I. Дискеты и их характеристики

…описание России, в котором поверхностный автор-француз пишет, что сидел под тенью величественной клюквы.

—Толковый словарь русского языка (под ред. Д.Н.Ушакова, т.1. С.1379)

I.1. Устройство гибкого магнитного диска

Изучай все не ради тщеславия, а ради практической пользы.

—Г.К.Лихтенберг. Афоризмы

Дискеты стали самым распространенным носителем программ и данных для персональных компьютеров. Они являются идеальным носителем для внешней памяти персональных компьютеров. Размеры дискет удобны, цена не высока и они достаточно надежны в эксплуатации (при известной осторожности в обращении дискеты очень редко портятся).

Сегодня невозможно представить себе работу на компьютере без этой небольшой круглой пластины из специального гибкого пластика, покрытого магнитным слоем. А ведь история этого носителя информации началась сравнительно недавно. Первые дискеты появились в начале 7О-х годов и в связи со своими многочисленными достоинствами начали быстро вытеснять другие методы регистрации компьютерной информации.

До эпохи гибких дисков безраздельно царили магнитные ленты, имевшие один крупный недостаток, — большое время доступа к информации. Проблема быстрого доступа к нужным данным хорошо знакома каждому, кто имел дело с двумя устройствами: магнитофоном и его противоположностью в рассматриваемом аспекте — граммофоном. Именно идеей структуры грампластинок воспользовались создатели дисководов и самих гибких дисков.

В дисководе дискета вращается со скоростью 300-360 оборотов в минуту, а головка чтения /записи перемещается в радиальном направлении. В совокупности это позволяет почти немедленно достичь любой точки на рабочей поверхности диска!

Всюду далее слова: «диск» , «дискета» , «микро-дискета» и «флоппи–диск» мы будем рассматривать как синонимы.

Кратко рассмотрим, что собой представляет *дискета*.

Дискета — круглый «кусок» гибкого пластика, покрытый магнитным окислом, напоминающим покрытие других известных магнитных носителей, например, магнитных лент. Магнитные диски, использующиеся на больших компьютерах, изготавливаются из жестких металлических пластин, а для микродискет используются гибкие пластиковые кружки, что и дало им популярное название «гибкие» или «флоппи–диски». То, что эти диски были сделаны гибкими, значительно уменьшило вероятность их повреждения при обращении с ними и это в значительной мере определило их успех.

Гибкие диски выпускаются главным образом двух размеров: диаметром 5.25 дюйма (около 13 см) и 3.5 дюйма (около 9 см). В компьютерах старых типов встречаются 8-дюймовые дискеты (около 20см). В некоторых компьютерах (в том числе Amstrad и Spectrum) применяются дискеты диаметром 3 дюйма, на выставках и ярмарках можно встретить дискеты диаметром 2 и 2.5 дюйма. В зависимости от качества магнитного слоя (основой которого является окись железа) данные могут записываться с одной или с обеих сторон диска. Это зависит также от типа НГМД (накопителя на гибких магнитных дисках), установленного в компьютере. От конструкции драйвера зависит также количество записываемой на дискету информации (двухсторонний флоппи-диск 3.5 дюйма может содержать до 1.44 Мбайт информации, т.е. около 720 машинописных страниц).

Круглый диск с магнитным покрытием всегда помещен в квадратный предохранительный конверт. Внутренняя поверхность этого конверта покрыта слоем белого фетро–подобного материала, помогающего в защите дискеты. Он служит как для смягчения ударов, так и для улавливания пыли.

Квадратный предохранительный конверт имеет *три отверстия*, каждое из которых имеет свое собственное назначение.

- *Первое* отверстие (в центре дискеты) предназначено для захвата диска приводом дисковода. Через это отверстие механизм дисковода захватывает гибкий диск, чтобы привести его во вращение.
- *Второе* отверстие в предохранительном конверте представляет собой продолговатую прорезь, закрытую металлической шторкой, через которую осуществляется доступ к дискете головок чтения/записи. Через эту прорезь происходит процесс чтения и записи информации.
- *Третье* отверстие — квадратная прорезь в углу корпуса дискеты — служит в качестве признака защиты от записи. Практически все магнитные носители используют те или иные признаки защиты от записи.

Если эта прорезь открыта, то дискета защищена от записи, если же она закрыта, то на эту дискету запись разрешена.

Не следует однако считать, что, открыв прорезь защиты записи, Вы полностью защитите свои дискеты от случайного стирания. Открытая про резь говорит только о том, что *правильно работающий дисковод* не будет *пытаться выполнять запись на такую дискету*. Неисправный дисковод будет делать все, что угодно, независимо от наличия или отсутствия признака защиты. Конечно, маловероятно утратить данные именно таким образом, но все же это может произойти.

Пластиковый диск покрыт магнитным слоем толщиной 2.2–2.9 микрометра. Он помещен между двумя слоями мягкой прокладки, по скользит во время вращения, и вместе с ними вложен в жесткий конверт. Отверстие для головки закрыто специальным щитком, а маркер защиты от записи встроен в конверт. Открытое отверстие маркера блокирует запись.

Продолжительность работы дисков в значительной степени зависит от нашей осторожности и правильного использования. Главные враги дисков — это грязь и пыль, а также табачный дым. За 2–3 года в дисководе может накопиться столько пыли, что нормальные диски перестанут читаться. Такое положение можно исправить только разобрав и вычистив дисковод (эта операция выполняется мастером). Для поддержания чистоты *головок* дисковода и продления жизни дисков применяются специальные чистящие диски. Они покрыты материалом, который снимает с головок эксплуатационные загрязнения. Очистку стоит производить 3-4 раза в месяц.

Записанную на диске информацию можно утратить, даже не заметив этого, если дискета будет подвергнута воздействию магнитного поля трансформаторы, телефоны, телевизоры, электродвигатели). Следует соблюдать осторожность, проходя через металлоискатели в аэропортах. В этом случае постарайтесь передать дискеты на ручной контроль.

Предлагаем несколько полезных советов:

1. в дискетах не предусмотрена идеальная защита записанных на них данных: защитные конверты рассчитаны только на предохранение поверхности носителей от повреждений на коротком пути от «дискотеки» до дисковода.

Во избежание порчи информации, записанной на дискетах, последние следует хранить подальше от телевизоров, видеомониторов, звонящих телефонов и других источников магнитных полей.

2. для защиты файлов, сохраненных на дискете, позаботьтесь о создании копий файлов на другой дискете или на магнитной ленте;
3. создайте архив файлов и снабжайте дискеты архива этикетками;
4. защищайте Ваши программы от неосторожного обращения путем установки переключателя защиты от записи на дискете.

Правильно эксплуатируемый диск выдерживает несколько миллионов проходов головки по дорожке, что означает несколько месяцев непрерывной работы на одной дорожке, а ведь таких дорожек на дискете 80! Дискеты высокого качества известных и опытных изготовителей гарантируют в сред нем 70 млн. проходов головки по дорожке, что на практике сводится к более чем 20–летней интенсивной эксплуатации. Таким образом, вероятнее ожидать аварии головки или других механизмов драйвера. Практика показывает, что иногда дискеты имеют повреждения, и чаще всего это случается с безымянными изделиями, т.е. с дешевыми дисками неизвестных фирм. Давно известно, что экономить на носителях информации не следует, и надо покупать только высококачественные изделия (таких фирм,

как SONY, BASF, JVC, Atari, IBM, Kodak).

Во время записи головка намагничивает микроскопические частички железа магнитного слоя и они приобретают одинаковую полярность. Для записи одного бита намагничивается отрезок дорожки длиной 0.004 мм и глубиной 0.015 мм. Такой магнитный бит сохраняет стабильность практически вечно, хотя теоретически можно представить себе изменения, вызванные неожиданной дестабилизацией магнитного поля Земли.

Способ записи данных на дискете зависит от вида дисковода, от его контроллера, а также от операционной системы, под контролем которой работает ЭВМ. Что же это такое — контроллер НГМД?

Это способ записи на дискету двоичного кода. Опишем два самых распространенных способа записи: FM (frequency modulated) и MFM (modified frequency modulated).

В записи FM соседние биты данных всегда отделены импульсом синхронизации. Благодаря этому снижаются требования к качеству магнитного носителя. К сожалению, снижается и объем хранимой на нем информации. Такой способ записи называют одинарной плотностью записи.

Удвоение этого объема возможно при записи MFM, где импульс синхронизации записывается на диск только тогда, когда два бита данных подряд имеют нулевое значение. Именно этот контроллер используется в стандарте MSX, а также на других распространенных компьютерах: IBM PC, Amstrad, Atari ST. Этот способ записи называют двойной плотностью записи. Ясно, что плотность записи полностью зависит от типа контроллера, применяемого в дисководе. Существует еще один способ записи (учетверенная плотность) — GCR (Group Code Recording). Он позволяет еще более увеличить плотность записи и применяется в драйверах для домашних компьютеров фирмы Commodore.

Дискеты бывают двух типов — *односторонние и двухсторонние*, хотя и в том и в другом случае *магнитный слой есть на обеих сторонах дисков!* Чем же они отличаются? Как правило, технология изготовления дискет предусматривает проверку качества магнитного слоя. Дискеты, контроль которых с обеих сторон показал полное отсутствие ошибок, именуется двухсторонними, а те дискеты, у которых ошибок не имеет только одна сторона, — односторонними. Естественно, что односторонние дискеты дешевле двухсторонних. Поэтому программисты в целях экономии иногда используют их как двухсторонние — записывают информацию с обеих сторон. В таких случаях необходимо воспользоваться специальными программами обнаружения и изоляции дефектных участков диска, потому как при записи или считывании информации возможно появление непредсказуемых ошибок. Для персональных компьютеров «YAMAHA» такими программами являются VFY.COM и DSKVER.COM.

Работая с дискетами, операционная система оперирует элементарными единицами информации на носителе: *треками, кластерами, секторами и блоками*.

Данные на дискете размещаются по *дорожкам* (трекам).

Трек («track» — «дорожка») — это замкнутая окружность на поверхности дискеты, расположенная на определенном расстоянии от центрального отверстия.

Вся поверхность дискеты разбита на *секторы («sector»)*, причем геометрическое понятие «сектор» отличается от понятия, принятого в вычислительной технике. Сектор на диске — это участок дорожки. Трек содержит 16 или 18 секторов — в зависимости от типа контроллера. *Кластер («cluster» — «связка», «гроздь»)* состоит из двух секторов. Такая единица информации введена для удобства работы операционной системы с областью данных дискеты.

На двухсторонней дискете содержатся: 80 (&H50) *треков*; на каждом треке *по 9 кластеров*; кластер занимает 1 кБ памяти и состоит из двух *секторов* по 512 байт в каждом.

Сектор представляет собой основную единицу хранения информации на дискете. Вне зависимости от типа носителя, сектор имеет объем 512 байт.

При чтении или записи дисковод считывает (записывает) всегда целый сектор, независимо от объема запрашиваемой информации!

Сектор, в свою очередь, можно рассматривать как совокупность *четырех блоков*, каждый из которых занимает 128 байт памяти.

На двухсторонней дискете (720 кБ):
2CAh - 1 = 713 кластеров (с номерами 2(!),3,...,714), 59Eh = 1438 секторов

Взгляните на следующую табличку:

Тип дискеты	Обозначения типа
односторонние, одинарной плотности	SS — «Single Syded» SD — «Single Density»
односторонние, двойной плотности	SS — «Single Syded» DD — «Double Density»
двухсторонние, двойной плотности	DS — «Double Syded» DD — «Double Density»
двухсторонние, учетверенной плотности	DS — «Double Syded» HD — «High Density»

Односторонность и двухсторонность свидетельствует о том, имеет ли дисковод одну или две магнитные головки, которые обеспечивают запись и считывание информации с одной или двух сторон гибкого диска. Эта характеристика зависит и от качества магнитного слоя дискеты.

