Тема No I3: «Файловая система FAT»

ОГЛАВЛЕНИЕ

Введение

FAT (англ. File Allocation Table) - «таблица размещения файлов». Файловая система разработана Биллом Гейтсом и Марком МакДональдом в I976-I977 годах.

Данная файловая система Использовалась в качестве основной файловой системы в операционных системах семейств DOS и Windows (до версии Windows 2000).

Существует четыре типа файловой системы FAT:

— FATI2. Год создания I980;

— FATI6. Год создания I987;

— FAT32. Год создания I996;

— exFAT[I]. Год создания 2006;

— FAT+[2]. Описание находится в состоянии draft;

Все файловые системы (за исключением exFAT) имеют очень схожую структуру и по факту наследуют друг от друга определённые решения. При этом данная тенденция справедлива как в прямую, так и в обратную сторону. Так в Windows 95 появилась надстройка над FATI2 и FATI6, называемая VFAT, которая включила в себя реализацию поддержки длинных имён файлов, предложенную в FAT32.

Для изложения материала будет рассматриваться образ носителя информации, созданный с пользованием утилиты mkfs.fat:

```
mkfs.fat floppy.img -s 2 -C I440
```

Для просмотра используется утилита hexedit.

```
hexedit -s floppy.img
```

I. Категории данных содержимого

К категории данных содержимого в файловой системе FAT относятся две сущности:

I. Сектор.

2. Кластер.

Сектор — минимальная совокупность смежных байт на цифровом носителе информации, которая может быть записана или прочитана за одно обращение к носителю. Условно говоря, если потребуется записать на носитель I байт, то он будет записан в составе блока данных большего размера.

[I]В рамках курса рассматриваться не будет!!!

[2]В рамках курса рассматриваться не будет!!!

Для того, чтобы получить размер сектора для файловой системы FAT, следует обратиться к II (0xB)и I2 (0xC) байтам тома. На рис.I видно, что размер сектора равен 5I2 (0x2000) байт.

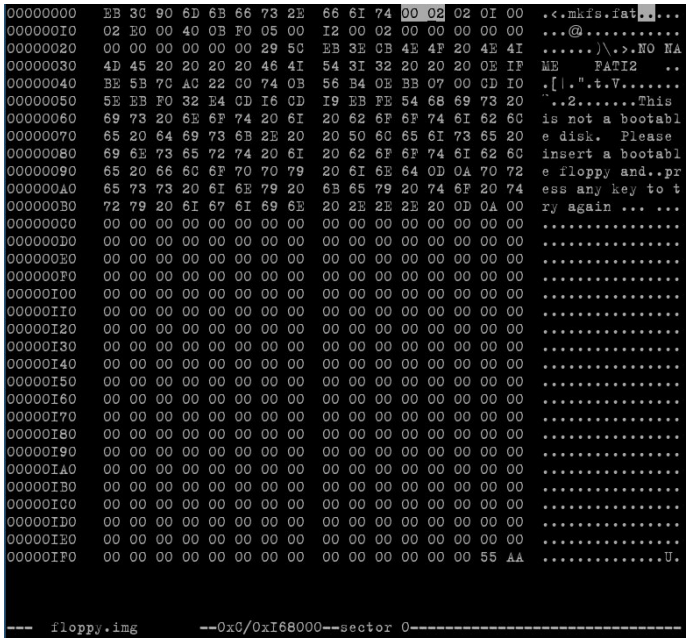


Рис I. Определение размера сектора

Следует отметить, что 5I2 байт в настоящее время является стандартным размером сектора для всех цифровых носителей информации. Однако данная ситуация таит в себе скрытую угрозу. Дело в том, что разработчики программного обеспечения зачастую игнорируют указанный параметр и жестко «вшивают» размер сектора в свои программные продукты. Таким образом, нестандартная разметка носителя может быть ими проигнорирована.

Второй сущностью, относящейся к данной категории, является кластер. Кластер — это совокупность смежных секторов на цифровом носителе. Чаще всего размер кластера соответствует степени двойки. Использование кластеров в качестве единицы разбиения носителя информации продиктовано стремлением увеличить допустимый к адресации объем носителя информации. Поскольку адресация в файловых системах осуществляет блоками, а максимальный размер адреса ограничен, то увеличение размера каждого блока позволяет увеличить общий объем адресуемых байт.

Учитывая специфику использования кластера, нетрудно догадаться, что в отличие от сектора размер кластера для каждого

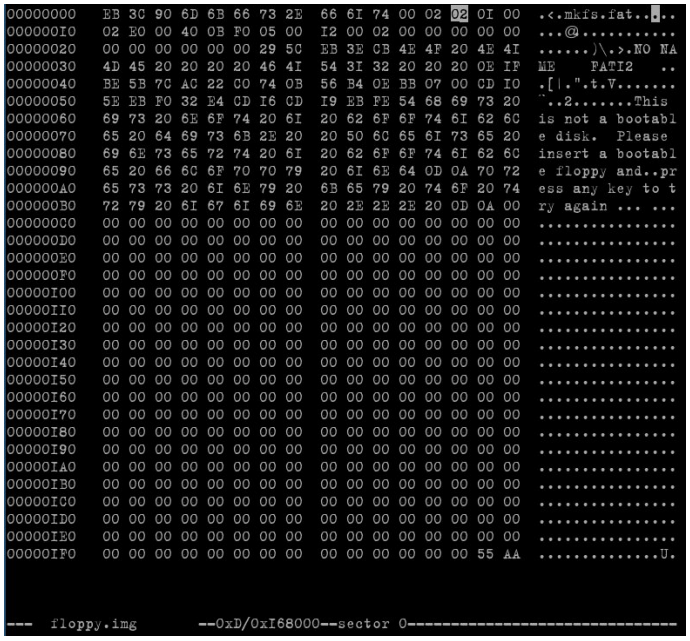


Рис 2. Определение размера сектора

носителя информации может быть свой. По этой причине, перед началом работы с файловой системой, необходимо определить размер кластера. Для этого достаточно обратиться к I3 (0xD) байту первого сектора. В рамках рассматриваемого примера размер кластера равен 2 сектора (рис.2).

Для носителя размером с дискету указанный размер кластера является нетипичным. Однако, с целью отделения кластера от сектора образ был сформирован именно так.

2. Метаданные FAT

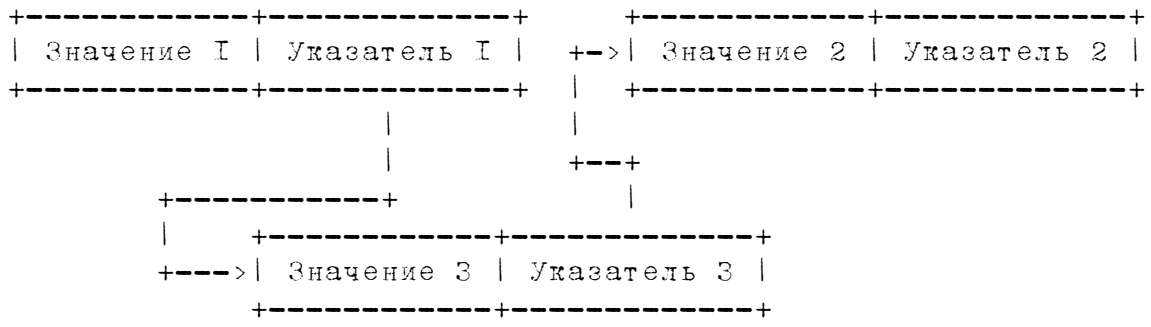
2.I. Структуры данных. Список.

