Главная Загрузочная Запись - Master Boot Record (MBR)

Для начала давайте определимся с терминологией. Под словами, вынесенными в заголовок статьи, зачастую понимают две разные вещи, которые в некоторых случаях могут быть эквивалентны, а в некоторых - нет. Первое понятие - собственно Master Boot Record. Это запись (программный код и данные), которая загружается в память с винчестера и обеспечивает опознание логических разделов на нем, определяет активный раздел и загружает из него загрузочную запись (Boot Record - BR), которая продолжит запуск операционной системы (ОС). И второе понятие - Загрузочный Сектор, Master Boot Sector (MBS) - это сектор, располагающийся на цилиндре 0, плоскости (головка) 0 и имеющий номер 1. В большинстве случаев MBS содержит весь необходимый код и все данные, поэтому его содержимое и есть MBR, однако бывают случаи (которые мы рассмотрим в конце статьи), когда код и данные не помещаются в одном секторе (просто не хватает места или по соображениям безопасности), и тогда код этого сектора обеспечивает загрузку в память остальных секторов. В этом случае MBR - это совокупность всех секторов, которые должны быть загружены, а MBS - всего лишь первый сектор.

Однако начнем мы со случая, когда MBR и MBS - одно и то же, и будем называть их более привычным и широко распространённым термином MBR. Слегка отступая от темы, замечу, что такой MBR (обеспечивающий загрузку любой ОС и занимающий только MBS) обычно называют термином Generic MBR.

Вообще MBR появился на жестких дисках начиная с MS DOS версии 3.0, в более ранних версиях жёсткий диск форматировался как дискета, и в первом секторе располагался BR. Соответственно диск представлял из себя один раздел и не мог быть разбит на логические части - правда, при тех размерах дисков, которые тогда выпускались, это было неактуально.

Размер сектора на жестком диске - 512 байт. Этого пространства вполне хватает для размещения там всего необходимого - и кода, и данных. Однако только одна структура должна там присутствовать обязательно - это сигнатура. Этим словом называется специальная, строго установленная, последовательность из 2 байт с шестнадцатеричными значениями 55h AAh, которая записывается в последние 2 байта сектора и соответственно имеет смещение от начала сектора 1FEh. Если хотя бы один из двух последних байтов отличается по значению, считается, что первый сектор не является MBR и не содержит осмысленной информации. Если компьютер при старте, прочитав первый сектор, не обнаружит правильной сигнатуры, он не будет передавать управление располагающемуся там коду, даже если он правильный, а выдаст сообщение о том, что главная загрузочная запись не найдена. Или будет пробовать найти её на других устройствах - например, на дискете. Слегка отклоняясь от темы, замечу, что BR также содержит сигнатуру 55h AAh в последних двух байтах.

Ну уж коли начали с хвоста, то пойдём от него к началу сектора. Перед сигнатурой, вплотную к ней, расположены 4 блока данных по 16 байтов каждый (соответственно со смещением от начала сектора 1BEh, 1CEh, 1DEh, 1EEh). Совокупность этих блоков называется Таблица Разделов, Partition Table (PT), а каждая отдельная запись - элементом таблицы разделов (Partition Table Entry) или просто разделом (Partition). Этих 16 байтов вполне достаточно, чтобы указать все необходимые характеристики раздела, а именно: тип раздела, признак активности раздела, начальный и конечный сектора раздела в формате Цилиндр (дорожка) - Головка (сторона) - Сектор (Cylinder - Head - Sector, CHS), относительный номер первого сектора (относительно MBR) и количество секторов в разделе.

Всё остальное пространство сектора занято программным кодом, который обеспечивает разбор PT, поиск активного раздела, загрузку в память BR этого раздела и передачу ему управления. Как легко подсчитать, на код остаётся 512 - 4 * 16 - 2 = 446 байт. Этого

пространства с избытком хватает для выполнения указанных действий.

Итак, общая структура MBR может быть представлена следующей таблицей:

Смещение	Длина	Описание
000h	446	Код загрузчика
1BEh	64	Таблица разделов
	16	Раздел 1
1CEh	16	Раздел 2
1DEh	16	Раздел 3
1EEh	16	Раздел 4
1FEh	2	Сигнатура (55h AAh)

Каждый 16-байтный блок, описывающий один раздел, имеет следующую структуру:

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

Код типа раздела представляет собой однобайтовый идентификатор. Если его значение - 00h, то считается, что в данном элементе PT не содержатся данные о разделе, и его содержимое игнорируется. Любое ненулевое значение означает, что в указанном пространстве находится раздел определённого типа. Некоторые значения однозначно указывают тип раздела, некоторым соответствуют несколько возможных типов, и определение конкретного типа возлагается на операционную систему, остальные зарезервированы для будущего использования. Сравнительно полный и актуальный справочник по кодам типов разделов можно найти в Ralf Brown Interrupt List в файле INTERRUPT.D, таблица 00652, который содержится в архиве (на момент написания статьи) inter61a.zip по адресу http://www.pobox.com/~ralf/files.html. Здесь же я приведу таблицу тех типов разделов, которые создаются операционными системами Windows 9x и Windows NT/2000/XP:

