

Організація даних на ЖД.

На попередній парі ми розглянули основні принципи збереження інформації на ЖД, фізичну та логічну архітектуру накопичувачів. Розглянемо організацію розміщення даних та способи їх доступу/обробки.

Структура розташування файлів (FAT, NTFS)

1. Файлова система FAT (FAT-12, FAT-16, FAT-32, числа 12,16,32 – це довжина 1 елемента таблиці у бітах)

1 сектор ($C=0, H=0, S=1$) – boot-сектор, група секторів, які визначають 2 копії FAT-таблиць, наступні декілька секторів - коренева директорія.

В boot-секторі розміщується запис, що завантажується (загрузочний сектор, boot record, BR). BR може складатися з ланцюга boot-секторів, тоді boot-сектор $C=0, H=0, S=1$ називається головним MBS. Цей запис (програмний код і дані) завантажується у пам'ять з вінчестера і забезпечує опис логічних розділів на ЖД. Наявність запису визначає активний розділ, з якого буде продовжено запуск операційної системи (ОС).

Розмір сектору на жорсткому диску - 512 байт. Цього простору цілком вистачає для розміщення там всього необхідного - і коду, і даних. Проте тільки одна структура повинна бути там присутньою обов'язково - це сигнатура. Сигнатурою називається спеціальна, строго встановлена, послідовність з 2 байт з шістнадцятковими значеннями 55h AAh, яка записується в останні 2 байти сектора і відповідно має зсув від початку сектора 1FEh. Якщо хоч би один з двох останніх байтів відрізняється за значенням, вважається, що перший сектор не є MBR і не містить осмисленої інформації. Якщо комп'ютер при старті, прочитавши перший сектор, не виявить правильної сигнатури, він не передаватиме управління розташованому там коду, навіть якщо код правильний, а видасть повідомлення про те, що головний завантажувальний запис не знайдений або пробуватиме знайти її на інших пристроях.

Далі - до початку сектора. Перед сигнатурою, впритул до неї, розташовано 4 блоки даних по 16 байтів кожен (відповідно із зсувом від початку сектора 1BEh, 1CEh, 1DEh, 1EEh). Сукупність цих блоків називається Таблиця Розділів, Partition Table (PT), а кожен окремий запис - елементом таблиці розділів (Partition Table Entry) або просто розділом (Partition). Цих 16 байтів цілком достатньо, щоб вказати всі необхідні характеристики розділу, а саме: тип розділу, ознака активності розділу, початковий і кінцевий сектори розділу у форматі Циліндр (доріжка) - Головка (сторона) - Сектор (Cylinder - Head - Sector, CHS), відносний номер першого сектора (відносно MBR) і кількість секторів в розділі.

Решта простору сектора зайнята програмним кодом, який забезпечує розбір PT, пошук активного розділу, завантаження в пам'ять BR цього розділу і передачу йому управління. Як легко підрахувати, на код залишається $512 - 4 * 16 - 2 = 446$ байт. Цей простір з лихвою вистачає для виконання вказаних дій. Отже, загальна структура MBR може бути представлена наступною таблицею:

Зсув	Довжина	Опис
000h	446	Код завантажувача
1BEh	64	Таблиця розділів
	16	Розділ 1
1CEh	16	Розділ 2
1DEh	16	Розділ 3
1EEh	16	Розділ 4
1FEh	2	Сигнатура (55h AAh)

Кожен 16-байтний блок, що описує один розділ, має таку структуру:

Зсув	Довжина	Опис
00h	1	Ознака активності розділу
01h	1	Початок розділу - головка
02h	1	Початок розділу - сектор (біти 0-5), доріжка (біти 6,7)
03h	1	Початок розділу - доріжка (старші біти 8,9 зберігаються в байті номеру сектора)
04h	1	Код типу розділу
05h	1	Кінець розділу - головка
06h	1	Кінець розділу - сектор (біти 0-5), доріжка (біти 6,7)
07h	1	Кінець розділу - доріжка (старші біти 8,9 зберігаються в байті номеру сектора)
08h	4	Зсув першого сектора
0Ch	4	Кількість секторів розділу