Прежде чем преступить к рассмотрению категории метаданных файловой системы, следует немного осветить абстрактный тип данных — список .

Связный список — базовая динамическая структура данных в информатике, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки («связки») на следующий и/или предыдущий узел списка[3].

[3] https://ru.wikipedia.org/wiki/Связный_список

Файловая система FAT основана на использовании односвязных списков. Т.е. состоит из элементов, которые содержат два атрибута: значение и указатель.

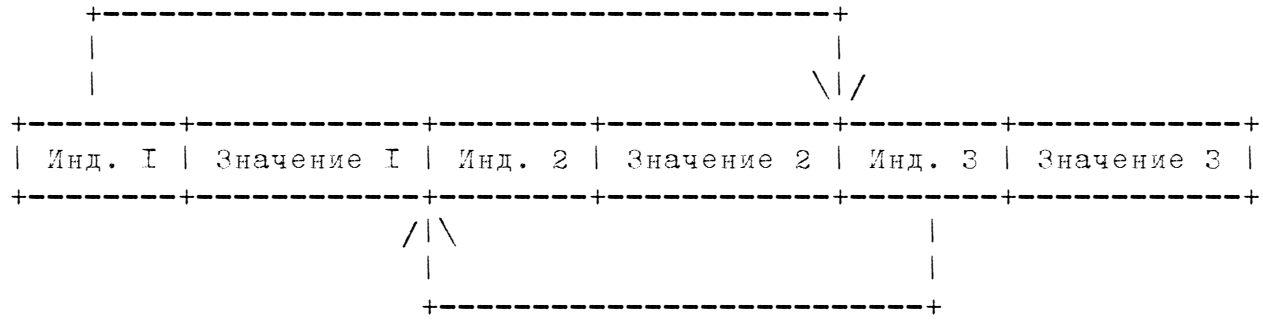


Пример однонаправленного списка.

Вместе с тем люди, которые начинают изучение этой файловой системы, не до конца понимают, где именно эти списки находятся и как реализуются.

В классическом представлении элементы списка могут располагаться в любой доступной области памяти процесса. В этом случае, в качестве указателя используется абсолютный адрес памяти. Такая реализация списка подходит для элементов, общее число которых изначально неизвестно.

В тех случаях, когда известно общее число элементов, которые следует хранить в списке, становится возможным выделением памяти сразу под все элементы и по мере необходимости задействовать их. В этом случае список может быть реализован в виде массива элементов, а адресом элемента будет не адрес памяти, а индекс в массиве.



Реализация списка в массиве.

Преимущество данной реализации списков относительно классического массива проявляется в тот момент, когда существует

необходимость хранения данных различного размера. В этом случае данные делятся на блоки размером в один элемент списка и аккуратно размещаются в нем. Поскольку ранее список мог использоваться, свободные блоки могут быть размещены не последовательно. По этой причине список заполняется в тех записях, которые помечены как свободные, а связь блоков реализуется за счет индексов. В классическом массиве вначале потребовалось бы произвести уплотнение данных...

Рассмотрим последнюю модификацию классического списка, прежде чем перейти вновь к рассмотрению файловой системы FAT. Итак, предположим, что максимальное число элементов известно. В этом случае список может быть реализован с использованием массива структур.

```
#include <stdio.h>
#define LIST_ITEM_COUNT (10)
struct list_s {
  int next;
  int val;
};

int main(int argc,char* argv[])
{
  strcut list_s list[LIST_ITEM_COUNT];
  int idx;
  /* ..... */
  printf("Item: %d\n",list[idx].val);
  idx = list[idx].next;
  printf("Item: %d\n",list[idx].val);
  /* ..... */
  return 0;
}
```

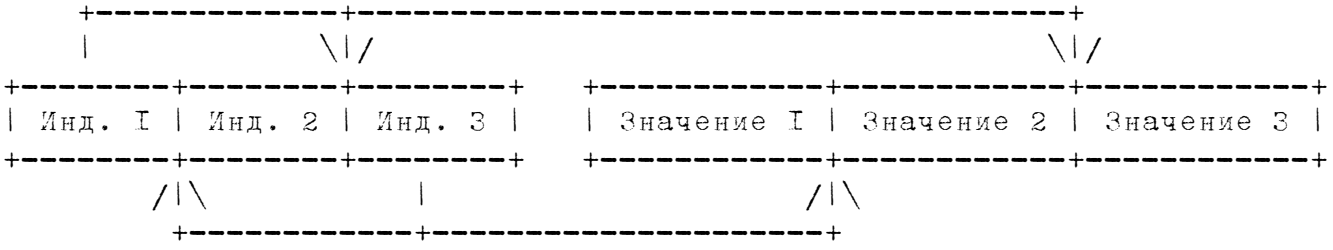
Как видно из кода, в действительности список реализован через два перемешанных массива: значений и индексов. Таким образом, структуру можно заменить двумя массивами. Тогда приведенный выше код может быть переписан следующим образом.

```
#include <stdio.h>
```

```
#define LIST_ITEM_COUNT (10)

int main(int argc,char* argv[])
{
  int next[LIST_ITEM_COUNT], vals[LIST_ITEM_COUNT], idx;
  /* ..... */
  printf("Item: %d\n",vals[idx]);
  idx = next[idx];
  printf("Item: %d\n",vals[idx]);
  /* ..... */
  return 0;
}
```

Графически это может быть представлено следующим образом.



Реализация списка в виде 2-х массивов.

Именно последний вариант реализации списка используется в файловой системе FAT. В качестве массива для хранения значений используется «область данных», разделенная на кластера. В качестве массива индексов используется таблица FAT.

2.2. Таблица FAT

Каждому файлу, расположенному на файловой системе FAT, соответствует цепочка кластеров. Данная цепочка реализована как односвязный список в табличке FAT3. При этом следует отметить, что размер одной записи таблицы в битах соответствует номеру в названии файловой системы. Таким образом, название файловой системы FATI2 может быть трактовано следующим образом: «Таблица размещения файлов с размером записи I2 бит».

Размер таблицы FAT зависит от числа кластеров, которые необходимо адресовать, а следовательно не может быть фиксированным. Размер таблицы можно определить, прочитав соответствующую запись

Рис 3. Вид начальных фрагментов для FAT различного типа.

в загрузочном секторе. Для файловых систем FATI2 и FATI6 число секторов, занимаемых таблицей FAT, хранится в 22 (0xI6) и 23 (0xI7) байтах загрузочного сектора (рис.6).