Важной характеристикой дискеты является *плотность записи*. Стандартные односторонние дискеты имеют плотность записи, называемую *двойной*, что позволяет записывать и считывать 40 треков данных на поверхности дискеты. Другой использующийся формат называется *учетверенной плотностью*, что на том же пространстве позволяет иметь 80 треков.

На дискете плотность записи обычно обозначается *числом треков на дюйм*: 135 TPI (TPI — «Tracks per Inch»), что соответствует примерно 110 дорожкам на поверхности дискеты (∅3.5").

Напомним, что дюйм условно обозначается двойным штрихом " и равен примерно 25.4 мм.

Приведем некоторые характеристики дискеты MF2-DD (двухсторонней, двойной плотности):

1. 1) емкость:
    - 1 МБайт у неформатированной дискеты;
    - 720 кБайт у дискеты, форматированной как двухсторонняя;
    - 360 кБайт у дискеты, форматированной как односторонняя;
2. 2) плотность при записи — 8717 бит/дюйм;
3. 3) плотность треков — 135 треков/дюйм;
4. 4) среднее время доступа — 95 ms;
5. 5) время перехода между треками — 6 ms;
6. 6) максимальное количество файлов — 112.

На практике используются четыре основных формата дискеты.

Различие между форматами определяется:

- числом используемых сторон диска (одно- или двухсторонний диск) и
- числом кластеров, приходящихся на трек (8 или 9 кластеров на трек).

Форматирование дискеты — это подготовка носителя к работе с данным типом драйверов т.е. магнитная разметка секторов, а также запись на диск (в нулевой сектор) программы загрузки операционной системы. Эта операция выполняется для того, чтобы при работе с диском магнитная головка смогла определится в пространстве и начать считывание нужного сектора.

Если Вы вставите новую неотформатированную дискету в дисковод и попытаетесь что–нибудь записать на нее или хотя бы примените команду «DIR», то получите сообщение о том, что в дисководе нет диска (Not ready error reading drive A). Генерация этого сообщения произошла при попытке головки драйвера сориентироваться, — после того, как она не нашла магнитную разметку секторов.

Параметры форматирования, зависящие от:

- типа драйвера: плотность записи и число кластеров в треке;
- технологии производства дискет: число сторон на диске и плотность записи информации (треков на дюйм).

> Для гауссовой строгости мы не имеем времени, господа.

Взгляните на приведенную ниже таблицу (учтите, что скорее всего Ваши дискеты относятся к типу 2DD с идентификатором носителя — F9).

Идентификатор носителя	**F8**	**F9**	**FA**	**FB**	**FC**	**FD**	**FE**	**FF**
Способ записи	MFM	MFM	FM	FM	Информация отсутствует !			
Тип носителя	1DD	2DD	1DD	2DD				
Байтов в секторе	512	512	512	512				
Треков на сторону	40	40	20	20	80	80	80	80
Сторон на дискете	1	2	1	2	1	2	1	2
Файлов на дискете	112	112	112	112	64	112	64	112
Количество FAT	2	2	2	2	2	2	2	2
Секторов в FAT	2	3	1	2	2	2	1	1
Кластеров в треке	9	9	8	8	9	9	8	8
Секторов в кластере	2	2	2	2	1	2	1	2

Дополненная версия таблицы здесь (добавлено 2023-05-18)

I.2. Схема хранения данных на дискете

> Золотое правило Мэрфи: у каждого обладателя золота — свои правила.
>
> *—Закон Мэрфи*

Опишем схему хранения данных, используемую в MSX-DOS:

1. для хранения данных используются стандартные 512-байтные сектора;
2. никакие сектора или области на дискете не резервируются;
3. распределение секторов данных и объединение секторов, образующихфайл данных, выполняется независимо от обслуживания самих секторов данных, с помощью механизма, получившего название

   > Таблица Размещения Файлов (ТРФ или FAT — «Files Allocation Table»)

4. каждая дискета имеет каталог, который служит для учета файлов, хранящихся на дискете.

Описанная схема хранения данных предполагает существование *четырех* различных типов секторов, один из которых используется для хранения данных, а три других имеют специальное назначение.

Любая дискета имеет *программу загрузки операционной системы*, записывающуюся в самом первом секторе дискеты (*нулевом секторе*), который называется *загрузочным сектором (англ. — «boot sector»)*.

Программа загрузки всегда записывается в загрузочный сектор любой форматируемой дискеты независимо от того, будет ли данная дискета когда–нибудь использоваться для запуска операционной системы. Сектор с программой загрузки и запуска системы представляет собой *первый* специальный тип сектора в MSX-DOS.

Второй специальный тип сектора используется для хранения Таблицы Размещения Файлов (FAT). FAT занимает два или три (в зависимости от параметров форматирования) сектора, непосредственно следующих за сектором с программой загрузки. Таблица Размещения Файлов служит для индикации занятости секторов данных на дискете.

Третий и последний специальный тип сектора используется для хранения *каталога* («Directory», директории, Справочник) дискеты. Каталог располагается вслед за Таблицей Размещения Файлов и занимает *семь* секторов на дискете.

Эти сектора специального назначения занимают *двенадцать первых секторов односторонней дискеты* или *четырнадцать — двухсторонней*. Все остальные сектора («Data Area») используются для хранения *данных*.

Каталог и Таблица Размещения Файлов располагаются в начале дискеты. На первый взгляд это представляется оптимальным размещением. Однако при доступе к файлу операционная система MSX-DOS сначала должна найти элемент этого файла в каталоге, а потом обратиться собственно к данным на носителе. В среднем, расстояние между каталогом и файлом на дискете составляет 20 треков, то есть, практически половину дискеты. Перемещение головок чтения–записи в дисководе является самой медленной операцией. Так что расстояние между каталогом и самим файлом может иметь важное значение с точки зрения временных характеристик работы.

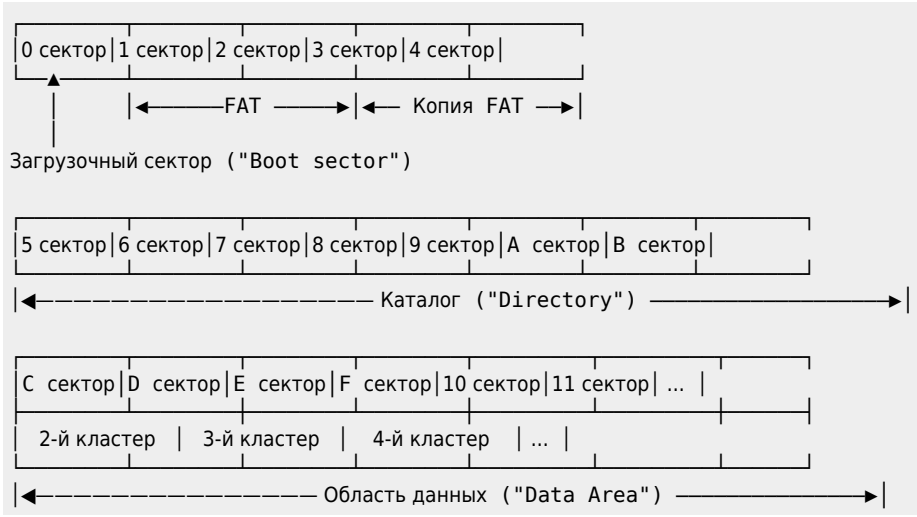
Если бы каталог располагался в середине дискеты, то среднее расстояние до секторов данных уменьшилось бы вдвое, т.е. до 10 треков. С другой стороны, работа с пространством данных, состоящим из двух частей (по обе стороны от каталога) будет значительно сложнее. Для операционных систем персональных компьютеров выгода от размещения каталога в середине дискеты слишком мала, чтобы браться за разрешение возникающих при этом дополнительных проблем.

Итак, область памяти на диске распределена следующим образом:

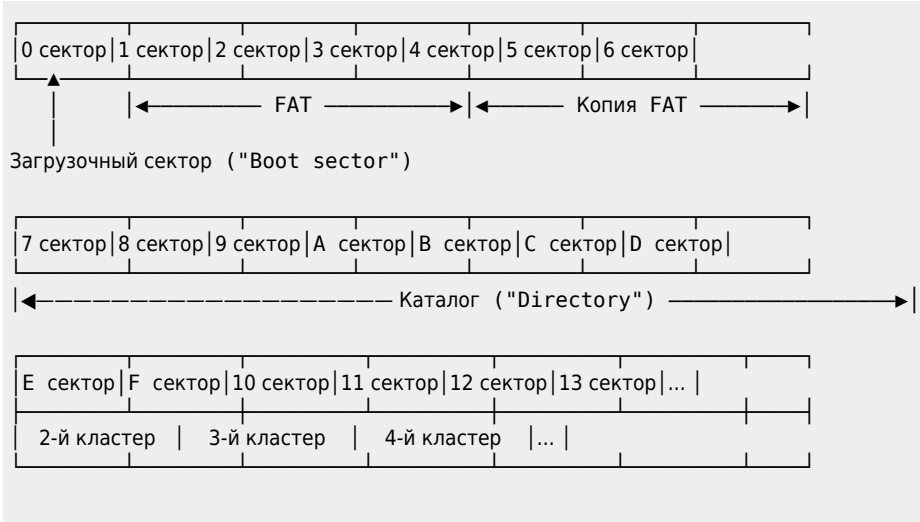
Загрузочный сектор («boot sector»)	Информация о дискете и программа, запускающая MSX-DOS
Таблица Размещения Файлов («FAT»)	Информация о занятости кластеров дискеты
Каталог («Directory»)	Информация о файлах
Область данных («Data Area»)	

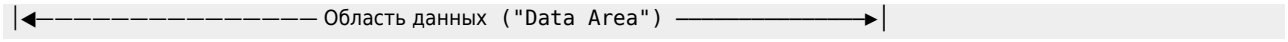
Приведем схему расположения и нумерации секторов для:

- a) *одностороннего диска*:



- b) *двухстороннего диска*:





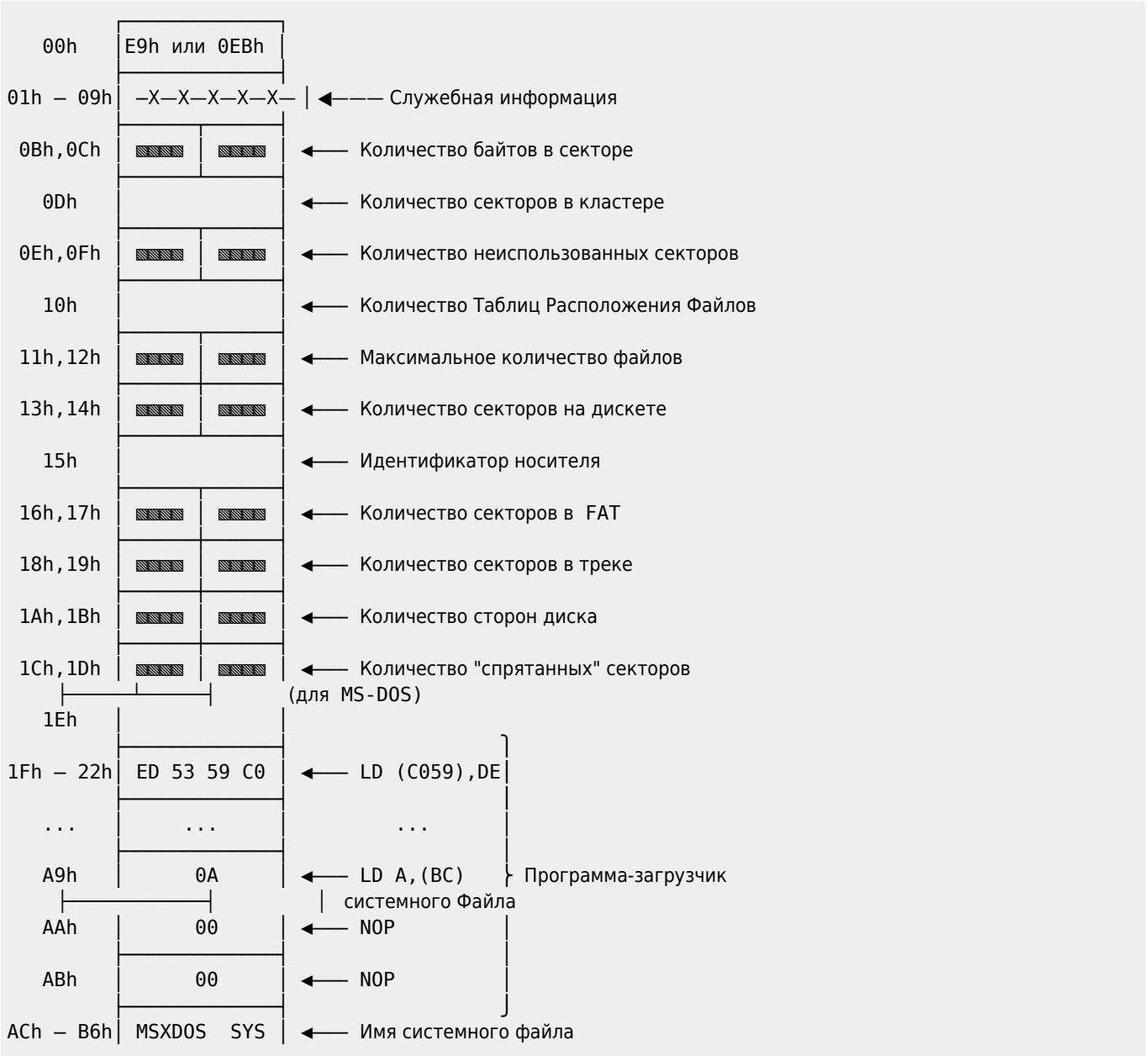
Обратите внимание, что первый кластер области данных имеет номер 002!

I.2.1. Нулевой сектор

> Я хорошо понимаю, что читателю не очень нужно все это знать, но мне-то очень нужно рассказать об этом.
>
> *—Жан Жак Руссо*

Часть содержимого нулевого сектора занимает такая информация о дискете, которая не изменяется в процессе работы с дискетой.

Содержимое нулевого сектора приведено на схеме:



```
    ├──────────────┤  (по умолчанию MSXDOS.SYS)
... │    Не    │
... │ используется│
    ├──────────────┤
  1FFh │      00      │
    └──────────────┘
```

> Что может быть честнее и благороднее,
> как научить других тому, что сам наилучшим образом знаешь.
>
> *—Квинтилиан*

Приведем пример использования дискового пространства в «личных» целях.

Программа-загрузчик MSX-DOS загружается в RAM с адреса C000h. Когда она «отработает» (система загрузится), в программе загрузки происходит переход по адресу 100h (здесь находится загруженная система). Если мы выполним в данном месте переход не по адресу 100h, а, например, по адресу C0D0h, где расположена первая половина нулевого сектора, Вы можете еще до запуска MSX-DOS «что–нибудь» сделать: отключиться от сети, запретить прерывания по таймеру, вывести какое-нибудь сообщение на экран, а потом выполнить переход по адресу 100h.

Вторая половина нулевого сектора находится где–то в RAM. Так как в первой половине этого сектора мало места, то Вашу программу придется разместить и во второй половине. Поэтому проще в первой половине дописать программу–загрузчик так, чтобы она загружала весь Boot sector по какому-либо известному адресу, а затем выполнить переход на адрес, по которому расположена вторая половина сектора.

В *boot–секторе* байты с номерами 53h, 54h и 55h нужно изменить: C3h, 00h, 01h (JP 100h) на C3h, D0h, C0h (JP C0D0h), а с адреса D0h разместить программу:

```
┌──────┬──────────────────────────────────────────────────────┬──────────┐
│Адрес│    Программа на Ассемблере с комментариями    │  Коды  │
├──────┼──────────────────────────────────────────────────────┼──────────┤
│0D0h:│ LD   DE,D000h ;    Установка нового                  │ 11 00 D0 │
│     │ LD   C,1Ah    ; адреса передачи                      │ 0E 1A    │
│     │ CALL F37Dh    ; boot–сектора.                        │ CD 7D F3 │
│     │ LD   HL,100h  ;    Загрузка одного сектора      │21 00 01│
│     │ LD   DE,0     ; (с номером 0) с диска,        │11 00 00│
│     │ LD   C,2Fh    ; находящегося в дисководе "A".       │ 0E 2F    │
│     │ CALL F37Dh    ;                                     │ CD 7D F3 │
│     │ JP   D100h    ; Переход на адрес Вашей программы.   │ C3 00 D1 │
│     │   ...                                               │          │
│100h:│               ; Здесь располагается Ваша программа.│          │
│     │   ...                                               │          │
│1FDh:│ JP   100h     ; Переход на адрес прогр. MSXDOS.SYS.│ C3 00 01 │
└──────┴──────────────────────────────────────────────────────┴──────────┘
```

Обратите внимание: *функции* BDOS вызываются командой `CALL F37Dh`, т.к. *программа загрузки* работает в рабочей области слота «BASIC» *(слот 0)*.

Примечание. Как только Вы начинаете работать с дискетой в системную область слота 3-2 помещается информация о дискете, которая *может изменяться* в процессе работы с дискетой.

Эта информация называется DPB
(«Drive Parameter Block«-»Блок Параметров Носителя»)

Кратко опишем:

- α) *содержимое* DPB (21 байт):

- β) *размещение* DPB:

Адреса в рабочей области слота 3-2	Назначение (или содержимое)
F195 ÷ F1A9	DPB дисковода "A"
F1AA ÷ F1BE	DPB дисковода "B"
F1BF	0
F1C0	0
F1C1	Текущий дисковод (0 соотв. "A")
F1C2	Текущий трек (дисковод A)
F1C3	Текущий трек (дисковод B)
F1C4	Протокол текущего дисковода
F1C5	Трек (разметка)
F1C6	Тип носителя (форматирование)
F1C7	Число дисководов в системе
F1C8	0
F161 ÷ F175	DPB дисковода "C"
F176 ÷ F18A	DPB дисковода "D"
F18B ÷ F194	Аналогично байтам F1BF ÷ F1C8
E595 ÷ EB94	Копия Таблицы размещения файлов
EB95 ÷ ED94	Копия сектора каталога, содержащего последний упомянутый Вами файл

Как спасти только что уничтоженный текстовый файл

От упырей и призраков, от длинноногих бестий и \
тварей, налетающих по ночам, Господи, избави нас.

—Древняя корнуоллская молитва

Слухи о моей смерти сильно преувеличены.

—Марк Твен

Прежде, чем начать восстановление, уберите дискету(ы) из дисковода, попробуйте спокойно вдохнуть и выдохнуть несколько раз, отойдите на пятнадцать минут (опытные данные) от компьютера. Выпейте чашечку чая, а лучше крепкого кофе. Далее, возьмите в руки настоящее руководство и ...

Для спасения Вашей информации есть *два* способа.

- *Первый:* «листать» область данных диска, разыскивая кластеры с данными, относящимися к «убитому» файлу и копировать эти кластеры на другой диск. Этот способ отнимает довольно много времени, вероятно будет нарушена структура файла (изменен порядок расположения кластеров, относящихся к данному), но скорее всего Вы спасете *весь* файл, без потерь.
- *Второй:* по начальному кластеру файла в режиме FAT Fixer определить номера кластеров, имеющих вместо ссылки на некоторую ячейку в FAT код 000 и просто восстановить *ссылки* на последующие кластеры файла (при ситуации, когда на протяжении большого промежутка времени Вы не удаляли ни одного файла, т.е. есть гарантия, что в FAT не было пробелов в заполненной части области данных). Используя этот способ, Вы сэкономите время, но при невнимательной работе потеряете часть файла.

Вначале изложим подробно *первый* из них, простой, но очень трудоемкий, который мы разобьем на этапы.

- Первый этап.
    1. Загрузить программу Disk Fixer.
    2. Вставить в дисковод «A» дискету с восстанавливаемым файлом.
       Вставить в дисковод «B» дискету, на которую Вы будете записывать нужную Вам информацию (перед этим откройте диск на запись).
    3. В режиме DISK FIXER необходимо изменить объем буфера с 512 байт (1 сектор) на 1 Кбайт (1 кластер) командой `[S]cale` (режим `[C]luster`).
    4. Перейти на дисковод «B» (команда: `dri[V]e ,[B]`). В системе FAT FIXER загрузить ТРФ (команды: `[F]AT`, `[R]ead`) и определить первый *свободный* кластер в конце Таблицы («прыгнуть» на ячейку с номером 2C9 и, двигаясь *вверх* и *влево* установить курсор на последнюю позицию с содержимым 000).
       Записать на бумаге номер этого кластера.
    5. Вновь вернуться на дисковод «A» (команда: `dri[V]e ,«A»`).
    6. Загрузить кластер с номером 002 (команда: `[R]ead, 002`).
- Второй этап.

  Просмотреть каждый кластер, определяя, относится ли он к утерянному файлу или нет.

    - Если *нет*, то просмотреть следующий кластер (клавиша F2).
    - Если *да*, то записать кластер на другой диск и прочитать новый командами:

      ```
      dri[V]e , [B] ;
      [W]rite , <номер пустого кластера на диске в дисководе "B">;
      dri[V]e , [A] ;
      [R]ead , <номер следующего кластера на диске в дисководе "A">.
      ```

  Это очень неприятно — запоминать (или записывать) номера кластеров, с которыми работаешь, но если уж убиваешь файл, думай о последствиях!

- Третий этап.

  После того, как Вам покажется, что файл уже полностью восстановлен, нужно в директории диска «B» создать новый файл. Для этого войти в систему FAT FIXER (команды: `[F]AT` , `[R]ead`) и вспомнить номер последнего кластера, в который был записан кусок восстановленной информации (никаких клавиш нажимать не надо!). Затем «прыгнуть» на ячейку с этим номером (команда: `[L], <номер>`) и установить признак конца файла (команда: `[TAB]`).

Заметим, что если кластер является последним в файле, то в позиции курсора будет находиться знак «$$$».

- Четвертый этап.

  Итак, Вы имеете перед собой цепочку кодов «000», заключенную между двумя кодами «$$$». Заполняйте это пространство ссылками на следующие кластеры. Данная процедура выполняется так:

  1. устанавливаете курсор на первый код «000» в цепочке и смотрите на раздел «ENTRY» в верхней части экрана. Там будет указан номер ячейки, в которой находится курсор;
  2. прибавьте к полученному числу (в шестнадцатеричной системе) единицу и запишите результат в ячейку (команда: `[CR] + <результат>`). Это действие повторяйте, пока не иссякнут нули в цепочке кодов.
- Пятый этап (последний аккорд!).

  Войти в режим редактирования в системе DIRECTORY FIXER (команды: `dri[V]e, «B»; [D]irectory; [R]ead; [CR]`), на чистом месте в графе имен файлов вписать имя Вашего файла, а в следующей графе — номер первого свободного кластера,о котором говорилось при изложении четвертого пункта первого этапа. О дате и времени последнего изменения файла не стоит беспокоиться, но объем файла необходимо указать обязательно!

  Объем вычисляется по следующей формуле:

  ```
  1024•(Nlast + 1 - Nfirst)
  ```
  ,

  где

  - `Nlast` — номер последнего записанного на диск кластера,
  - `Nfirst` — номер первого свободного кластера на диске в дисководе «B» (см. четвертый пункт первого этапа).