Код типу розділу подається однобайтним ідентифікатором. Якщо його значення - 00h, то вважається, що в даному елементі РТ не містяться дані про розділ, і його вміст ігнорується. Будь-яке ненульове значення означає, що у вказаному просторі знаходиться розділ певного типу, які створюються операційними системами Windows NT/2000/XP:

Код	Тип розділу
01h	12-бітна FAT
04h	16-бітна FAT до 32 Мбайт
05h	Розширений розділ
06h	16-бітна FAT більше 32 Мбайт
07h	Windows NT NTFS
0Bh	32-бітна FAT
0Ch	32-бітна FAT з застосуванням розширеного управління Int13
0Eh	LBA VFAT (те ж саме, що й 06h, з застосуванням розширеного управління INT13)
0Fh	LBA VFAT (те ж саме, що й 05h, з застосуванням розширеного управління INT13)
17h	Прихований розділ NTFS
1Bh	Прихований розділ 32-бітної FAT (те ж саме, що 0Bh)
1Ch	Прихований розділ 32-бітної FAT з застосуванням розширеного управління INT13 (те ж саме, що 0Ch)
1Eh	Прихований розділ LBA VFAT (те ж саме, що, з застосуванням розширеного управління INT13)
86h	Розділ FAT-16 stripe-масиву Windows NT
87h	Розділ NTFS stripe-масиву Windows NT

B6h Дзеркальний master-розділ FAT-16 Windows NT
B7h Дзеркальний master-розділ NTFS Windows NT
C6h Дзеркальний slave-розділ FAT-16 Windows NT
C7h Дзеркальний slave-розділ NTFS Windows NT

Ознака активності розділу - тобто ознака того, що операційну систему потрібно завантажувати саме з цього розділу, має значення 80h (розділ активний) і 00h (розділ не активний). У загальному випадку кількість активних розділів повинна бути не більше 1 (інакше як зробити вибір?). Якщо активних розділів немає - означає з цього жорсткого диска ОС не може бути завантажена. Інші значення вважаються помилковими і ігноруються.

Трьохбайтний блок адреси початку та адреси кінця розділу мають ідентичний формат. Тут фактично використовується упаковка значень для того, щоб вони мали мінімальний об'єм. Формат упаковки повністю відповідає тому, як ці дані передаються процедурам роботи з жорстким диском (Int 13h), котрі знаходиться в BIOS комп'ютера. При цьому циліндри і доріжки нумеруються, починаючи з нульового значення, а сектори - з першого.

Сектор, на який указує адреса початку розділу, також містить в собі спеціальний запис, який називається завантажувальним записом (BR).

Зсув першого сектора розділу - це фактично номер цього сектора, якщо всі сектори жорсткого диска перенумерувати починаючи з 0 (відповідно до нумерації LBA, що використовується в Int 25h/26h BIOS) в порядку зростання спершу по секторах однієї доріжки, далі в порядку збільшення номерів головок і, нарешті, циліндрів.

Нагадаємо формули попереднього заняття. Отже, якщо позначити:

C_M - циліндр, на якому розміщується MBR;

H_M - доріжка, на якій розміщується MBR;

S_M - сектор, в якому розміщується MBR;

$C_S, H_S, S_S, C_E, H_E, S_E$ - те ж саме, для секторів початку (S) і кінця (E) розділу;

H_H - кількість доріжок у ЖД;

S_H - кількість секторів на одній доріжці у ЖД,

тоді:

- абсолютний номер сектора, в якому розміщується PT:

$$Num_{PT} = C_M * H_H * S_H + H_M * S_H + S_M - 1$$

- абсолютний номер сектору початку розділу:

$$Num_S = C_S * H_H * S_H + H_S * S_H + S_S - 1$$

- абсолютний номер сектору кінця розділу:

$$Num_E = C_E * H_H * S_H + H_E * S_H + S_E - 1$$

- зсув першого сектору розділу:

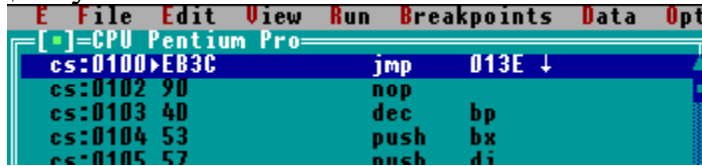
$$Offset_S = Num_S - Num_{PT}$$

- кількість секторів розділу:

$$Amount = Num_E - Num_S + 1$$

Блок BPB.