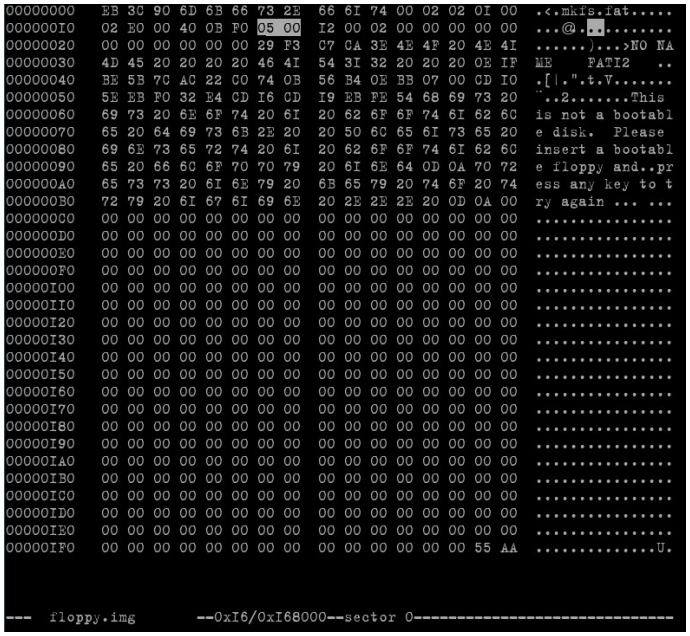


Рис 4. Определение размера таблицы FAT

Следует отметить, что первые две записи таблицы FAT имеют назначение, отличное от всех остальных. Изначально предполагалось, что первые 8 бит нулевой записи будет содержать некоторый идентификатор, а оставшиеся биты нулевой записи и первая запись целиком будет заполнены единицами[4]. При этом будет является идентификатором изначально определено не было. В последствии в качестве идентификатора стали использовать тип цифрового носителя (об этом позже). В дальнейшем слегка модифицировалось предназначения и некоторых оставшихся бит. Однако найти неких документ, похожий на стандарт и описывающий все эти изменения, не удалась. Так что они рассматриваться не будут

[4]http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-I07.pdf

В связи с тем, что первые две записи таблицы FAT зарезервированы, адресация к первым двум кластерам файловой системы невозможна. Следовательно, минимальный номер адресуемого кластера — 2.

Все оставшиеся записи таблицы, кроме непосредственно индекса следующего элемента списка, могут хранить значения, указанные в таблице ниже.

Значение кода	FAT12	FAT16	FAT32
Свободный кластер	0	0	0
Дефектный кластер	0x0FF7	0x0FFF7	0x0FFFFFF7
Последний кластер в списке кластер	0x0FF8 - 0x0FFF	0x0FFF8 - 0x0FFFF	0x0FFFFFF8 - 0x0FFFFFFF

Обратите внимание на значение «дефектный кластер». Это значение было введено в связи с низкой надежностью цифровых носителей прошлого. Дело в том, что через определенный промежуток времени они могли частично терять свои свойства, что приводило к частичной либо полной потере данных. По этой причине периодически рекомендовалось проводить проверку носителя. В ходе проверки в кластер записывалось некоторое значение, а потом считывалось. Если записанное и считанное значения совпадали, то кластер признавался хорошим. В противном случае, кластер помечался как дефектный. При дальнейшем использовании указанные кластеры системой игнорировались. Такой подход позволял продлить срок эксплуатации цифрового носителя информации.... пусть и в неком урезанном варианте.

Описанный выше механизм относится исключительно к области данных файловой системы. Для повышения надежности в части самой таблицы FAT, в файловой системе предусмотрено резервирование. Т.е. в файловой системе может существовать более одной таблицы. В случае повреждения основной таблицы копии мугут использоваться для работы. Общее число таблиц FAT указано в 16 (0x10) байте загрузочного сектора

(рис.5). Обычно предусматривается единственная копия, т.е. в системе присутствуют 2 таблицы.

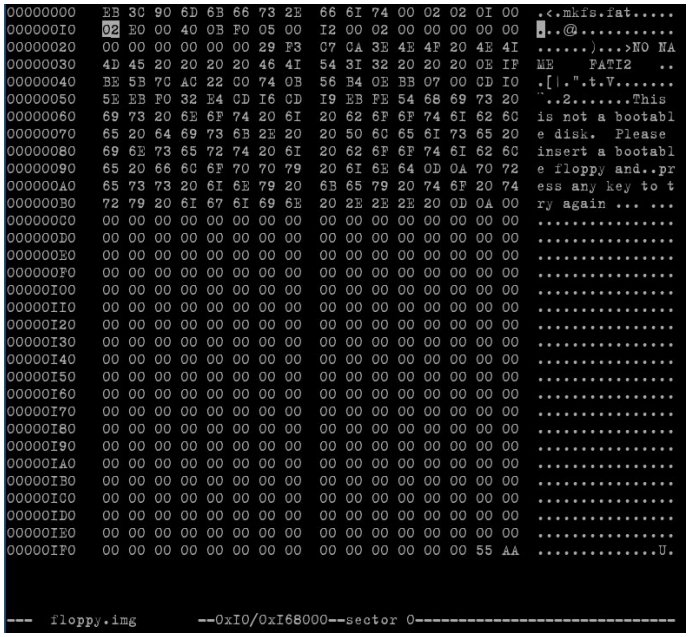


Рис 5. Определение числа таблицы FAT

2.3. Директория

Файловая система класса FAT является древовидной файловой системой. Это означает, что неотъемлемой частью данной файловой системы являются директории (каталоги).

Директория представляет собой массив 32-байтовых элементов — описателей файлов. С точки зрения ОС все директории (кроме корневой для FATI2 и FATI6) выглядят как файл и могут содержать произвольное количество записей.

Корневая директория (Root Dyrectory) - главная директория тома, с которого начинается дерево поддиректорий.

Для корневой директории в FATI2 и FATI6 выделено место в системной области тома. При этом число записей корневой директории жестко зафиксировано в I7 (0xII) и I8 (0xI2) секторах загрузочного сектора.

В файловой системе FAT32 корневой каталог является файлом произвольного размера.

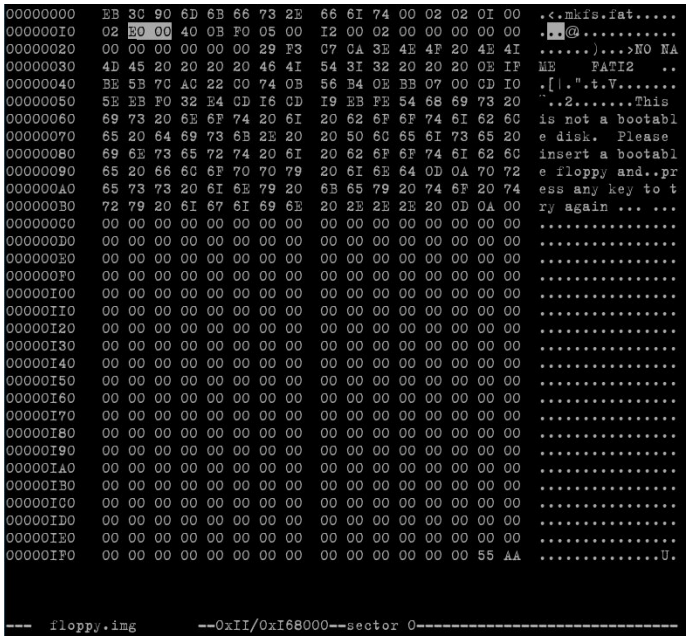


Рис 6. Определение количества элементов к корневой директории

Ниже в таблице представлена общая структура каталога для файловых систем FAT. Символом (*) выделены элементы, которые использующиеся только в FAT32. В FATI2 и FATI6 данные поля считаются зарезервированными и содержат значение 0.