  Полученное натуральное число надо перевести в шестнадцатеричную систему счисления.

- Шестой этап.

  Вписав все данные о файле, «сбросить» их на диск (команды: `[ESC]; [W]rite; [ESC]; [F]AT; [W]rite`).

  Выйти из программы Disk Fixer (команда: `[E]xit , [Y]es , [Y]es`).

  Дошедшему до конца — *браво!* Остается лишь проверить, удались ли Ваши манипуляции, или все надо начинать с начала. Загрузите полученный текст в текстовый редактор и внимательно просмотрите его...

Примечание. Можно, конечно, писать файл и на тот же диск, но при этом возникает опасность погубить окончание файла, если ячейки FAT, относящиеся к «невинно убиенному» файлу были последними в FAT.

Переходим к описанию *второго*, более интеллектуального способа. Аналогично предыдущему изложению разобьем последовательность действий на ряд этапов.

- Первый этап.
  1. Загрузить программу Disk Fixer.
  2. Вставить диск с восстанавливаемым файлом в дисковод «A».
  3. В системе DIRECTORY FIXER найти имя уничтоженного Вами файла (оно будет содержать вместо первого символа знак «*»). Это можно осуществить командами:
     `[D]irectory, [R]ead`; поиск — клавишами управления курсором.
  4. В графе, следующей за той, где указано имя файла, посмотрите точку входа в ТРФ (номер первого кластера файла). В последней графе найдите объем файла. Теперь рассчитайте число кластеров в файле по формуле:

     ```
     Число_кластеров = 1 + Объем_файла div #400
     ```

     и переведите полученный результат в десятичную систему. Номер первого кластера и число кластеров записать или запомнить.

  5. Восстановить имя файла, вписав на место знака «*» первый символ имени (или любой другой, кроме символа «E» русского алфавита) при помощи команд: `[CR], <нужный символ>, [ESC]; [W]rite`.
- Второй этап.
  1. Загрузить систему FAT FIXER (команды: `[ESC]; [F]AT, [R]ead`).
  2. «Прыгнуть» в ячейку с номером первого кластера «погибшего» файла командой:
     `[L], <Номер ячейки>`.
  3. Передвигая курсор по экрану, посмотреть номер следующей «пустой» ячейки (с кодом «000»). Вписать ее номер в предыдущую «пустую» ячейку командами:
     `[CR], <номер кластера>, [ESC].`

  Этими манипуляциями Вы восстановите одну ссылку, принадлежащую утерянному файлу. Продолжайте в том же духе,

попутно считая каждую восстановленную ссылку. Когда их число достигнет величины Число_кластеров_в_файле_без_единицы, в следующей ячейке (содержащей нулевой код) поставьте признак конца файла командой:
`[CR], [TAB], [ESC].`

- Третий этап.

  Записать восстановленную ТРФ на диск при помощи команды
  `[W]rite, [Y]es`, и с радостью выйти из программы Disk Fixer командами:
  `[ESC]; [E]xit, [Y]es, [Y]es`.

  Вздохните свободно: если никаких сбоев не было, то ваш файл спасен! Загрузите его в текстовый редактор и внимательно просмотрите. Запишите текст на диск снова (желательно под другим именем).

Примечания:

- α) проблема восстановления файлов значительно усложняется, если было уничтожено несколько файлов; и еще больше, если среди нескольких уничтоженных файлов надо восстановить один самый нужный. В этом случае придется восстанавливать все файлы в один большой, затем указывать общий объем файла, а потом в текстовом редакторе отбрасывать «все лишнее»;
- β) следует помнить, что в тексте может встретиться код ASCII символа, обозначающего конец текстового файла (1Ah). Поэтому после восстановления Таблицы Размещения Файлов в режиме DISK FIXER надо просмотреть содержимое всех восстановленных кластеров и заменить все коды 1Ah на код 20h (соответствует символу «Пробел»).

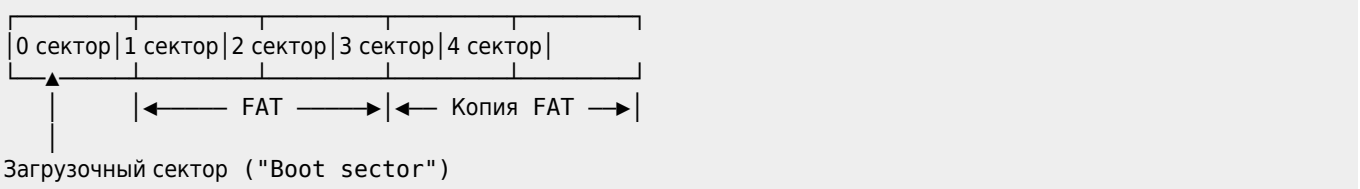
I.2.2. Таблица Размещения Файлов

> — Как Вы считаете, следует писать — сэндвич или сандвич?
> — Да зачем нам вообще этот иноземный термин, если существует прекрасное русское слово — «бутерброд»?
>
> *—Народный фольклор*

Для двухстороннего диска в секторах с номерами 1÷3 и 4÷6 содержится Таблица Размещения Файлов (FAT — «File Allocation Table»), в которой располагается информация о размещении файлов на дискете.

Сектора 1÷3 и сектора 4÷6 — это две копии одной и той же Таблицы Размещения Файлов.

Если же диск отформатирован как *односторонний* («Single»), то FAT занимает уже не 3, а 2 сектора, и размещается в секторах 1÷2, а копия — в секторах 3÷4.



Предполагается, что копии Таблицы Размещения Файлов должны быть идентичными. Хранение двух экземпляров Таблицы Размещения Файлов представляет собой простую предосторожность, связанную с большой важностью информации, содержащейся в таблице. Восстановление поврежденной FAT представляет собой непростую задачу.

Для Таблицы Размещения Файлов выделено 4 (или 6) секторов не столько для того, чтобы хранить две копии,

сколько для обеспечения возможности увеличения этой Таблицы в будущем. *Основной принцип* организации Таблицы Размещения Файлов заключается в создании таблицы, каждый элемент которой соответствует одному кластеру.

Элементы таблицы содержат признаки занятости кластера.
Доступные (свободные) элементы таблицы содержат *нулевые* значения.
Участки пространства на дискете, принадлежащие одному файлу, связаны в *цепочку*. Опишем организацию этой цепочки.

Во-первых, определенный элемент *Справочника* файла (см. ниже раздел I.2.3) содержит номер элемента в Таблице Размещения Файла, который соответствует первому из кластеров, выделенных для файла.

Во-вторых, каждый элемент Таблицы Размещения Файлов содержит номер *следующего* кластера файла и так далее, пока не будет достигнут последний кластер файла.

В-третьих, в последнем элементе Таблицы Размещения Файлов для данного файла находится *признак конца файла*.

> Некий поэт спросил, нравится ли Шамфору
> написанное им двустишье. «Очень,— заявил
> Шамфор,— только нельзя ли его сократить?»
>
> *—С.Шамфор. Характеры и анекдоты*

Нумерация элементов Таблицы Размещения Файлов начинается с 2. Это означает, что первые два элемента Таблицы Размещения Файлов (с номерами 0 и 1) не используются для хранения информации о размещении данных. Они зарезервированы для хранения очень важной информации — *сведений о формате дискеты*.

Код формата хранится в первом байте Таблицы Размещения Файлов

Теперь можно подробно рассмотреть *кодировку* информации в Таблице Размещения Файлов. Вам, конечно уже известно, что на двухсторонней дискете для Области данных отведено 713 кластеров.

Элемент Таблицы Размещения Файлов должен позволять хранить номер другого элемента Таблицы (*номер кластера*, а максимальный номер, который может храниться в одном байте равен всего 0FFh=255, следовательно, длина элемента Таблицы должна быть *больше* одного байта.

С другой стороны, если бы длина элемента Таблицы Размещения Файлов была равна *двум* байтам (максимальный номер, который может храниться в двух байтах равен 0FFFFh = 65535), то с учетом того, что FAT занимает 3 сектора на дискете (3×512 = 1536 байт), мы получим, что можно закодировать 1536/2=768 кластеров, что вполне достаточно для этого типа дискет, но на поиск нужной ячейки будет уходить в два раза больше времени, чем при однобайтном кодировании. Истина, как всегда, находится посредине!

Для решения этой проблемы была разработана довольно сложная схема представления элементов FAT в виде *трех* шестнадцатеричных цифр, занимающих *полтора байта* (12 двоичных разрядов) на дискете. Всего же таким способом можно закодировать 1024 кластера.

Информация о формате дискеты

Схема хранения чисел в виде полуторабайтных кодов выглядит довольно странно для программиста, хотя использование этого факта на машинном языке реализуется очень просто.

Последовательные элементы Таблицы Размещения Файлов разбиваются на пары, объединяющие два полуторабайтовых значения в последовательность из трех байтов для каждой пары.

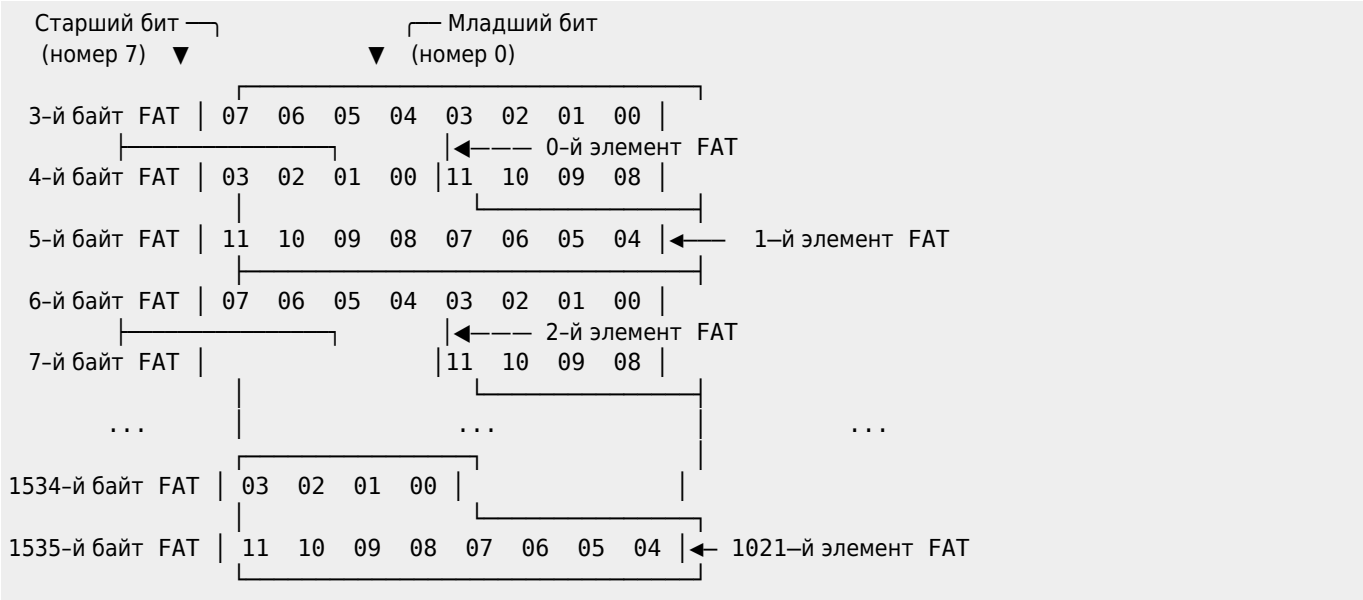
Для получения значения, хранящегося в элементе Таблицы Размещения Файлов с номером X, нужно выполнить следующие действия:

1. 1) умножить X на 1.5 (для этого выполняется умножение на 3 с последующим делением на 2), прибавить 3 и найти целую часть полученного значения:

   ```
   M := [X×1.5+3]
   ```

2. 2) байты с адресами M и M+1 можно теперь загрузить в шестнадцатибитный регистр микропроцессора (кто не знает ассемблера микропроцессора Z80 — не бойтесь: это не больно!). Теперь в регистре находится четыре шестнадцатеричных цифры, а необходимы только *три* из них.