У момент старту ОС boot-запис вантажиться за адресою пам'яті **0:7c00**. У цю область передається управління (встановлюється регістр **IP**). Зазвичай там є команда **JMP** і адреса переходу. У цьому нескладно переконатися. Досить записати цей файл на диск, дати йому розширення **COM** і завантажити у відладчику.



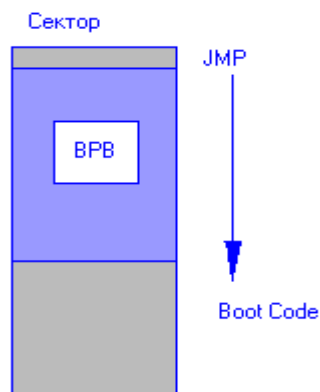
```
File Edit View Run Breakpoints Data Opt
[1]=CPU Pentium Pro
cs:0100>EB3C jmp 013E ↓
cs:0102 90 nop
cs:0103 40 dec bp
cs:0104 53 push bx
cs:0105 57 push di
```

За командою **JMP** йдуть параметри жорсткого диска – блок **BPB**. **JMP** дозволяє перейти на код завантаження, цей код читає таблицю розділів диска, визначає активний розділ і запускає програму завантаження ОС.

Розглянемо, що знаходиться в boot-секторі після команди **JMP** і які там параметри.

BS_jmpBoot +0 3 Команда **JMP** xxxx - перехід типа NEAR на програму початкового завантаження. Код початкового завантаження, зрозуміло, займає решту частини об'єму сектора 0, що розташовується за **BPB**.

JmpBoot[0] = 0xEB - найчастіше використаний формат переходу.



BS_OEMName +3(3) 8 Назва фірми-виробника ОС і версія, наприклад: **"IBM 5.0"**.

BPB_BytsPerSec +0Bh(11) 2 Байтів на сектор. Це значення може приймати тільки **512**, **1024**, **2048** або **4096**. Для максимальної сумісності бажано використовувати тут величину **512**. Система **MS Windows** правильно підтримує **1024**, **2048** і **4096**, але ці величини не рекомендуються.

BPB_SecPerClus +0Dh(13) 1 Секторів на кластер. Значення тут можуть бути такі - **1**, **2**, **4**, **8**, **16**, **32**, **64** або **128**.

BPB_RsvdSecCnt +0Eh(14) 2 Кількість зарезервованих секторів. Для **FAT12** і об'ємів **FAT16**, ця величина – **1**, для об'ємів **FAT32** - **32**.

BPB_NumFATs +10h(16) 1 Кількість таблиць **FAT**. Тут має бути **2**.

BPB_RootEntCnt +11h(17) 2 Максимальна кількість дескрипторів файлів, що містяться у кореневому каталозі диску.

BPB_TotSec16 +13h (19) 2 Загальна кількість секторів на носії даних (в розділі **DOS**). Це значення вже застаріло і може бути нулем, але тоді за зсувом **20h BPB_TotSec32** повинно бути не нуль.

BPB_Media +15h(21) 1 Байт-описувач середовища носія даних. **0xF8** - стандартна величина для незмінного носія. Для змінного носія, найчастіше застосовується **0xF0**. Можуть бути такі величини для цієї області - **0xF0, 0xF8, 0xF9, 0xFA, 0xFB, 0xFC, 0xFD, 0xFE** і **0xFF**.

BPB_FATSz16 +16h(22) 1 Кількість секторів, що займає одна копія **FAT**. Для **FAT16** тут реальне значення, для **FAT32** **0**

BPB_SecPerTrk +18h(24) 2 Кількість секторів на доріжці для переривання **0x13**.

BPB_NumHeads +1Ah(26) 2 Кількість головок для переривання **0x13**

BPB_HiddSec +1Ch(28) 4 Кількість прихованих секторів для розділу, що перевищує розмір 32 мегабайти.