Смещение	Размер	Описание
0x00	II	Короткое имя файла
0x0B	I	Атрибуты файла
0x0C	I	Зарезервированно для Windows NT (должна быть 0)
0x0D*	I	Поле уточняет время создания фала (содержит десятки миллисекунд) Значение поля может находиться в приделах [0,I99]
0x0E*	2	Время создания файла
0xI0*	2	Дата создания файла
0xI2*	2	Дата последнего обращения к файлу для записи или считывания данных
0xI4*	2	Старшее слово номера первого кластера файла
0xI6	2	Время выполнения последней операции записи
0xI8	2	Дата выполнения последней операции записи

0x1A	2	Младшее слово номера первого кластера файла
0x1C	4	Размер файла в байтах

Необходимо отметить, что первый символ имени файла также указывает на его существование. Первый символ имени файла может быть трактован одним из четырех способов:

- элемент каталога свободен (значение 0xE5);
- элемент каталога свободен и является началом чистой области каталога (значение 0x00);
- первый символ имени начинается с символа 0xE5 [5] (значение x05);
- символ не интерпретируется (любое значение, кроме первых трех).

Для хранения времени используется следующий формат:

- биты [0..4] — день;
- биты [5..8] — месяц;
- биты [9..15] — количество лет начиная с 1980 года;

Рассмотрим пример функции, получающей как параметр атрибут времени файла и выводящая его на экран. Код функции представлен в листинге 1.

Листинг 1. Разбор атрибута времени.

```
/* Output format "dd-mm-yyyy"*/
void f(unsigned short data)
{
  printf("%02d-",(data&0x1f));
  printf("%02d-",((data>>5)&0x0f));
  printf("%04d\n",1980+(data>>9));
}
```

Если в программе вызвать данную процедуру и передать ей значение 0x508a, то на экране появиться сообщение: "01-04-2020".

[5] для имен файла в кодировки Unicode

Как указывалось ранее, директория — это тоже файл, а, следовательно, программному модулю, который будет работать с данной файловой системой, необходимо понимать, чему соответствует та или иная запись. Для этого ему достаточно проанализировать значение атрибута записи каталога. Ниже в таблице приведены флаги, которые используются для определения атрибутов файла.

Значение		Описание
Двоичное	HEX	
0000 0001	0x01	Доступен только для чтения
0000 0010	0x02	Скрытый файл
0000 0100	0x04	Системный файл
0000 1000	0x08	Метка тома
0000 1111	0x0f	Длинное имя файла
0001 0000	0x10	Каталог
0010 0000	0x20	Архивный файл

Обратите внимание: атрибут длинного имени представляет собой поразрядную комбинацию первых четырех атрибутов. Компания Microsoft выяснила, что старые ОС[6] игнорируют записи каталогов со всеми установленными битами и введение нового атрибута не создаст проблем.

3. Категория данных имен файлов

Как уже могло стать понятным, в файловой системе FAT присутствуют два типа имен файлов:

— короткое имя;

— длинное имя.

Короткое имя файла имеет размер 11 байт и делится на две компоненты:

[6] скорее всего ее собственные

— имя файла (8 байт);

— расширение (3 байта).

Именно в следствии указанного разбиения формат короткого имени файлов еще называют форматом «8+3». При этом не обязательно, чтобы имя или расширение имели указанный размер. Если имя или расширение короче отведенного для них объема записи, то недостающие символы заполняются пробелами.

Начиная с Windows 95, для файловых систем FAT появилась возможность сохранять файлы с именами, превышающими размер 8+3. Для этого были введены специальные типы записей каталогов, позволяющие хранить длинные имена файлов. Отметим, что длинные имена файлов изначально появились в файловой системе FAT32. Позже, в связи хорошей обратной совместимостью версий, они стали использоваться и в FAT12 и FAT16.

Каким образом это стало возможным. Дело в том, что при выставлении атрибута записи директории «Длинное имя файла», запись начинает интерпретироваться следующим образом.

Смещение	Размер	Описание
0x00	1	Номер фрагмента
0x01	10	Первый участок фрагмента имени
0x0B	1	Атрибуты файла
0x0C	1	Зарезервировано
0x0D	1	Контрольная сумма короткого имени (одинаковое значение для всех элементов файла)
0x0E	12	Второй участок фрагмента имени
0x1A	2	Номер первого кластера (должен быть равен 0)
0x1C	4	Третий участок фрагмента имени

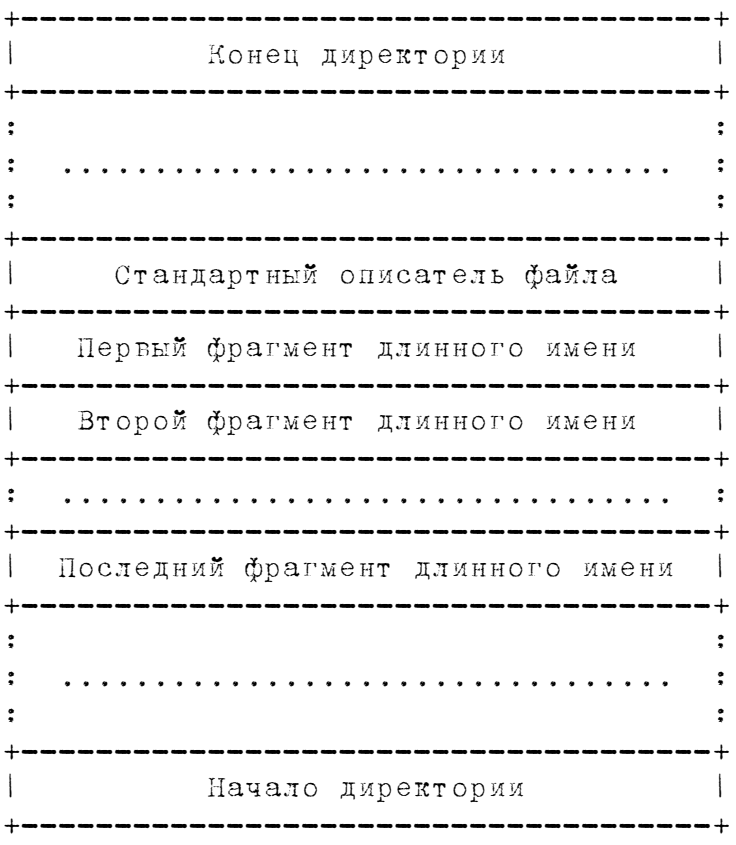
Поле «контрольная сумма» содержит значение, полученное в результате контрольного суммирования короткого имени файла. В листинге 2 приведена функция подсчета этой контрольной сумму.

Листинг 2. Подсчет контрольной суммы короткого имени файла.

```
c = 0;
for ( i = 0 ; i < 11 ; i++ ) {
  /* Shift right */
  c = ((c & 0x1)?0x80:0)+(c>>1);
  /* add ASCII-code*/
  c = c + shortname[i];
}
```

Если значение в данном поле не верно, то это означает, что том использовался ОС, не имеющей поддержку длинных имен файлов, и возможно данные повреждены.

