Достоинства NTFS:

- 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 также не чинит 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).

Недостатки NTFS:

- существенные требования к памяти системы;
- медленные диски и контроллеры без BusMastering сильно снижают быстродействие NTFS;
- работа с каталогами средних размеров затруднена тем, что они почти всегда фрагментированы;
- диск, долго работающий в заполненном на 80% – 90% состоянии, будет показывать крайне низкое быстродействие.

NTFS – система, которая закладывалась на будущее. На данный момент NTFS обеспечивает стабильное и равнодушное к целому ряду факторов, но, пожалуй, всё же невысокое быстродействие. Основное преимущество NTFS с точки зрения быстродействия заключается в том, что этой системе безразличны такие параметры, как сложность каталогов (число файлов в одном каталоге), размер диска, фрагментация и т.д. В системах FAT же, напротив, каждый из этих факторов приведет к существенному снижению скорости работы.

ЗАКЛЮЧЕНИЕ

Данное учебное пособие позволяет студентом тщательным образом изучить организацию жесткого диска и познакомиться со спецификой работы файловых систем FAT и NTFS.

Стандарты время от времени меняются. И это естественно, так как технологии развиваются и, чтобы их плоды эффективно работали, необходимы новые спецификации. Основной особенностью данного учебного пособия является описание логической структуры жесткого диска с двумя схемами разделов: устаревшей, но все еще широко используемой MBR и созданной относительно недавно, но получающей все большее распространение GPT.

Операционная система, которая является основой работы любой компьютерной техники, организует работу с электронными данными, следуя определенному алгоритму, в цепочке которого файловая система занимает важнейшее место.

Файловая система – это часть операционной системы, которая связана непосредственно с размещением, удалением, перемещением электронной информации на определенном носителе, а также безопасностью ее дальнейшего использования в будущем. Именно это ресурс также применим в случаях, когда требуется восстановление утерянной информации по причине программного сбоя, как такового. То есть это основной инструмент работы с электронными файлами. В данной учебном пособии описаны особенности работы файловых систем FAT и NTFS с точки зрения операционной системы, то есть рассмотрены тонкости организации работы с файлами на низком уровне.

Данное учебное пособие помогает:

- углублять и закреплять знаний, полученных студентами на лекциях и в ходе самоподготовки;
- развивать у студентов способность к творческому, самостоятельному анализу учебной и методической литературы;
- вырабатывать умение систематизировать и обобщать усвоенный материал, критически оценивать его;
- формировать и укреплять навыки практического применения своих знаний, аргументированного, логического и грамотного изложения своих мыслей;
- прививать студентам навыки комплексного системного подхода к изучению и применению в практической деятельности полученных знаний;

- служить материалом для самопроверки при изучении и закреплении отдельных тем дисциплины «Системное программное обеспечение».

Учебное пособие рекомендуется для студентов, обучающихся по направлению «Информатика и вычислительная техника», а также может быть использовано студентами других специальностей, изучающих основные принципы работы с дисками и файлами.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Сравнение структур разделов GPT и MBR [Электронный ресурс] – Режим доступа : <https://habr.com/post/327572/>. – 24.07.2018.
2. Specifications [Электронный ресурс] : форум поддержки Unified Extensible Firmware. – Режим доступа : <http://www.uefi.org/specifications>. – 24.07.2018.
3. Михайлов, Д. Быстродействие FAT и NTFS [Электронный ресурс] / Д. Михайлов. – Режим доступа : <http://www.ixbt.com/storage/ntfs3.html>. – 24.07.2018.
4. Лекции по операционным системам [Электронный ресурс] – Режим доступа : <http://baumanki.net/lectures/10-informatika-i-programmirovanie/334-lekcii-po-operacionnym-sistemam>. – 24.07.2018.
5. Файловые системы и восстановление данных [Электронный ресурс] // КомпьютерМастер. – 2004 – Режим доступа : <http://computermaster.ru/articles/filesystem.html>. – 24.07.2018.
6. Файловые системы FAT и NTFS [Электронный ресурс] – Режим доступа : <https://www.windxp.com.ru/sistem2.htm>. – 24.07.2018.

СПИСОК СОКРАЩЕНИЙ

APFS	Apple File System	Файловая система Apple
API	Application Programming Interface	Программный интерфейс приложения
ASCII	American standard code for information interchange	Американский стандартный код для обмена информацией
BIOS	Basic Input/Output System	Базовая система ввода/вывода
CD	Compact Disc	Компакт диск
DOS	Disk Operating System	Дисковая операционная система
DVD	Digital Versatile Disc	Цифровой многоцелевой диск
EBR	Extended Boot Record	Расширенная загрузочная запись
EFI	Extensible Firmware Interface	Расширяемый интерфейс прошивки
EFS	Encrypting File System	Шифрованная файловая система
exFAT	extended File Allocation Table	Расширенная таблица размещения файлов
FAT	File Allocation Table	Таблица размещения файлов
FEK	File Encrypting Key	Ключ шифрования файла
GPL	General Public License	Главная публичная лицензия
GPT	GUID Partition Table	Таблица разделов GUID
GUID	Globally Unique Identifier	Глобальный уникальный идентификатор
HDD	Hard Disk Drive	Жесткий диск
HFS	Hierarchical File System	Иерархическая файловая система
HPFS	High Performance File System	Высокопроизводительная файловая система

LBA	Logical Block Addressing	Логическая адресация блоков
LDT	Logical Disks Table	Таблица логических дисков
MBR	Master boot record	Главная загрузочная запись
MFT	Master File Table	Главная таблица файлов
NSB	Non-System Bootstrap	Внесистемный загрузчик
NTFS	New Technology File System	Файловая система новой технологии
OS	Operation System	Операционная система
PC	Personal Computer	Персональный компьютер
SB	System Bootstrap	Системный загрузчик
SDD	Solid-State Drive	Твердотельный накопитель
SMBR	Secondary MBR	Первичная главная загрузочная запись
UEFI	Unified Extensible Firmware Interface	Унифицированный расширяемый интерфейс прошивки
USB	Universal Serial Bus	Универсальная последовательная шина
UTF	Unicode Transformation Format	Формат преобразования Юникода

Учебное издание

Караваева Ольга Владимировна

ДИСКИ И ФАЙЛОВАЯ ОРГАНИЗАЦИЯ

Учебное пособие

Авторская редакция

Технический редактор М. Н. Котельников

Подписано в печать 12.07.2018. Печать цифровая. Бумага для офисной техники.
Усл. печ. л. 4,83. Тираж 5 экз. Заказ № 5283.

Федеральное государственное бюджетное образовательное учреждение высшего
образования «Вятский государственный университет».

610000, г. Киров, ул. Московская, 36, тел.: (8332) 74-25-63, <http://vyatsu.ru>