BPB_TotSec32 +20h(32) 4 Загальна кількість секторів на логічному диску для розділу, що перевищує розмір 32 мегабайти.

Далі є відмінності для **FAT 16** і **FAT32**.

З блоку **BPB** можна отримати кількість секторів, зайнятих однією копією **FAT (KCFat)**. Якщо $KCFat * 512 * 8 /$ (довжину елементу **FAT**-таблиці), то отримуємо максимальну кількість кластерів дискового простору, доступних для обслуговування. Доступну кількість кластерів можна отримати, якщо розділити весь об'єм диска в байтах на величину = (значення Секторів на кластер з **BPB** * 512).

Структура таблиці **FAT**.

Складається з елементів, довжиною 12, 16 або 32 біти.

Заняті	елементи	n=2:0000	N=3:0000	n=4:0000	n=5:0000	n=6:0000
...						
			...	XXXXXX	XXXXXX	XXXXXX

Простір диска (кластери)

1	2	...				
512*SPC	512*SPC					

Значення елементів **FAT** (для прикладу **FAT-16**)

0000 – вільний кластер

FFF1 – bad кластер

FFF6 – резервний кластер

FFFF –останній кластер файлу

xxxx – номер кластеру

Між елементами **FAT**-таблиці та кластерами взаємно однозначна відповідність (позначено кольором). Якщо потрібно записати файл, розміром 20506 (при розмірі кластера 10246, тобто $SPC=2$), необхідно використати 2 повних кластера і кластер, в якому буде зайнято 2 б. Це буде оформлено так:

- 1) шукаємо у FAT-таблиці елемент з 0000. Нехай буде $n=2$. У кластер 1 записуємо 1024 байти.
 - Це перший фрагмент? Так. – Запам'ятовуємо номер елементу (тут2!).
 - Чи повністю файл збережено? Ні.
- 2) далі шукаємо у FAT-таблиці елемент з 0000. В нашому прикладі, $n=3$.
 - в елементі з $n=2$ записуємо значення 0003.
 - у кластері 2 записуємо 1024 байти.
 - Чи повністю файл збережено? Ні.
- 3) далі шукаємо у FAT-таблиці елемент з 0000. В нашому прикладі, $n=4$.
 - в елементі з $n=3$ записуємо значення 0004.
 - у кластері 3 записуємо 2 байти.
 - Чи повністю файл збережено? Так. В FAT-таблиці елемент з $n=4$ буде мати значення FFFF.

Коренева директорія.

З блоку BPB отримуємо Максимальну кількість дескрипторів файлів, що містяться в кореновому каталозі диска. Довжина елементу (дескриптора) кореневої директорії 32 байти, тоді величина $32 * (\text{макс. кількість дескр.}) / 512$ дає розмір області на диску в секторах, яку займає коренева директорія.

Зауваження. Коренева директорія – область логічного диска. Будь-яка інша директорія (папка) – файл.

Отже, КД складається з записів довжиною 32 б. Будь-який запис складається з 8 полів, що вирівнюються за лівою границею та доповнюються, за необхідності пропусками:

1. ім'я файлу, 8 байт. Якщо перший байт E5 – цей елемент не використовується, якщо 2E – цей елемент вказує на піддиректорію.
2. розширення файлу, 3 байти.
3. атрибут, 1 байт (0 біт – лише читання, 1 біт – прихований файл, 2 біт – системний файл, 3 біт – мітка тому, 4 біт – піддиректорія, 5 біт – архівний файл, 6 і 7 біти – резерв).
4. 10 байт – резерв.
5. час, 2 байти. 5 біт – години, 6 - хвилини, 5 секунди.
6. дата, 2 байти. 7 біт – роки з 1980, 4 - місяці, 5 дні.
7. номер початкового елемента з FAT-таблиці (!!! у прикладі = 2)
8. об'єм файлу, 4 байти. Потрібен для відновлення (читання) файлу.

Питання. Скільки файлів можна описати в головній директорії? Скільки - у піддиректоріях (папках)?