Если номер элемента Таблицы Размещения Файлов нечетный, то нужно отбросить *последнюю* цифру, а если он четный — то *первую*.



Элементы FAT с содержимым 2CBh÷FFFh обозначают особые типы *зарезервированных* кластеров, например, EEEh в MSX-DOS используется для выделения кластеров с дефектными секторами. Дело в том, что на поверхности дискеты могут встречаться участки с дефектами магнитного покрытия, которые нельзя эффективно использовать для хранения данных.

В организации Таблицы Размещения Файлов могут возникать определенные дефекты. Два наиболее заметных дефекта — это *«беспризорные» кластеры* и *перекрещивающиеся файлы*.

Если элемент Таблицы содержит значение, указывающее, что элемент используется (значение не равно нулю) и, то же время, этот элемент не входит ни в одну из цепочек определения размещения файлов, то такой элемент Таблицы Размещения Файлов становится как бы *«беспризорным»*. «Беспризорные» кластеры чаще всего возникают, когда программы начинают создавать файл (так что для него выделяются кластеры), но не закрывают его, вследствие чего не завершается создание элемента Справочника для данного файла.

Во втором случае, может случиться так, что две (или более) различные цепочки, определяющие размещение файлов, приводят к одному и тому же кластеру. Такие файлы называются *перекрещивающимися*.

Если «беспризорные» кластеры встречаются достаточно часто, то перекрещивающиеся файлы возникают очень редко. Если такая ситуация все же Вам встретится, нужно скопировать каждый из файлов на другую дискету для последующего восстановления (если оно потребуется). После этого копии всех файлов будут содержать информацию из общих секторов, хотя она может принадлежать только одному из них.

При работе MSX-DOS в памяти хранится копия Таблицы Размещения Файлов для каждого используемого дисковода (см. раздел I.2.1). Когда в Таблице производится какое-либо изменение, оно записывается в обе копии на дискете. При новом обращении к дискете MSX-DOS считывает Таблицу Размещения Файлов, чтобы установить формат дискеты.

Чем незначительнее изменение проекта,

тем разрушительнее его последствия.

—Закон Мэрфи

I.2.2. Как уберечь диск от случайных уничтожения файлов

Наиболее полезны те советы, которым легко следовать.

—Вовенарг

Вся информация о содержимом дискеты хранится в двух специальных областях: в Таблице Размещения Файлов и в Справочнике диска. Если в конце напряженного рабочего дня позаботиться о сохранении такой ценной информации, то на следующий день Вы не слишком разочаруетесь, когда вдруг произойдет сбой в системе или «упадет» напряжение в сети в момент, когда Вы только начали записывать измененный текст. Пропадет труд только одного дня. Остальное можно будет восстановить, затратив минимум времени.

Идея состоит в том, чтобы сектора с номерами 004 ÷ 00D (копии FAT и Справочник) копировать в зарезервированную каким-либо образом область (объемом 5 Кбайт).

Трудность заключается в следующем: необходимо «скрыть» этот объем от операционной системы. Такой «фокус» можно осуществить двумя способами:

- 1) создать на дискете с помощью текстового редактора файл в 5120 байт, «скрыть» его от системы (установить в байте атрибутов файла код 02, запрещающий MSX-DOS работать с данным файлом).

  «Скрытый» файл невозможно уничтожить средствами операционной системы!

- 2) проставить коды, обозначающие «плохие» кластеры в любом свободном месте Таблицы Размещения Файлов (лучше - в самом ее конце); операционная система не будет пытаться ни записать, ни прочитать эти кластеры (информация же в кластерах сохраняется!).

«Фокус» номер *два* мы считаем наиболее удачным, так как в нем не используется Справочник, т.е. вероятность возникновения ошибок при работе с диском практически равна нулю (чем проще конструкция, тем меньше сбоев она дает).

I.2.3. Справочник дискеты

... старейшим из дошедших до нас каталогов признается список литературных произведений на шумерской глиняной плитке, относящийся к 2ООО г. до н.э.

—Михайлов А.Н.,Черный А.Ч.,Гиляревский Р.С.

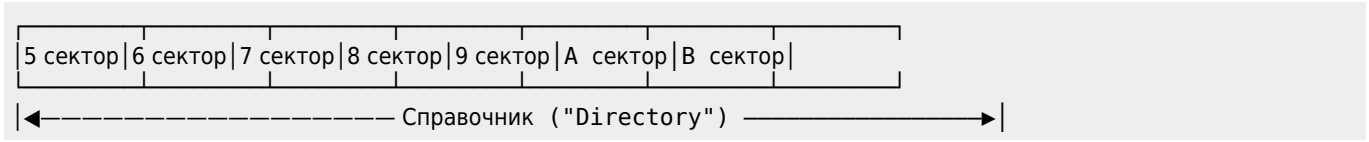
Справочник (каталог, директория, оглавление) дискеты содержит список всех файлов, находящихся на дискете. Элементы справочника содержат всю необходимую информацию о файле, за исключением информации о размещении файла (которая хранится в Таблице Размещения Файлов).

Справочник располагается

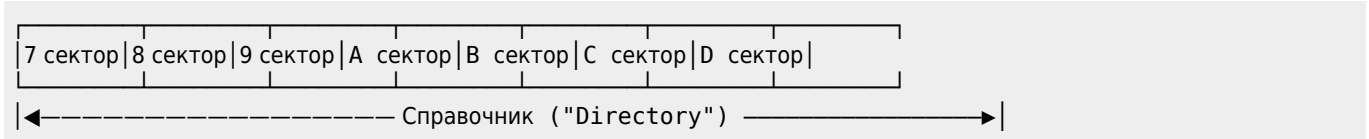
- α) в секторах 07h ÷ 0Dh, если диск *двухсторонний*,

- β) в секторах 05h ÷ 0Bh, если диск *односторонний*.

или более подробно для *односторонней* дискеты:

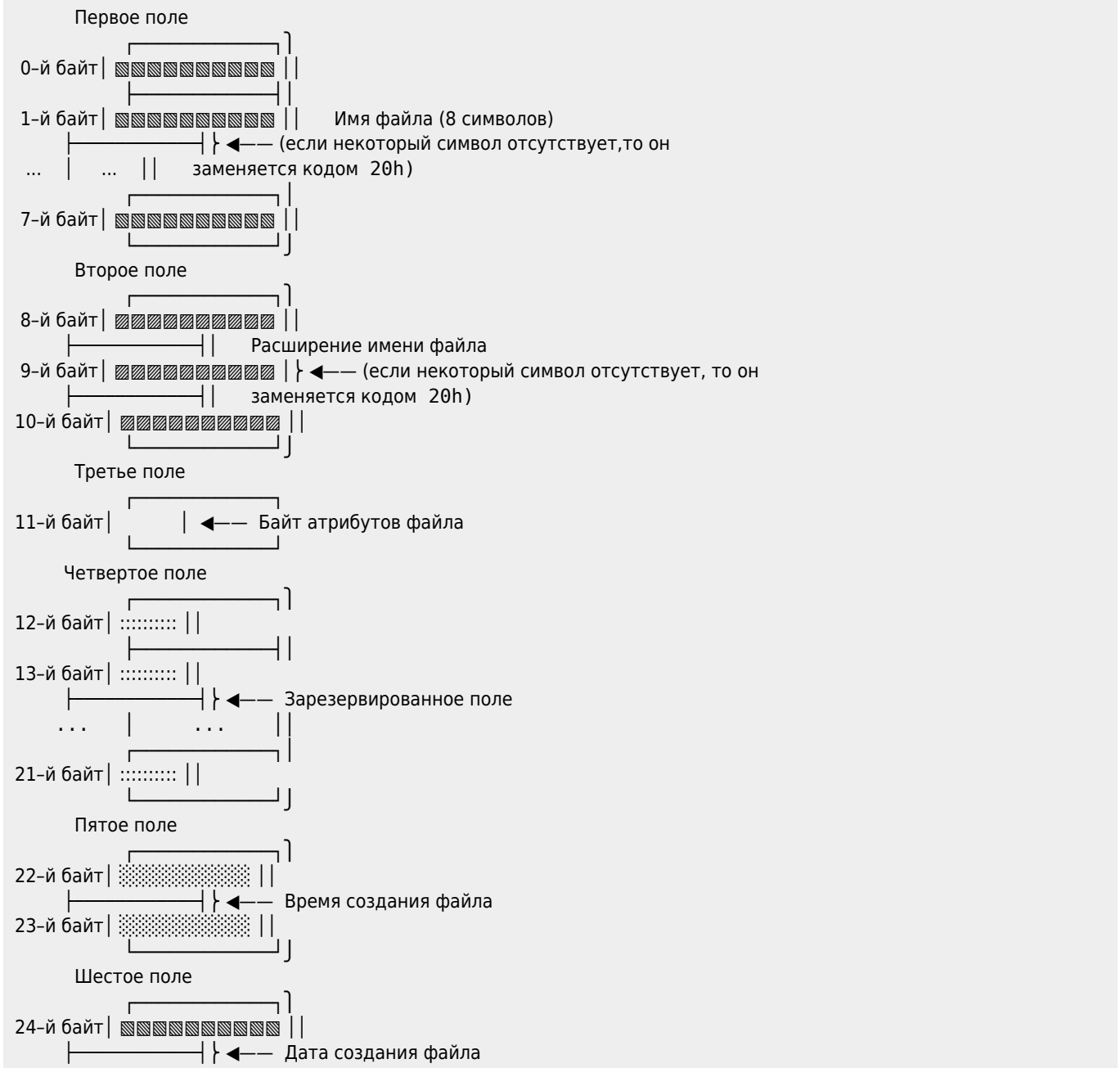


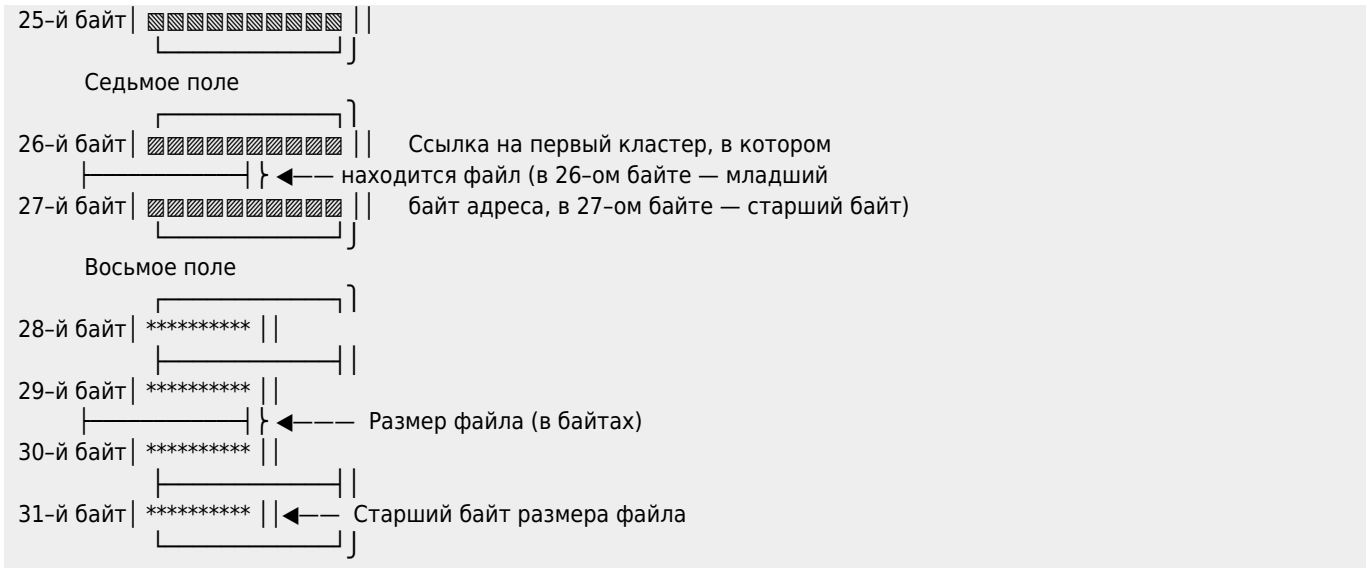
и для *двухсторонней* дискеты:



Каждый элемент Справочника имеет длину 32 байта. Следовательно, в 512-байтном секторе помещается ровно 16 элементов Справочника. На дискете выделено 7 секторов для Справочника, что позволяет хранить 112 элементов (7×512/32 = 112).