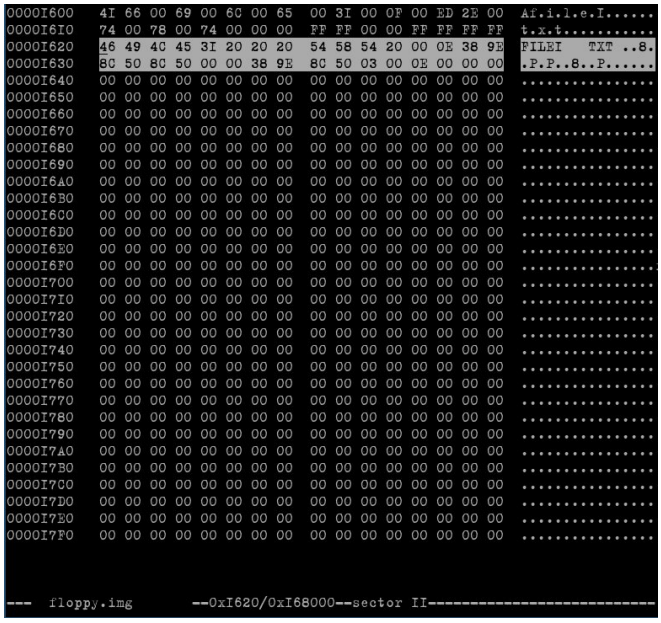
Следует отметить, что записи с длинным именем файла размещаются не в произвольном порядке, а следуют непосредственно перед основной записью о файле.

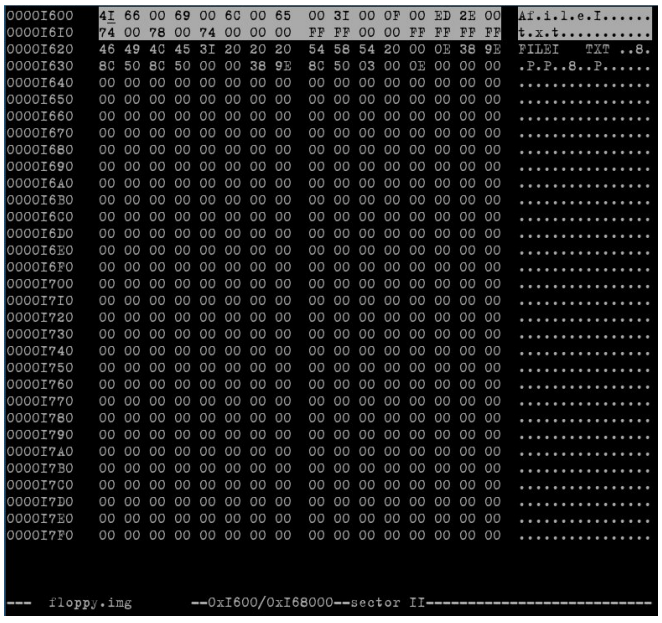


Следует отметить, что несмотря на теоретическую возможность создания файла с достаточно длинным именем, в документации на файловую систему FAT указывается, что максимальная длина имени файла не должна превышать 255 байт.

Еще одной особенностью записи о длинном имени файла является то, что номер последнего фрагмента хранится в модифицированном виде, а именно — побитово просуммированный со значением 0x40.

Ниже показано, каким образом в директории размещается информация о файле, с учетом записи о длинном имени файла.





Основная запись директории Длинное имя файла

4. Категория данных файловой системы

4.I. Загрузочный сектор

В первом секторе тома FAT располагается загрузочный сектор, частью которого является блок параметров BIOS (BIOS Paramert Block — BPB). В давние времена BPB использовался для определения конфигурации цифрового носителя информации. В настоящее время указанная конфигурация определятся иными методами. В основном за счет запроса к контроллеру соответствующего устройства.

Первые 36 байтов загрузочного сектора, приведенные ниже, идентичны для всех рассматриваемых версий FAT.

Смещение	Размер	Описание
0x00	3	Инструкция перехода (jmp) на загрузочный код.
0x03	8	Текстовая строка с атрибутами фирмы (OEM).
0x0B	2	Число байт в секторе (всегда 512).
0x0D	1	Число секторов в кластере.
0x0E	2	Число резервных секторов в резервной области раздела, начиная с первого сектора раздела.
0x10	1	Число таблиц (копий) FAT.
0x11	2	Для FAT12 и FAT16 - количество 32-байтовых дескрипторов файлов в корневом каталоге; Для FAT32 это поле имеет значение 0.
0x13	2	Общее число секторов в разделе.
0x15	1	Тип носителя
0x16	2	Для FAT12 и FAT16 - количество секторов, занимаемое одной копией FAT; для FAT32 поле имеет значение 0.
0x18	2	Число секторов на дорожке.
0x1A	2	Число головок.
0x1C	4	Число скрытых секторов, предшествующих разделу, содержащему данный том. Значение равно 0 для носителей, не подлежащих разбиению на разделы.
0x20	4	Общее число секторов в разделе.

Стоит отметить, что поле 0x20 используется вместо поля 0x13, если в разделе свыше 65535 секторов. В противном случае данное поле содержит значение 0.

Поле 0x15 отвечает за определение типа носителя. Об этом поле уже упоминалось выше, когда речь шла о нулевой записи таблицы FAT. Данное поле может принимать одно из следующих значений.

1. 0xF0 — гибкий диск, 2 стороны, 18 секторов на дорожке.

2. 0xF8 — Жесткий диск.

Начиная со смещения 0x24, структура загрузочного сектора для FAT16 (FAT12) и FAT32 различаются. Для FAT16 (FAT12) он имеет следующую структуру.

Смещение	Размер	Описание
0x24	1	Номер диска для прерывания 13h.
0x25	1	Зарезервировано для Windows NT, имеет значение 0.
0x26	1	Признак расширенной загрузочной записи (0x29). Показывает, что следующие три поля присутствуют.
0x27	4	Номер логического диска.
0x2B	11	Метка диска (Текстовая строка).
0x36	8	Текстовая строка с аббревиатурой файловой системы.

Структура загрузочного сектора FAT32 имеет больше полей, однако, начиная с определенного смещения, она дублирует структуру загрузочного сектора FAT16 (FAT12).

Смещение	Размер	Описание
0x24	4	Количество секторов, занимаемый одной копией FAT.
0x28	2	Номер активной FAT.
0x2A	2	Номер версии FAT32 (major:minor).
0x2C	4	Номер первого кластера корневого каталога.
0x30	2	Номер сектора в резервной области логического раздела, где хранится структуры FSINFO.
0x32	2	Номер сектора в резервной области логического диска, используемый для хранения резервной копии загрузочного сектора.
0x34	12	Зарезервировано (должна быть 0)
0x40	1	Номер диска для прерывания 13h.
0x41	1	Зарезервировано для Windows NT, имеет значение 0.
0x42	1	Признак расширенной загрузочной записи (0x29). Показывает, что следующие три поля присутствуют.

Смещение	Размер	Описание
0x43	4	Номер логического диска.
0x47	11	Метка диска (Текстовач строка).
0x52	8	Текстовая строка с аббревиатурой файловой системы.

Все оставшееся пространство загрузочного сектора занимает код загрузчика. В тех случаях, когда для хранения загрузчика оставшегося объема не хватает, при форматировании носителя выделяются дополнительные сектора, которые входят в область зарезервированных секторов.