ОС Windows у файловій системі FAT підтримує довгі і короткі імена файлів. Короткі імена утворюються шляхом використання перших 6 символів від довгого імені плюс знак "~" плюс цифра. Далі слідує розширення файлу. Якщо у довгому імені є пропуски, то вони ігноруються. Якщо є декілька файлів з довгими іменами, які розрізняються деталями за межами перших 6 символів, то у коротких імен, що їм відповідають буде змінюватися остання цифра. Якщо таких схожих файлів більше чотирьох, то перші 6 символів визначаються за спеціальним алгоритмом, а кінець імені завжди однакове: "~5".

Наприклад, якщо файл має ім'я "поточна назва дисертації.txt", то його коротке ім'я "ПОТОЧН~1.УКР". Якщо у деякого файлу у цьому каталозі перші 6 символів імені та три символи розширення будуть такими ж, наприклад, "поточний стан україни.txt2020", то короткий варіант імені буде виглядати як "ПОТОЧН~2.txt".

Починаючи з Windows 95 було введено практику використання “довгих” імен для каталогів і файлів. У таких іменах допускається застосування символів різного регістру, наявність точок, пропусків та інших спеціальних символів. В каталозі під такий файл відводиться декілька суміжних 32-байтних блоків.

Ці елементи мають специфічне значення елементу “атрибут”, встановлено одночасно флаги “мітка тому”, “системний”, “прихований” і “лише читання”, завдяки чому ОС їх не сприймає при пошуку та не відображає. Після цих записів іде елемент зі звичною структурою, в якому зберігається коротке ім'я у вищезгаданому форматі. Цей елемент містить в собі номер початкового елемента таблиці FAT (початкового кластера), дату створення та атрибути файлу або каталогу.

Усі блоки з частинами довгого імені нумеруються послідовно у порядку слідування символів, причому блок з номером 01h буде розміщуватися безпосередньо перед блоком з коротким іменем.

Якщо у цей 01h-ий блок не вміщається довге ім'я, то перед ним буде блок з номером 02h і так далі, поки ім'я повністю не запишеться.

Структура довгого імені (32 байти, записуються перед коротким іменем)

Смещение	Длина в байтах	Описание
00h	1	Порядковый номер блока
01h	10	Имя (5 символов Unicode по 2 байта каждый)
0Bh	1	Атрибуты 0Bh
0Ch	1	Тип (00h)
0Dh	1	Контрольный код (берется из короткого имени)
0Eh	12	Имя (6 символов Unicode по 2 байта каждый)
1Ah	2	Байты 0000h
1Ch	4	Имя (2 символа Unicode по 2 байта каждый)

Один такий запис дозволяє зберігати $5+6+2=13$ символи з імені файлу.

FAT успішно застосовується у складі різних ОС (перш за все DOS і старі версії Windows). Проте, не слід забувати, що у той час, коли вона створювалася, основними пристроями для зберігання даних були гнучкі диски та диски малого об'єму. Потім з'явилися жорсткі диски великої місткості. В результаті окремі технологічні рішення наразі втратили свою актуальність.

До основних недоліків FAT слід віднести такі:

- обмеження, що накладаються на розмір файлів і дискового простору;
- обмеження довжини імені файлу;
- фрагментація файлів, що призводить до зменшення швидкодії системи і зносу устаткування;
- непродуктивні витрати пам'яті, викликані великими розмірами кластерів;
- схильність до втрат даних.

Windows підтримує дві файлові системи: давно вже існуючу файлову систему FAT і власну, нову файлову систему NTFS.

Нова файлова система NTFS володіє кращими показниками продуктивності і надійності в порівнянні з FAT. Ця файлова система підтримує об'єктно-орієнтовані додатки, обробляючи всі файли як об'єкти, які мають визначені користувачем або системою атрибути. NTFS дозволяє задавати права доступу до окремого файлу, а не до каталогу в цілому.

2. Структура файлової системи NTFS

Каждый файл на томе NTFS представлен записью в специальном файле, называемом Главной таблицей файлов (Master File Table, MFT). В отличие от разделов FAT все пространство тома NTFS представляет собой либо файл, либо часть файла. Основой структуры тома NTFS является Главная таблица файлов (Master File Table, MFT), которая содержит по крайней мере одну запись для каждого файла тома, включая одну запись для самой себя. Каждая запись имеет длину 2К.