Каждый элемент Справочника состоит из восьми полей следующего назначения:





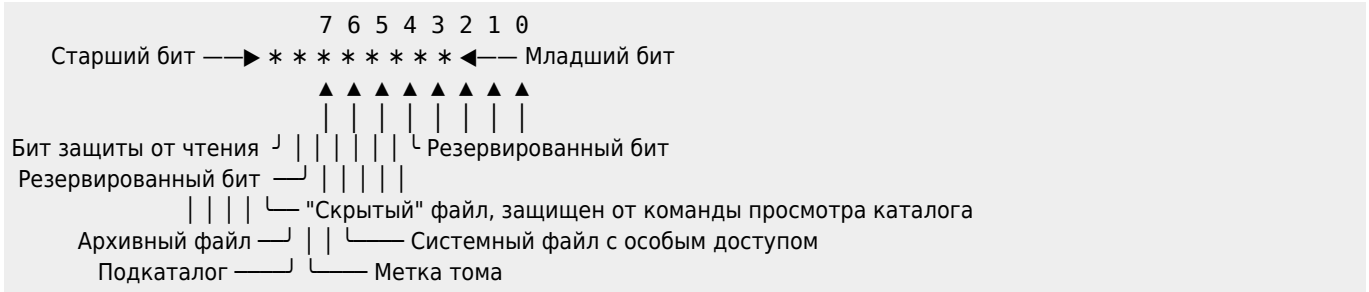
> Помните, что для человека звук его имени самый сладкий и самый важный звук в человеческой речи.
>
> *—Д.Карнеги*

Имя файла. Это поле имеет длину 8 байт и содержит имя файла. Если длина имени меньше восьми символов, то недостающие символы заменяются символом «Пробел» (код 20h). Если *первый байт имени файла* содержит число E5h, то это означает, что файл, которому соответствует этот элемент Справочника, уже уничтожен, т.е. после уничтожения файла первый символ имени файла заменяется шестнадцатеричным кодом E5. *Вся информация в элементе каталога, за исключением первого символа имени, сохраняется*.

Расширение. Это поле имеет длину три байта. Оно содержит расширение имени файла. Как и в случае с самим именем, короткое расширение дополняется символами «Пробел». Если файл не имеет расширения, то это поле содержит три символа «Пробел».

Атрибут. Это поле состоит из *одного* байта. Поле атрибута в основном используется для установления признака «скрытого» файла, т.е. такого файла, имя которого не обнаруживается обычными программами работы с оСправочниками.

В операционной системе MSX-DOS биты байта атрибутов несут следующую нагрузку:



Бит 0 в системе MSX-DOS зарезервирован.

Биты 1-й и 2-й служат для индикации атрибутов «скрытого» и «системного» файла. *Первый* бит байта атрибутов, установленный в 1, определяет «скрытый» файл, а *второй* бит, установленный в 1, — системный файл. Таким образом, «видимый» файл будет иметь нулевой байт атрибута (00000000), для «скрытого» файла байт атрибута содержит значение 2 (00000010), для системного файла байт атрибута содержит значение 4 (00000100), а для «скрытого» системного файла байт атрибута содержит значение 6 (00000110).

Хотя обработка системного атрибута осуществляется независимо от скрытого, оба этих атрибута практически совпадают по своему функциональному назначению. При использовании любого из них файл становится «невидимым».

Если *бит 1* или *бит 2* равен 1, то файл исчезает из каталога. Его нельзя ни стереть, ни прочитать, т.к. при всяком обращении к нему операционная система сообщает: «File not found».

Бит 3 (содержимое байта атрибутов: 00001000) указывает, что элемент Справочника содержит метку тома. Сама метка хранится в полях имени файла и расширения, которые воспринимаются в этом случае как одно целое.

Если бит 3 равен 1, то файл исчезает из каталога. Его нельзя ни стереть, ни прочитать, т.к. при всяком обращении к нему говорится: «File not found»

В операционной системе MSX-DOS *бит 4* (содержимое байта атрибутов: 00010000) используется для указания элементов Справочника, соответствующих Справочникам нижнего уровня. Поскольку Справочники нижнего уровня хранятся на диске подобно обычным файлам данных, им необходим собственный элемент в корневом Справочнике. В этом элементе используются все поля, кроме размера файла, в данном случае равного нулю. Действительный размер файла Справочника нижнего уровня легко определяется из соответствующей последовательности в Таблице Размещения Файлов.

В операционной системе MSX-DOS равенство бита 4 единице приводит к тому, что при просмотре каталога командой `DIR`, Вы увидите на экране слово `<DIR>` в том месте каталога, где должна находиться информация об объеме файла. Это означает, что этот файл — теперь не файл вовсе, а имя некоторого *подкаталога*.

Бит 5 (содержимое байта атрибутов: 00100000) предусмотрен для облегчения создания резервных копий файлов на *жестких* дисках. Для файлов на *гибких* дисках этот атрибут *бесполезен*.

Бит 6 зарезервирован в системе MSX-DOS.

Если *бит 7* равен 1, то файл становится нечитаемым (например, Вы не сможете загрузить его в текстовый редактор и т.д.).

Зарезервированное поле. Это поле зарезервировано для возможного использования в будущем. Любые *новые* операции над Справочником файлов могут использовать это поле. Эти байты имеют значение 00. Любые другие значения этого поля указывают на какой-либо вариант его использования.

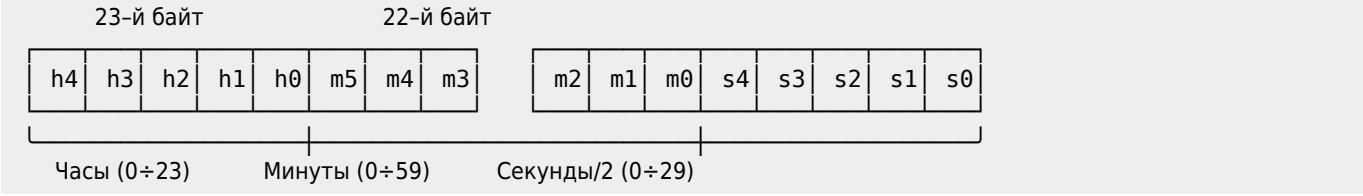
Счастливые часов не наблюдают.

—А.С.Грибоедов. Горе от ума

Время. Это поле имеет длину два байта в формате шестнадцатиразрядного целого числа без знака. В этом поле хранится время создания или модификации файла.

Большинство операций, использующих это поле, таких как операция распечатки содержимого Справочника (DIR), выдают время с точностью до минут, хотя число, хранящееся в поле времени, позволяет определить время с точностью до двух секунд. Код времени, хранящийся как шестнадцатиразрядное целое число без знака, вычисляется по следующей формуле:

```
Время = Часы×2048+Минуты×32+Секунды/2
```



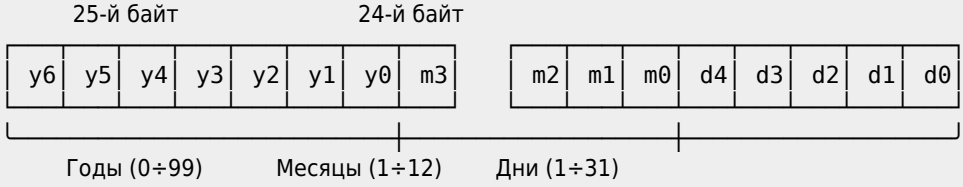
Все врут календари.

—А.С.Грибоедов. Горе от ума

Дата. Это поле состоит из двух байтов. Как и время, даты хранятся в виде целого шестнадцатиразрядного числа без знака, которое вычисляется по формуле:

```
Дата = (Год-1980)×512+Месяц×32+День
```

Диапазон значений лет составляет от 1980 до 2079, причем хранятся они в виде относительных величин от 0 до 99. Хотя формат позволяет задавать относительный номер года 127 (что соответствует 2107 году), операционная система MSX-DOS позволяет работать с годами только до 2079. Никто, правда, не ожидает, что MSX-DOS будет использоваться так долго!



Как формат, так и размещение полей даты и времени подобраны таким образом, чтобы вместе они образовывали единое четырехбайтовое поле, которое можно использовать в операциях сравнения. Достаточно просто извлекать компоненты даты и времени из соответствующих полей и вычислять их разность.

Например, для разделения даты на составные части можно использовать следующие формулы, записанные на языке программирования Pascal:

```
Год:=1980+поле_даты div 512
Месяц:=(поле_даты mod 512) div 32
День := поле_даты mod 32
```

Номер начального кластера. Это двухбайтовое поле содержит шестнадцатиразрядное число, являющееся смещением до начальной точки файла в Таблице Размещения Файлов.

Размер файла. Это поле состоит из четырех байтов. Размер файла задается в байтах и хранится в формате четырехбайтового целого числа без знака. Размер файла не всегда указывает точное число байтов. Для всех файлов это поле должно соответствовать размеру файла в секторах.

Для программных файлов, представленных в виде «COM»–файла и для файлов, созданных из данных фиксированной длины, только поле размера файла позволяет точно определить, где находится конец данных. Для этих файлов значение в поле размера файла хранится с точностью до байта.

Для файлов некоторых других форматов такая точность не обязательна и размер файла, указанный в соответствующем поле, может отличаться от действительного. Наиболее часто это случается с текстовыми файлами. Текстовые файлы в кодах ASCII имеют маркер (признак) конца файла, хранящийся в самом файле, который фиксирует точный конец данных.

Примеры, взятые из жизни

> Опыт увеличивает нашу мудрость,
> но не уменьшает нашей глупости.
>
> *—Джош Биллингс*

Пример номер один

Приведем один простой *пример* из жизни.

Опишем ситуацию: в дисководе «A» находится закрытый на запись диск. В дисководе «B» - диск с разрешенной записью. В командной строке была допущена ошибка вида:

```
A>copy con batch.bat
```

(перед именем `batch.bat` не было указано имя дисковода `B`).

С консоли создали некоторый текстовый файл и нажали клавиши Ctrl+Z. Естественно, что операционная система

попыталась записать эту информацию на дискету, находящуюся в дисководе A и в результате сообщила:

```
Write protect error writing drive A:
Abort, Retry, Ignore?
```

.

Оператор поменял диски местами (что категорически запрещается делать — сейчас поймете, почему) и нажал клавишу R. В результате этого «запрещенного» приема все текстовые файлы потеряли: кто — начало, кто окончание. Возникла паника, так как диск содержал уникальную информацию.