4.2. FSInfo

Если обратить внимание на различия в полях загрузочного сектора, то можно заметить, что размер адресов в FAT32 по сравнению в FATI6 (FATI2) увеличился в два раза. Подобное увеличение повлекло за собой экспоненциальный рост сложности работы с объектами файловой системы. В файловых система FATI6 (FATI2) для поиска свободного кластера использовался метод тотального перебора. Т.е., начиная со второй, просматривались все записи таблицы с целью поиска свободных кластеров. Максимальное число опробований в таком случае не превышало 4096 для FATI2 и 65536 для FATI6. В FAT32 требовалось проверить более 4 млн. записей.

Для оптимизации работы в файловой системе FAT32 появилась специальная структура, получившая название FSInfo. Данная структура размещается в резервных секторах файловой системы и состоит из следующих полей.

Смещение	Размер	Описание
0x000	4	Сигнатура: 0x4I625252
0x004	480	Зарезервировано (должна быть 0)
0xIE4	4	Сигнатура: 0x6I4I7272
0xIE8	4	Текущее число свободных кластеров на диске.
0xIEC	4	Номер кластера для начала поиска свободных кластеров

0xIF0	12	Зарезервировано (должна быть 0)
0xIFC	4	Сигнатура: 0xAA550000

С ее появлением алгоритм поиска очередного свободного кластера начинался со значения, сохраненного в поле 0xIE8. При этом, используя данные следующего поля, можно было понять сколько еще свободного пространства осталось. После записи очередной партии данных, значения этих полей соответствующим образом изменялись. Следует отметить, что кластеры, освободившиеся в начале раздела, оставались нетронутыми до тех пор, пока было достаточно кластеров в конце раздела.

В том случае, когда в конце раздела заканчивались кластеры, значения полей 0xIE8 и 0xIEC устанавливались равными 0xFFFFFFFF, и система для поиска свободных кластеров начинала использовать метод перебора.

В этот момент наблюдалось значительное снижение производительности систем. Решением проблемы в данном случае становилось дефрагментирование носителя, в рамках которого производилось уплотнение занятых кластеров и «вытеснение» свободных в конец раздела. На последнем этапе соответствующие поля FSInfo снова становились пригодными дя использования.

5. Основные характеристики файловой системы

После рассмотрения основных компонентов файловый систем FAT можно оценить ограничения, накладываемые на их использование.

Критерий	FAT12	FAT16	FAT32
Максимальная длина имён	8+3 символов		255 байт
Допустимые символы в имени	ANSI кроме NULL		Юникода кроме NULL
Максимальная длина пути	Нет установленных ограничений		
Максимальный размер файла	32Мб	2Гб	4Гб

Максимальный размер тома	1Мб - 32Мб	16Мб - 2Гб	512Мб - 8Тб

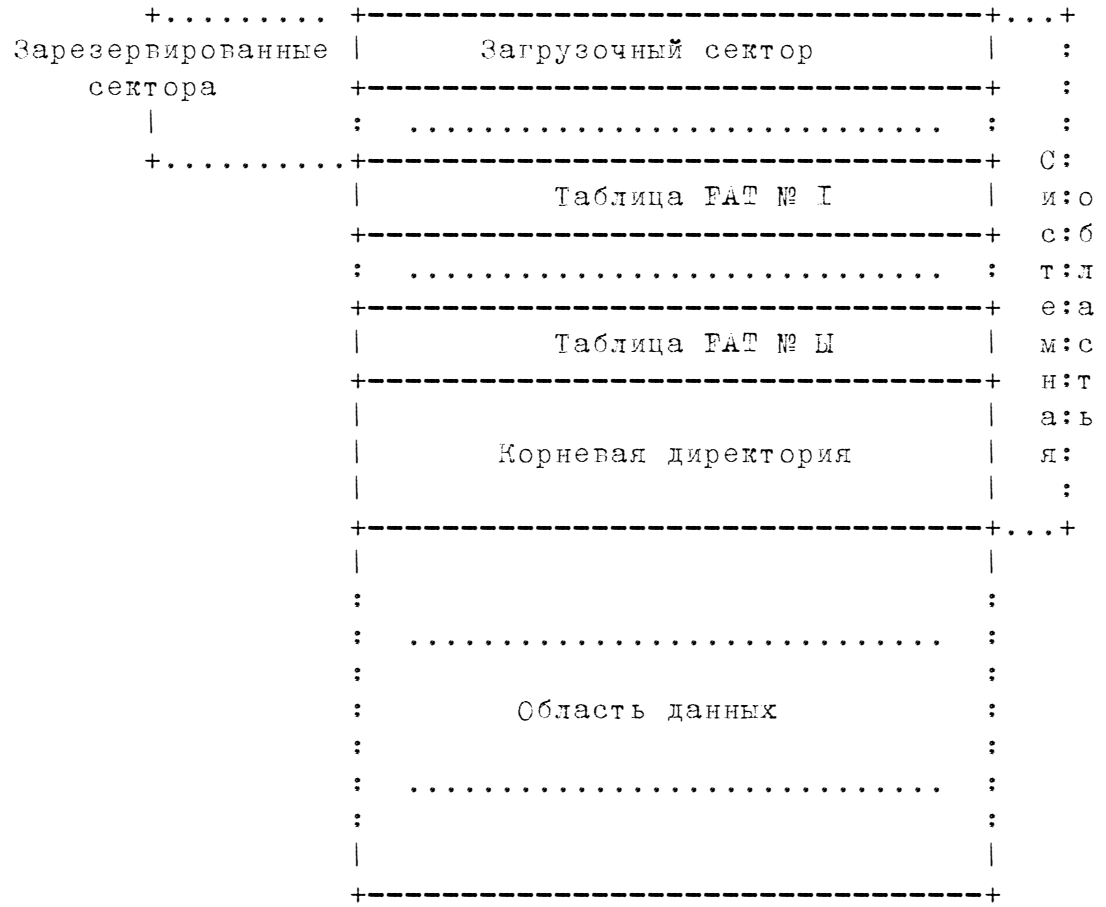
Несмотря на приведенные показатели, корпорация Microsoft в спецификации к FAT32 определяет максимальный размер тома — 32Гб.

С целью недопущения создания томов большего объема программное обеспечение Microsoft не позволяет производить форматирование носителей, объем которых превышает выше обозначенный. При этом если произвести форматирование подобного носителя сторонними средствами, операционная система Windows будет с ними корректно взаимодействовать.

6. Связь элементов файловой системы

6.1. FAT12 / FAT16

Структуры FAT12 и FAT16 идентичны с точностью до замены размера записи таблицы FAT. Общая структура этих файловых систем приведена ниже.



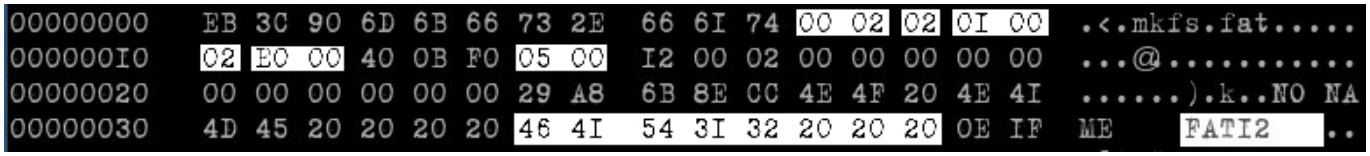


Рис 7. Кусок загрузочного сектора FATI2

Вот получил вы полезные нужные знания![7] Теперь, попробуем ими воспользоваться, а именно: вручную получим информацию о файле, хранящемся на образе цифрового носителя.