Все файлы на томе NTFS идентифицируются номером файла, который определяется позицией файла в MFT. Каждый файл и каталог на томе NTFS состоит из набора атрибутов.

Базовая единица распределения дискового пространства для файловой системы NTFS - кластер. Размер кластера выражается в байтах и всегда равен целому количеству физических секторов. В качестве адреса файла NTFS использует номер кластера, а не физическое смещение в секторах или байтах. Загрузочный сектор тома NTFS располагается в начале тома, а его копия - в середине тома. Загрузочный сектор состоит из стандартного блока параметров BIOS, количества секторов в томе, а также начального логического номера кластера основной копии MFT и зеркальной копии MFT.

Файлы NTFS состоят по крайней мере из следующих атрибутов:

- заголовок (H - header)
- стандартная информация (SI - standard information)
- имя файла (FN - file name)
- данные (data)
- дескриптор безопасности (SD - security descriptor)

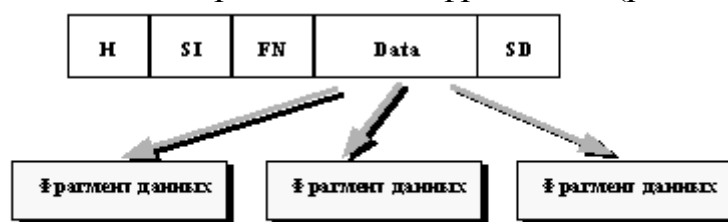
H	SI	FN	Data	SD
----------	-----------	-----------	-------------	-----------

H - заголовок, **SI** - атрибут стандартной информации,
FN - имя файла, **Data** - данные файла,
SD - дескриптор безопасности

Рис. 2.1. Небольшие файлы

Небольшие файлы (small). Если файл имеет небольшой размер, то он может целиком располагаться внутри одной записи MFT размером 2K (рис. 2.1). Из-за того, что файл может иметь переменное количество атрибутов, а также из-за переменного размера атрибутов нельзя наверняка утверждать, что файл уместится внутри записи. Однако, обычно файлы размером менее 1500 байт помещаются внутри записи MFT.

Большие файлы (Large). Если файл не вмещается в одну запись MFT, то этот факт отображается в значении атрибута "данные", который содержит признак того, что файл является нерезидентным, то есть, что файл находится вне таблицы MFT. В этом случае атрибут "данные" содержит виртуальный номер кластера для первого кластера каждого фрагмента данных (data run), а также количество непрерывных кластеров в каждом фрагменте (рис. 2.2).



Data- данные в нерезидентной

Рис. 2.2. Большие файлы

Очень большие файлы (huge). Если файл настолько велик, что его атрибут данных не помещается в одной записи, то этот атрибут становится нерезидентным, то есть он находится в другой записи таблицы MFT, ссылка на которую помещена в исходной записи о файле (рис.2.3). Эта ссылка называется внешним атрибутом (external attribute). Нерезидентный атрибут содержит указатели на фрагменты данных.

Сверхбольшие файлы (extremely huge). Для сверхбольших файлов внешний атрибут может указывать на несколько нерезидентных атрибутов (рис. 2.4). Кроме того, внешний атрибут, как и любой другой атрибут может храниться в нерезидентной форме, поэтому в NTFS не может быть атрибутов слишком большой длины, которые система не может обработать.