А случилось вот что: операционная система считала ТРФ в оперативную память и произвела необходимые изменения. Затем начала поиск того сектора каталога, в который можно было бы вписать новое имя создаваемого файла. Номер этого сектора оказался равным 9. Он был загружен в оперативную память компьютера и MSX-DOS вписала все необходимые данные в копию сектора. И только после этого была сделана попытка записать файл на диск. Диск оказался защищенным от записи. Программа выдала сообщение об ошибке. После перестановки дисков со стороны операционной системы не было предпринято никаких попыток заново загрузить FAT и Справочник, она сразу (о ужас!) записала все данные на дискету. FAT перестала соответствовать реальному расположению файлов на диске.

Но все было легко восстановить по двум причинам:

- *во-первых*, до «гибели» файлы на дискете были расположены в идеальном порядке, т.е. не было ни одного «скачка» в ссылках;
- *во-вторых*, заполненная часть Справочника диска располагалась в двух секторах (7 и 8), а измененный сектор имел номер 9, а это означает, что вся информация о началах файлов сохранилась.

Итак,началось восстановление «погибшей» информации. Вначале мы очистили ТРФ и записали строгую цепочку последовательных ссылок от начала FAT и до конца последнего файла. Конец последнего файла нашли по следующей формуле:

```
Номер_первого_кластера_последнего_файла + (Его_объем div 400h) + 1 ,
```

(все вычисления производятся в шестнадцатеричной системе).

Распечатали оглавление диска (этого можно не делать) и установили признаки концов файлов в соответствии с каталогом по формуле:

```
Последний_кластер_файла = Начальный_кластер - 1
```

.

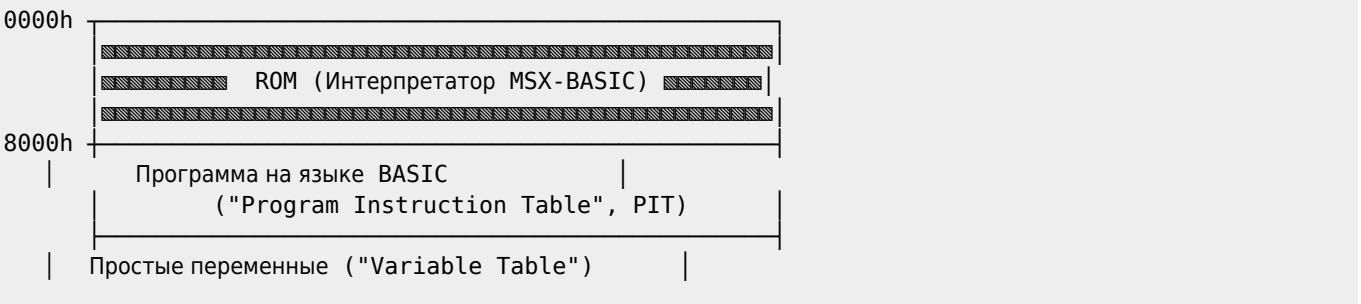
Естественно, что начальный_кластер — это *первый кластер следующего файла!* Потом освободили Справочник от бесполезной информации, находящейся в 9-м секторе и записали оглавление и ТРФ на дискету.

На этом восстановление уникальной информации благополучно завершилось.

А теперь(специально для любителей попрограммировать на языке MSX BASIC) покажем, как хранится на диске файл с программой, а потом приведем пример восстановления такого файла. Сначала — необходимое введение...

Таблица программных команд (PIT — Program Instruction Table)

PIT — это место в оперативной памяти компьютера, где хранится Ваша программа, преобразованная MSX BASIC–системой во внутренний код. Следующая табличка представляет собой карту памяти слота 0 (слот «MSX BASIC»), в котором все изменения, редактирование и запуск программ производятся под управлением интерпретатора языка.



```
      ├──────────────────────────────────────────────┤
   │    Массивы ("Array Variable Table")          │
      ├──────────────────────────────────────────────┤
   │     Свободная область ("Free Area")          │
      ├──────────────────────────────────────────────┤
   │        Стек ("Stack Area")                   │
      ├──────────────────────────────────────────────┤
   │  Строковая область ("Character String Area")  │
      ├──────────────────────────────────────────────┤
   │ Блок управления файлами ("Files Control Block")│
F380h ├──────────────────────────────────────────────┤
   │          Рабочая область                     │
   │   (системные переменные и таблица ловушек)   │
FFFFh └──────────────────────────────────────────────┘
```

Таблица PIT обычно начинается по адресу &H8000. Однако ее можно «сдвинуть», изменив значение системной переменной TXTTAB в таблице системных переменных. Для помещения PIT с адреса &HA000 достаточно выполнить следующую программу:

```
5 'Адрес  &HA001, находящийся в двух ячейках с номерами,
  начиная с  &hF676,  определяет место, с которого начнется текст программы.
10 POKE &HF676,&H01   'Заполнение младшего байта слова  TXTTAB
20 POKE &HF677,&HA0   'Заполнение старшего байта слова  TXTTAB
30 POKE &HA000,0      'Первый байт  PIT(&HA000)  должен быть нулевым!
40 NEW                'Стираем данную программу!
```

Напомним Вам, что адрес &HA001 (как и любой другой!) размещается в двух байтах так:

```
  Адрес    ──►   &HF676     &HF677   ◄──   Адрес
младшего байта    │     │    старшего байта
                  ┌─────┴─────┴─────┐
Содержимое   │ ┌──▼──┐  ┌──▼──┐ │   Содержимое
 младшего ◄──────── &H01 │   │ &HA0 ────────► старшего
  байта    │ └─────┘  └─────┘ │    байта
             └─────────────────┘
```

Очевидно, что в результате этих «манипуляций» размер свободной области уменьшится на &H2000 байтов (&HA000-&H8000=&H2000), и область, расположенная между &H8000 и началом PIT, будет защищена от «вторжения» программ на MSX BASIC. Ясно, что величина PIT зависит от размера текста программы. После выполнения данной программы нажмите кнопку сброса «RESET». А теперь мы поведаем Вам о том, как хранится программа, написанная на языке MSX BASIC в PIT.

Все строки программы на MSX BASIC начинаются с *2-байтного указателя.* За этим указателем идут *два байта, содержащие номер строки.* Затем идет *текст строки с* последующим *нулевым байтом*. *За последней строкой следуют два* дополнительных *нулевых байта*, адрес которых находится в указателе последней строки программы. Цифры и зарезервированные служебные слова записываются во внутреннем коде (один или два байта на слово, цифру). Для остального текста используется код ASCII.

Введем в память следующую короткую программу:

```
10 B=5
20 END
```

Теперь прочитаем, что же реально содержится в PIT, используя в непосредственном режиме команду

```
PRINT HEX$(PEEK(A))
```
,

где значение переменной A (адреса) изменяется от &H8000 до &H8010.

Вы обнаружите:

```
┌──────────┬──────────┬──────────────────────────────────────────────┐
│Значение A│HEX$(PEEK(A))│                Комментарии                   │
├──────────┼──────────┼──────────────────────────────────────────────┤
│  8000    │    0     │    Первый байт PIT всегда нулевой            │
├──────────┼──────────┼──────────────────────────────────────────────┤
│•• 8001 ••│   09     │   Ссылка на начало следующей строки          │
│•• 8002 ••│   80     │(указатель следующей строки находится по      │
│••••••••••│          │        адресу &H8009)                        │
├──────────┼──────────┼──────────────────────────────────────────────┤
│  8003    │    A     │  "Визуальный" номер первой строки            │
│    8004  │      0   │                  &H000A = 10                 │
├──────────┼──────────┼──────────────────────────────────────────────┤
│  8005    │   42     │ Шестнадцатеричный код ASCII буквы "B"        │
│    8006  │     EF   │       Внутренний код знака равенства         │
│  8007    │   16     │    Внутренний код цифры 5                    │
│  8008    │    0     │    Конец первой строки                       │
├──────────┼──────────┼──────────────────────────────────────────────┤
│••  8009 ••│     0F   │      Ссылка на начало следующей строки       │
│••  800A ••│     80   │ (указатель следующей строки находится по     │
│••••••••••│          │       адресу &H800F)                         │
├──────────┼──────────┼──────────────────────────────────────────────┤
│    800B  │     14   │      "Визуальный" номер первой строки        │
│    800C  │     00   │                  &H0014 = 20                 │
├──────────┼──────────┼──────────────────────────────────────────────┤
│    800D  │     81   │        Внутренний код оператора END          │
│  800E    │   00     │      Конец второй строки                     │
├──────────┼──────────┼──────────────────────────────────────────────┤
│    800F  │      0   │                Конец программы               │
│    8010  │      0   │                                              │
└──────────┴──────────┴──────────────────────────────────────────────┘
```

Теперь, надеемся, Вам стало ясно, как можно изменить программу с помощью оператора POKE.

Попробуйте выполнить следующее:

```
POKE &H8005,&H41   '41 - шестнадцатеричный код ASCII буквы "A"
POKE &H8007,&H17   '17 - внутренний код цифры "6"
```

А теперь наберите команду LIST, затем нажмите клавишу Ввод ⏎ и ...:

```
10 A=6
20 END
```

Теперь Вам ясно, что «инструкции» PEEK и POKE таят в себе поистине безграничные возможности. По существу, они позволяют нам распоряжаться памятью компьютера по своему усмотрению.

Например, они позволяют нам при желании подшутить над компьютером: если известно, где хранится программа, то мы можем сделать так, что после одной из строк программы окажется строка с меньшим номером. Пусть исходная программа имеет вид:

```
10 PRINT 4
20 PRINT 2
```

Вам, конечно, уже известно, что строки программы на языке MSX BASIC начинаются с двухбайтного указателя, за которым следуют два байта, содержащие номер строки. Поэтому вначале выполним команду:

```
PRINT HEX$(PEEK(&H8002));" ";HEX$(PEEK(&H8001))
80 9
Ok
```

Таким образом, указатель следующей (с номером 20) строки располагается в ячейках с адресами &H8009 и &H800A,

а, следовательно, номер второй строки находится в ячейках с адресами &H800B и &H800C.

Проверим этот факт:

```
PRINT HEX$(PEEK(&H800C));" ";HEX$(PEEK(&H800B))      PRINT &H14
0 14                                                  20
Ok                                                   Ok
```

А теперь:

```
POKE &H800B,1
Ok
list
10 PRINT 4
1 PRINT 2
Ok
```

Программа действует, но строку с номером 1 нельзя ни стереть, ни исправить. Вы можете написать еще одну 1–ю строку и даже новую 20–ю строку!

Однако не пытайтесь изменять с помощью оператора POKE длину строки или путать указатели: результат будет катастрофическим!

Ну а если Вы нечаянно нажали RESET, — не спешите отчаиваться! Вашу программу еще можно спасти. Это очень легко сделать, набрав ту же команду

```
POKE &H8001,1
```

а затем

```
auto
```

.

На экране появятся строки:

```
10*
20*       и так далее ...
```

Строки с * — спасенные. Теперь достаточно «скомандовать»: LIST и... о, чудо! Но это еще не все! Оказывается, спасены и все строки между теми, номера которых не делятся нацело на 10!

Если же Вы захотите защитить свою программу от запуска(команды RUN), то примените в непосредственном режиме команду:

```
POKE &H8000,1
Ok
```

(разумеется, Ваша программа должна располагаться с адреса &H8000).

По команде SAVE компьютер выполняет следующие простые действия...

Программа сохраняется в файле на диске таким образом:

- α) значение первого байта файла устанавливается в &HFF и
- β) текст программы во внутреннем представлении копируется из PIT на дискету.

Надеемся, что эти примеры немного развлекли Вас и переходим к цели нашего повествования...

Пример номер два

На диске находилась сложная программа, сохраненная во внутреннем коде. Однажды «Некто» под тем же именем сохранил пустую программу. Копии большой программы не было, и по этому ее нужно было восстановить во что бы то ни стало. При просмотре диска мы обнаружили некоторые интересные вещи:

1. 1) объем файла стал равным трем байтам (содержимое: FF,00,00);
2. 2) за этими тремя байтами следовали еще 253 неизвестных байта;
3. 3) остальная информация сохранилась полностью, восстановления требовали только FAT и директории.