Для начала определим геометрию системной области. Для этого необходимо проанализировать первые 62 байта загрузочного сектора (рис.7).

Учитывая инвертированые области на картинке, получаем следующую информацию о носителе информации и файловой системе.

I. Размер сектора: 5I2 (0x200) байт.

2. Размер кластера: 2 сектора => I024 байта.

3. Количество зарезервированных секторов: I штук (загрузочный сектор).

4. Количество таблиц FAT: 2 штуки.

5. Количество записей в корневой директории: 224 (0x00E0) записи.

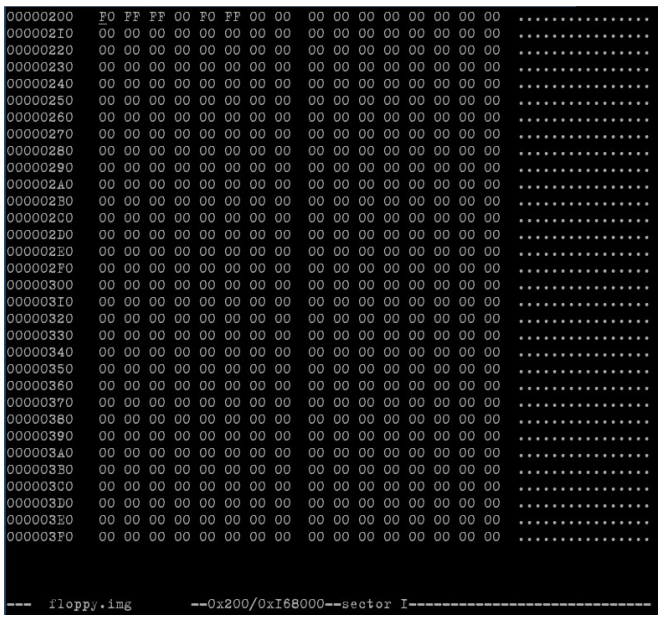
6. Размер одной таблицы FAT: 5 секторов => 2560 байт.

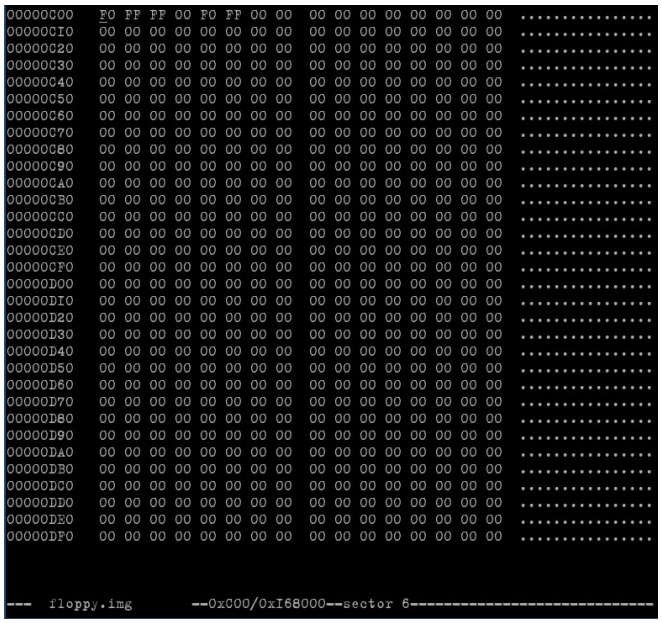
7. Тип файловой системы: FATI2.

Теперь начнем вычислять дополнительные значения. Во-первых, определим положение двух таблиц FAT. В соответствии с общей схемой раздела, первая таблица располагается непосредственно за последним зарезервированным сектором. В нашем случае, такой сектор единственный, следовательно первая таблица файлов размещается в первом секторе, либо по смещению 5I2 (0x200) байт от начала раздела. Вторая таблица располагается непосредственно за первой, что к адресу начала первой таблице необходимо прибавить ее размер и получить

[7]....

адрес второй. В нашем случае, вторая таблица располагается в 6 (I+5) секторе, либо по смещению 3072 (0xC00) байта от начала раздела.





Первая таблица FAT Вторая таблица FAT

Аналогичным образом можно получить информацию о расположении корневой директории. Для этого достаточно к адресу второй таблицы FAT прибавить размер таблицы. В следствии указанной операции получаем, что корневая директория располагается начиная с II (6+5) сектора, либо по смещению 5632 (0xI600) байта от начала раздела.

Теперь перейдем непосредственно к анализу записей корневой директории (рис.8).

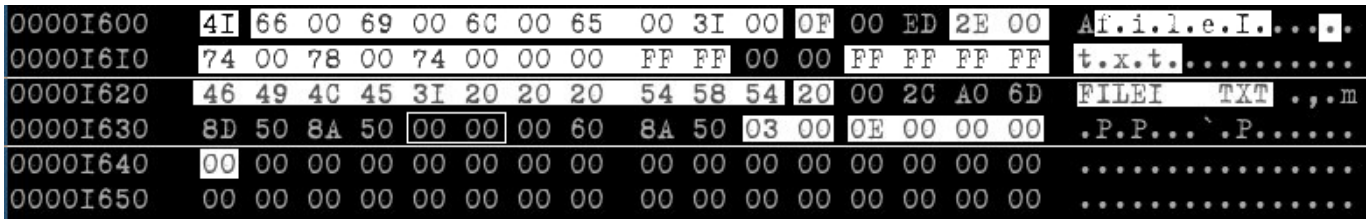


Рис 8. Первые три записи корневого каталога

Несложно заметить, что первая запись корневой директории начинается с записи о длинном имени файла. О об этом свидетельствует значение атрибута 0x0F. Первый байт записи (0x4I) раскладывается на две составляющие, обозначающие что «это последний из фрагментов» (0x40), а его порядковый номер I (0xI). Таким образом, все имя

файла поместилось в одну запись. Далее не сложно прочитать, что файла называется «fileI.txt».

Вторая запись директории, судя по значению атрибута, соответствует архивному файлу (не путать с архивированным). Таким образом, первые II байт записи соответствуют короткому имени файла («FILEI.TXT»).

Поскольку в данном примере используется файловая система FATI2 поле «Старшее слово номера первого кластера файла» равно нулю и не учитывается. В случае с FAT32 значение данного боле необходимо было бы использоваться в качестве старших байт четырехбайтового номера кластера. Таким образом, первый кластер, в котором размещаются данные файле - кластер № 3. Если обратиться к соответствующей записи в таблице FAT, то там можно найти код завершения цепочки кластеров.

```
    0       I       2       3
+-------+-------+-------+-------+.....
| 0xff0 | 0xfff | 0x000 | 0xfff |
+-------+-------+-------+-------+.....
```

Таким образом, все данные файла размещены в одном кластере. Это не удивительно, ведь размер файла всего I4 (0x0E) байт.