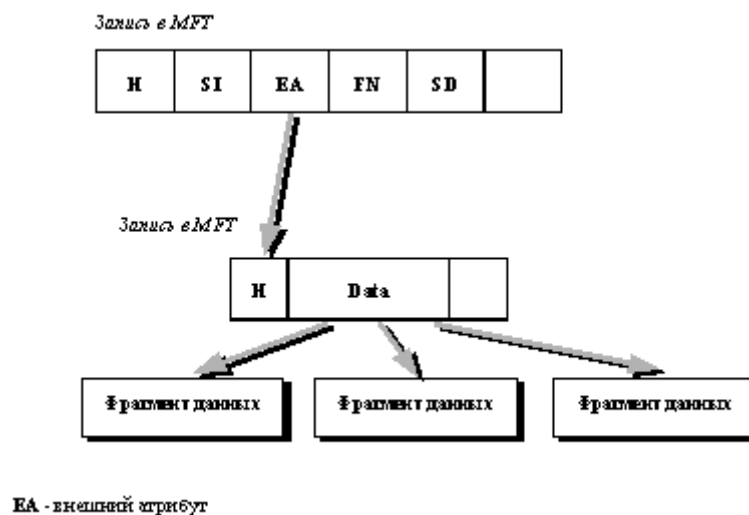


Рис. 2.3. Очень большие файлы

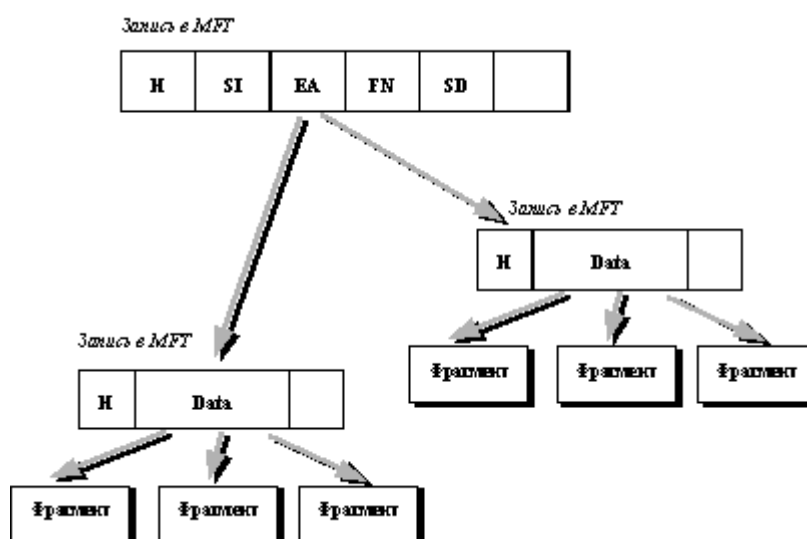


Рис. 2.4. Сверхбольшие файлы

Каждый каталог NTFS представляет собой один вход в таблицу MFT, который содержит список файлов специальной формы, называемый индексом (index). Индексы позволяют сортировать файлы для ускорения поиска, основанного на значении определенного атрибута. Обычно в файловых системах FAT используется сортировка файлов по имени. NTFS позволяет использовать для сортировки любой атрибут, если он хранится в резидентной форме. Имеется две формы списка файлов.

Небольшие списки файлов (small indexes). Если количество файлов в каталоге невелико, то список файлов может быть резидентным в записи в MFT, являющейся каталогом (рисунок 2.5). В этом случае он называется небольшим каталогом. Небольшой список файлов содержит значения атрибутов файла. По умолчанию это имя файла, а также номер записи MFT, содержащей начальную запись файла.

H	SI	FN	Список файлов (index) <a.bat, 27><c.sys, 92>... <xyz, f(xyz)> <####>	SD
---	----	----	--	----

- признак конца списка файлов

Рис. 2.5. Небольшие каталоги

Большие списки файлов (large index). По мере того, как каталог растет, список файлов может потребовать нерезидентной формы хранения. Однако начальная часть списка всегда остается резидентной в корневой записи каталога в таблице MFT (рисунок 2.6). Имена файлов резидентной части списка файлов являются узлами В-дерева. Остальные части списка файлов размещаются вне MFT. Для их поиска используется специальный атрибут "размещение списка" (Index Allocation - IA), представляющий собой набор номеров кластеров, которые указывают на остальные части списка. Одни части списков являются листьями дерева, а другие являются промежуточными узлами, то есть содержат наряду с именами файлов атрибут Index Allocation, указывающий на списки файлов более низких уровней.