Программу можно было попытаться спасти,но вызывал затруднения следующий нюанс: содержимое «испорченного» сектора будет «портить» всю программу; при использовании команды LIST на экране может появиться все, что угодно, но только не Ваша программа!

Как избежать этого? Логично было бы вручную исправить начальный сектор файла таким образом, чтобы не потерять ни одной ссылки, ни одного байта из оставшейся части программы.

Восстановление началось с того, что мы определили цепочку ссылок в FAT, вписали прежний размер файла в каталог и сохранили все это на диске. Затем необходимо было чем-то заполнить первые 256 байт, чтобы потом эту программу можно было загрузить в MSX BASIC–системе. Помня, что в строке может быть не более 255 символов, мы сделали 2 строчки, заполнили их кодами &HF1 (внутренний код символа «+») и расставили ссылки на следующие строки. Весь сектор (256 байт) был заполнен примерно таким образом:

Адрес байта	Содержимое
000	FF — первый стандартный для файлов на BASIC байт
001	05 ┐ 8105 — адрес след. строки
002	81 ┘
003	01 ┐ 0001 — номер строки
004	00 ┘
005	F1 ┐
•••	••• │ строка из 255 символов "+"
103	F1 ┘
104	00 — конец строки
105	FC ┐ 81FC — адрес след. строки
106	81 ┘
107	02 ┐ 0002 — номер строки
108	00 ┘
109	F1 ┐
•••	••• │ строка из 242 символов "+"
1FA	F1 ┘
1FB	00 — конец строки
1FC	?? ┐ 82?? — неизвестный адрес
1FD	82 ┘ след. строки (его определяют по содержимому 2-го сектора)
1FE	03 ┐ 0003 — номер строки
1FF	00 ┘

Обратите внимание на тот факт, что адресу 1FC должна лежать ссылка на существующую строку программы, поэтому перед тем, как заполнить этот байт, Вы должны внимательно просмотреть начало следующего сектора (приблизительно 252 байта) и найти адрес (в разделе ENTRY) байта, стоящего перед байтом с содержимым 82h!

Только после этого можно загружать программу в MSX BASIC и, уничтожив две первые строки, вспоминать, каким было начало Вашей программы!

I.2.4. Область данных на дискете. Размещение файлов

> Если Вы не тот, кто наверху, значит, Вы тот, кто внизу.
>
> *—Стивен Поттер*

Область данных (область для хранения содержимого файлов) расположена сразу после Справочника:

- α) с сектора 0Eh, если диск *двухсторонний*,
- β) с сектора 0Ch, если диск *односторонний*.

При форматировании в Область данных записываются символы с кодом E5h

Следует отметить, что в этой области диска элементарной единицей количества информации является *кластер* (совокупность двух секторов). Операционная система работает с кластером как с неразрывным целым. Отсюда, в частности, следует, что файл занимает на диске участок памяти, размер которого *всегда* кратен 1024 байтам.

Кластеры нумеруются следующим образом:

Номера секторов	0Ch÷0Dh	0Eh÷0Fh	10h÷11h	...	59Bh÷59Ch	59Dh÷59Eh
Номер кластера	02h	03h	04h	...	712h	713h

Номер первого кластера с содержимым файла находится в Справочнике диска. Ссылки на остальные кластеры находятся в Таблице Размещения Файлов.

Работая с Таблицей Размещения Файлов можно легко восстановить файл, стертый командой `DEL` или любой аналогичной командой, т.к. при уничтожении файла происходит следующее:

1. код первого символа имени файла устанавливается равным E5h;
2. стирается (обнуляется) вся информация в FAT, относящаяся к этому файлу, а место в ней освобождается для информации о новом файле;
3. информация из «стертого» файла сохраняется на дискете до тех пор, пока на дискету не будет записан новый файл. И до тех пор, пока на диск не записан новый файл, старый файл *можно восстановить* (если, конечно, Вы знаете, в каких кластерах он размещался до «стирания»!).

Теперь мы можем рассмотреть метод выделения кластеров для файлов.

При записи на дискету данных из файла для них по одному выделяются кластеры дискеты. Когда необходим очередной кластер для данных, то выбирается доступный кластер с наименьшим номером. Такая простая схема используется как при создании файла, так и при его «удлинении» (путем добавления данных в конец файла).

Операционная система MSX-DOS выделяет место на дискете по одному кластеру, когда в этом есть необходимость, не заботясь о том, чтобы все данные файла хранились в одной непрерывной области диска. При разработке любой схемы распределения дискового пространства в любой операционной системе приходится выбирать между свободным выделением пространства по одному кластеру (в результате файл может оказаться «размазанным» по всей дискете), и выделением места большими непрерывными сегментами, что усложняет задачу управления пространством на дискете. Операционная система MSX-DOS использует более простой первый метод.

Если дискета пуста, то все доступное на ней место представляет собой одну большую «чистую» область. Когда файлы копируются на такую дискету, они оформляются в виде удобно размещенных *непрерывных* сегментов дискового пространства. Но впоследствии, если файлы будут «удлиняться», то дополнительное место будет выделяться в первых свободных кластерах, которые могут находиться в *любом* месте дискеты.

Когда файлы копируются на новую дискету и не изменяются программистом, их размещение на диске остается экономичным. Но если создаются или удаляются какие-либо данные, то использование места на дискете становится

весьма запутанным. Это очень часто происходит, когда программа создает одновременно два файла. Если файлы увеличиваются параллельно, то их расположение на дискете обязательно будет перемежающимся.

Итак, при распределении пространства на дискете операционная система MSX-DOS равномерно использует все пространство и не требует Вашего вмешательства. Однако все эти преимущества достигаются ценой «размазывания» файлов по дискете. Из-за этого может увеличиться время доступа к файлу, поскольку магнитная головка чтения-записи должна перемещаться по всей дискете для поиска нужных треков. Поскольку обращения к дискете являются наиболее медленными операциями и основными факторами, ограничивающими производительность компьютера, такое размещение файлов может породить проблемы, хотя обычно этого не случается.

По этой причине необходимо уделять внимание потенциальной проблеме фрагментации пространства, занимаемого файлами на дискете. В тех случаях, когда фрагментация файлов покажется Вам нежелательной, скопируйте файл на новую дискету. Это же полезно делать, если Вам покажется, что дисковод слишком долго выполняет поиск треков при чтении файла.

При копировании файлов на новую дискету можно заодно улучшить их последовательность, разместив в начале часто использующиеся файлы или переставив их в таком порядке, в каком Вам хотелось бы их видеть в распечатке содержимого дискеты.

Теперь мы расскажем Вам о структуре текстовых файлов.

Текстовый файл в коде ASCII — это наиболее часто использующийся и один из самых важных форматов для файлов данных, размещаемых на дискетах. Такие файлы содержат обычную алфавитную информацию, тексты или отчеты, состоящие из алфавитных символов.

Большинство текстовых редакторов используют текстовый формат. Текстовый формат используется для хранения исходного текста программ (исходной, нетранслировавшейся версии). Практически все языковые процессоры (такие, как интерпретатор языка MSX BASIC, компилятор языка Pascal или Макроассемблер) ожидают ввода файла в текстовом формате.

Вообще говоря, имеется три различных категории кодов ASCII и две из них будут нас интересовать.

Одна категория используется для кодировки данных, таких как буквы, цифры и знаки пунктуации.

Код	Клав.	Код	Клав.	Код	Клав.	Код	Клав.	Код	Клав.	Код	Клав.	Код	Клав.
32	SPACE	55	7	78	N	101	e	124	|	211	с	234	Й
33	!	56	8	79	O	102	f	125	}	212	т	235	К
34	"	57	9	80	P	103	g	126	~	213	у	236	Л
35	#	58	:	81	Q	104	h	191	¤	214	ж	237	М
36	$	59	;	82	R	105	i	192	ю	215	в	238	Н
37	%	60	<	83	S	106	j	193	а	216	ь	239	О
38	&	61	=	84	T	107	k	194	б	217	ы	240	П
39	'	62	>	85	U	108	l	195	ц	218	з	241	Я
40	(	63	?	86	V	109	m	196	д	219	ш	242	Р
41	)	64	@	87	W	110	n	197	е	220	э	243	С
42	*	65	A	88	X	111	o	198	ф	221	щ	244	Т
43	+	66	B	89	Y	112	p	199	г	222	ч	245	У
44	,	67	C	90	Z	113	q	200	х	223	ъ	246	Ж
45	-	68	D	91	[	114	r	201	и	224	Ю	247	В
46	.	69	E	92	\	115	s	202	й	225	А	248	Ь
47	/	70	F	93	]	116	t	203	к	226	Б	249	Ы
48	0	71	G	94	^	117	u	204	л	227	Ц	250	З
49	1	72	H	95	_	118	v	205	м	228	Д	251	Ш
50	2	73	I	96	`	119	w	206	н	229	Е	252	Э
51	3	74	J	97	a	120	x	207	о	230	Ф	253	Щ
52	4	75	K	98	b	121	y	208	п	231	Г	254	Ч
53	5	76	L	99	c	122	z	209	я	232	Х		
54	6	77	M	100	d	123	{	210	р	233	И		

Вторая категория используется для форматирования, т.е. для указания места, где заканчивается одна строка и

начинается другая; эти коды используются не только для управления форматом печати, но и для структурирования файлов.

И, наконец, *третья* категория служит для управления передачей данных и никак не связана с форматом текстовых файлов.

Символы ASCII двух специальных категорий — форматирующие и символы связи, — имеют коды от 0 до 31 и 127.

Код	Клавиша (и)	Код	Клавиша (и)	Код	Клавиша (и)
0	CTRL+@	11	CLS/HOME	22	CTRL+V
1	CTRL+A	12	CTRL+L	23	CTRL+W
2	CTRL+B	13	RETURN	24	SELECT
3	CTRL+C	14	CTRL+N	25	CTRL+Y
4	CTRL+D	15	CTRL+O	26	CTRL+Z
5	CTRL+E	16	CTRL+P	27	ESC
6	CTRL+F	17	CTRL+Q	28	⟶
7	CTRL+G (BEEP)	18	INS	29	⟵
8	BS	19	CTRL+S	30	▲ стрелка вверх
9	TAB	20	CTRL+T	31	▼ стрелка вниз
10	CTRL+J (LF)	21	CTRL+U	127	DEL

Теперь рассмотрим структуру *текстового* файла в коде ASCII. Символы текстового файла организуются в *строки*, подобно строкам на бумаге. Границы строк отмечаются форматирующими символами кода ASCII:

- *возврат каретки* — символ с кодом 13 (0Dh) и
- *перевод строки* — символ с кодом 10 (0Ah).

Можно было бы использовать любой символ для обозначения конца строки, но используются именно эти два символа, поскольку они управляют процессом перехода к новой строке на печатающем устройстве.

Одной из причин использования двух символов, обозначающих конец строки, состоит в обеспечении возможности наложения строк. Наиболее часто такая возможность используется для выполнения подчеркивания. При этом на печать в конце строки выдается только символ «возврат каретки», а затем накладывающиеся символы. Поскольку перевод строки не выполнялся, новые символы печатаются поверх уже напечатанных. Однако заметим, что такую операцию можно произвести с печатающим устройством, но нельзя с дисплеем.

Формат текстового файла определяется не только разделением на строки. Все файлы в коде ASCII имеют один общий символ *маркер конца файла*, код ASCII которого равен 26 (1Ah). Этот маркер однозначно указывает конец файла. Обнаружение положения кода 26 позволяет точно установить размер файла, отличающийся от размера, указанного в Справочнике.

---

https://sysadminmosaic.ru/msx/floppy_disk_filesystem_structure/010

2023-06-04 12:52