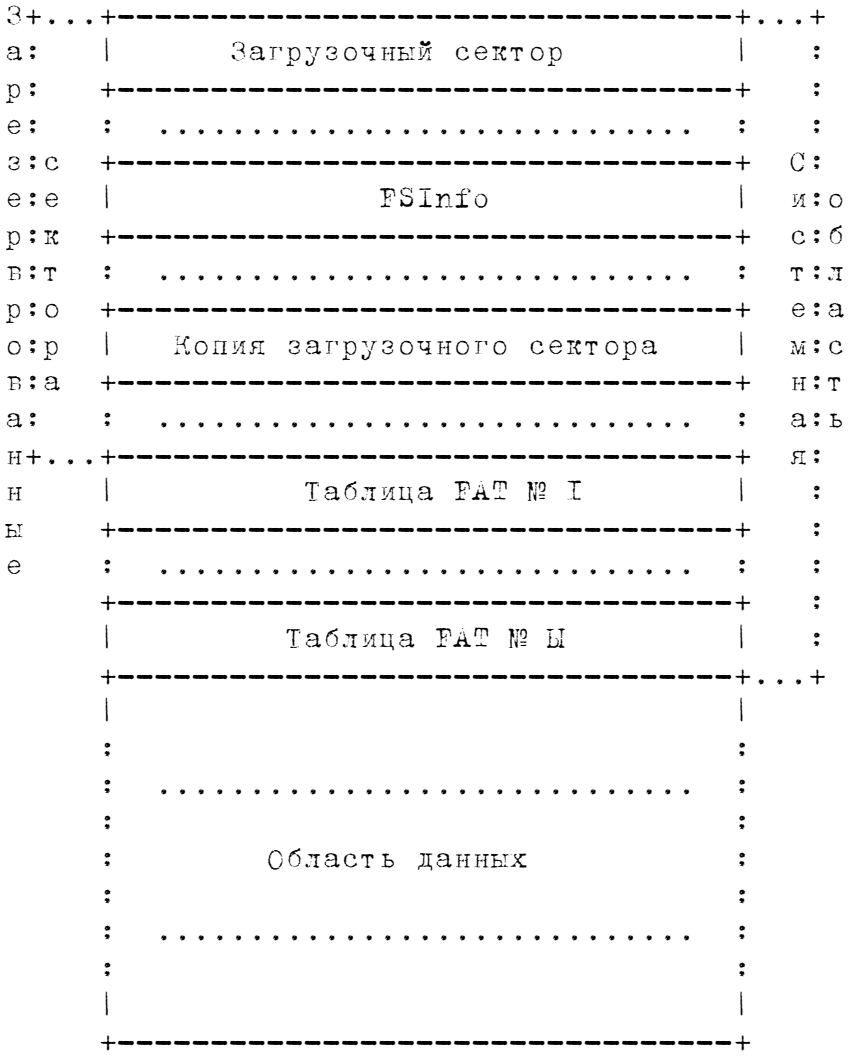
Последним шагом на пути к содержимому файла является определение положения области данных. Для того, чтобы узнать, где начитается область данных, необходимо к адресу начало корневой директории прибавить ее размер. Размер корневой директории определяется путем перемножения количества записей в директории на размер одной записи. Т.е. 224 * 32 = 7I68 (0xIC00) байт, либо I4 секторов.

Таким образом, область данных начинается с 25 (II+I4) сектора, либо со смещения I2800 (0x3200) байт от начала раздела. Поскольку файл размечается в третьем кластере, то от начала области данных следует отступить еще один кластер. Следовательно, данные файла находятся в 27 секторе (25+2), либо по смещению I3824 (0x3600) от начала раздела.

```
00003600   48 65 6C 6C 6F 2C 20 77  6F 72 6C 64 2I 0A 00 00  Hello, world!...
000036I0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

6.2. FAT32

Общая структура носителя информации с файловой системой FAT32 имеет следующую структуру.



Поскольку корневая директория находится в области данных, то под ее хранение может выделять произвольное число кластеров. По этой причине чтение корневой директории требует дополнительного анализа таблицы FAT.

Подробнее процесс поиска и чтения файлов с файловой системы FAT32 будет оставлен на самостоятельную проработку.

7. Процесс использования файловой системы FAT

7.I. Создание файла

Ранее была рассмотрена операция чтения файла. В данном разделе рассматривается вопрос создания файлов.

Рассмотрим процесс создания файл с имение «X:\dir\I.txt», данные которого должны размещаться в двух кластерах.

Тогда запись файла будет происходить по следующему алгоритму:

I. Чтение загрузочного сектора.

2. Получение информации о корневой директории и таблицах FAT.

3. Чтение корневой директории и поиск записи соответствующей директории «dirI».

4. Чтение записей директории «dirI» с целью поиска свободной записи (непрерывной группы записей, если необходимо записать длинное имя файла).

5. Заполнение найденной записи (всех полей кроме адреса начального кластера).

6. Поиск свободного кластера в таблице FAT и запись его номера в запись каталога «dirI».

7. Размещение в найденном кластере первой части файла.

8. Поиск следующего свободного кластера и занесение его номера в поле таблицы, соответствующей предыдущему найденному кластеру.

9. Запись оставшихся данных на диск.

IO. Запись в поле таблицы, соответствующей последнему найденному кластеру, значения «Последний кластер».

II. Синхронизация записей во всех таблицах.

Таким же образом осуществляется создание директорий.

7.2. Удаление файла

Рассмотрим алгоритм удаления файл с имение «X:\dir\I.txt».

1. Чтение загрузочного сектора.
2. Получение информации о корневой директории и таблицах FAT.
3. Чтение корневой директории и поиск записи соответствующей директории «dirI».
4. Чтение записей директории «dirI» с целью поиска всех записей соответствующих файлу «I.txt».
5. Определив номер первого кластера, заполнение всех элементов цепочки кластеров в таблице FAT значениями «Свободный кластер».
6. Изменение первого байта всех записей директории, соответствующих файлу, значением 0xe5.
7. Синхронизация записей во всех таблицах.

Следует обратить внимание, что при штатном удалении файлов изменения затрагивают исключительно метеоданные файловой системы и не затрагивают данные самого файла.

7.3. Восстановление данных

Если рассмотреть процесс удаления файла, описанный выше, нетрудно заметить, что в файловой системе частично остается информация об удаленном файле. К этой информации относятся:

1. Частичное имя файла.
2. Длина файла.
3. Атрибуты.
4. Номер первого кластера.

Зная данную информацию без труда можно восстановить файл размером не превышающим размер кластера[8].

Для этого нужно просто изменить первый символ короткого имени файла на значение, отличное от 0xE5 и 0x00, а в записи таблицы FAT, соответствующей первому кластеру файла, установить значение «Последний кластер».

При этом если для файла присутствовали записи о длинном имени файла, то, используя контрольную сумму из этих записей, можно полностью восстановить короткое имя файла.

Если размер файла превышает размер одного кластера, то восстановление файла становиться проблематичным. Причиной тому является необходимость восстановления цепочки кластеров в таблице FAT. Для решения данной проблемы подходит один из следующих подходов:

1. Включать в цепочку все кластеры подряд.
2. При восстановлении цепочки пропускать занятые кластеры.
3. При восстановлении цепочки пропускать кластеры, содержимое которых по статистическим и сигнатурным характеристикам не соответствуют восстанавливаемому файлу.

[8]Если поверх данных удаленного файла не было записанно других данных