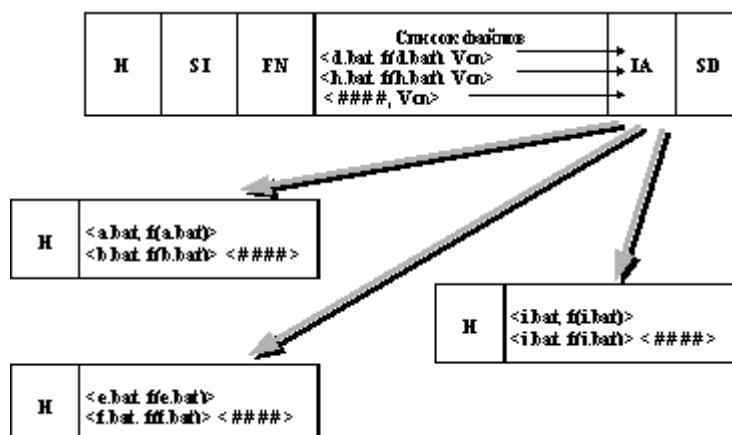


Рис. 2.6. Большие каталоги

Атрибуты файлов и каталогов

Каждый атрибут файла NTFS состоит из полей: тип атрибута, длина атрибута, значение атрибута и, возможно, имя атрибута.

Имеется системный набор атрибутов, определяемых структурой тома NTFS. Системные атрибуты имеют фиксированные имена и коды их типа, а также определенный формат. Могут применяться также атрибуты, определяемые пользователями. Их имена, типы и форматы задаются исключительно пользователем. Атрибуты файлов упорядочены по убыванию кода атрибута, причем атрибут одного и того же типа может повторяться несколько раз. Существует два способа хранения атрибутов файла - резидентное хранение в записях таблицы MFT и нерезидентное хранение вне ее. Сортировка может осуществляться только по резидентным атрибутам.

Ниже приведен список атрибутов.

- *Attribute List* - определяет список атрибутов, которые являются допустимыми для данного конкретного файла;
- *File Name* - этот атрибут содержит длинное имя файла, а также номер входа в таблице MFT для родительского каталога; если этот файл содержится в нескольких каталогах, то у него будет несколько атрибутов типа "File Name"; этот атрибут всегда должен быть резидентным;
- *MS-DOS Name* - этот атрибут содержит имя файла в формате 8.3;
- *Version* - атрибут содержит номер последней версии файла;
- *Security Descriptor* - этот атрибут содержит информацию о защите файла: список прав доступа ACL и поле аудита, которое определяет, какого рода операции над этим файлом нужно регистрировать;
- *Volume Version* - версия тома (только в системных файлах тома);
- *Volume Name* - отметка тома;
- *Volume Information* - номер версии NTFS;
- *Data* - содержит обычные данные файла;
- *MFT bitmap* - этот атрибут содержит карту использования секторов на томе;
- *Index Root* - корень B-дерева, используемого для поиска файлов в каталоге;
- *Index Allocation* - нерезидентные части индексного списка B-дерева;
- *External Attribute Information* - содержит номер первого кластера и количество кластеров нерезидентного атрибута;
- *Standard Information* - этот атрибут хранит всю остальную стандартную информацию о файле, например, время создания файла, время обновления...

Короткие имена

NTFS поддерживает имена файлов длиной до 255 символов. Имена файлов NTFS используют набор символов UNICODE с 16-битовыми символами. NTFS автоматически генерирует поддерживаемое MS-DOS имя для каждого файла. NTFS также позволяет приложениям MS-DOS и Windows (младших версий) работать с файлами, имеющими длинные имена NTFS.

Поскольку NTFS использует набор символов UNICODE для имен файлов, существует возможность использования некоторых запрещенных в MS-DOS символов. Для генерации короткого имени файла в стиле MS-DOS NTFS удаляет все запрещенные символы, точки (кроме одной), а также любые пробелы из длинного имени файла. Далее имя файла усекается до 6 символов, добавляется тильда (~) и номер. Например, каждому недублированному имени файла добавляется ~1, повторяющиеся имена файлов заканчиваются символами ~2, ~3 и т.д. Расширение имени файла усекается до 3 символов.

В файловой системе NTFS короткое имя образуется по алгоритму, когда за основу берутся не первые 6 символов длинного имени, а их эквивалент в UNICODE.

Например, для файла, длинное имя которого "Очень хороший файл. С расширением.gif", короткое имя будет выглядеть следующим образом: 9a10~1.gif