Код	Тип раздела
01h	12-битная FAT
04h	16-битная FAT до 32 Мбайт
05h	Расширенный раздел
06h	16-битная FAT свыше 32 Мбайт
07h	Windows NT NTFS (и некоторые другие – тип определяется по содержимому BR)
0Bh	32-битная FAT
0Ch	32-битная FAT с использованием расширенного управления INT13
0Eh	LBA VFAT (то же что и 06h, с использованием расширенного управления INT13)
0Fh	LBA VFAT (то же что и 05h, с использованием расширенного управления INT13)
17h	Скрытый раздел NTFS
1Bh	Скрытый раздел 32-битной FAT (то же что 0Bh)
1Ch	Скрытый раздел 32-битной FAT с использованием расширенного управления INT13 (то же что 0Ch)
1Eh	Скрытый раздел LBA VFAT (то же что и 06h, с использованием расширенного управления INT13)
86h	Раздел FAT-16 stripe-массива Windows NT
87h	Раздел NTFS stripe-массива Windows NT
B6h	Зеркальный master-раздел FAT-16 Windows NT
B7h	Зеркальный master-раздел NTFS Windows NT
C6h	Зеркальный slave-раздел FAT-16 Windows NT

C7h Зеркальный slave-раздел NTFS Windows NT

Признак активности раздела - т.е. признак того, что операционную систему следует загружать именно из этого раздела - может иметь значения 80h (раздел активен) и 00h (раздел не активен). В общем случае количество активных разделов должно быть не более 1 (иначе как сделать выбор?). Если активных разделов нет - значит с этого жёсткого диска ОС не может быть загружена. Другие значения считаются ошибочными и игнорируются. Впрочем, решение о передаче управления принимает код загрузчика, поэтому значение байта признака загрузочности - аксиома только для стандартных загрузчиков.

Трёхбайтный блок адреса начала и адреса конца раздела имеют идентичный формат. Здесь фактически используется упаковка значений с тем, чтобы они имели минимальный объём. Формат упаковки полностью соответствует тому, как эти данные передаются процедурам работы с жёстким диском (Int 13h), находящимся в BIOS компьютера, поэтому и накладные вычислительные расходы получаются минимальными. При этом цилиндры и дорожки нумеруются, начиная с нулевого значения, а сектора - почему-то с первого. Уж и не знаю почему - так сложилось исторически.

Сектор, на который указывает адрес начала раздела, содержит в себе специальную запись, которая называется загрузочной записью (BR). Её назначение и состав мы рассмотрим в отдельной статье.

Смещение первого сектора раздела - это фактически номер этого сектора если все сектора жёсткого диска перенумеровать начиная с 0 (в соответствии с нумерацией, применяемой Int 25h/26h) в порядке возрастания сперва по секторам одной дорожки, далее в порядке увеличения номеров головок и, наконец, цилиндров. А что такое количество секторов в разделе - понятно без объяснений.

Естественно, что все эти значения связаны простыми зависимостями, ведь содержащаяся в них информация избыточна. Потому приведу формулы зависимости между ними.

Итак, если обозначить:
C_M - цилиндр, на котором располагается MBR;
H_M - дорожка, на которой располагается MBR;
S_M - сектор, в котором располагается MBR;
C_S, H_S, S_S, C_E, H_E, S_E - то же, для секторов начала (S) и конца (E) раздела;
H_H - количество дорожек у жёсткого диска;
S_H - количество секторов на одной дорожке у жёсткого диска,
то:
Абсолютный номер сектора, в котором находится PT:
$Num_{PT} = C_M * H_H * S_H + H_M * S_H + S_M - 1$
Абсолютный номер сектора начала раздела:
$Num_S = C_S * H_H * S_H + H_S * S_H + S_S - 1$
Абсолютный номер сектора конца раздела:
$Num_E = C_E * H_H * E_H + H_E * E_H + E_E - 1$
Смещение первого сектора раздела:
$Offset_S = Num_S - Num_{PT}$
Количество секторов раздела:
$Amount = Num_E - Num_S + 1$

Из приведённых формул, кроме того, видно, что важное значение имеет то, сколько дорожек имеет один цилиндр жёсткого диска и сколько на каждой дорожке секторов. Эти значения зависят как от геометрии жёсткого диска, так и от выбранного в установках BIOS режима трансляции. Поэтому диск, поделённый на разделы в одном режиме трансляции, может оказаться нечитаемым при изменении режима трансляции.

Для IDE-накопителей существуют несколько режимов трансляции:

CHS (Cylinder-Head-Sector) - при этом геометрия диска считается такой, какой он её сообщает компьютеру. Не обольщайтесь - у большинства накопителей реальная геометрия совсем не такая. Однако контроллер, входящий в состав жёсткого диска, производит

необходимые преобразования самостоятельно, и то, как он это делает, для нас несущественно. При этом максимальное количество цилиндров - 1024 (от 0 до 1023), дорожек - 16 (от 0 до 15), секторов - 63 (от 1 до 63), а максимальный объём диска, доступный в данной трансляции без применения специальных программ - 504 Мбайт. При дисковых операциях адрес сектора передается BIOS компьютера контроллеру жёсткого диска без изменения.

LBA (Logical Block Addressing) - при этом режиме трансляции используется не та геометрия диска, которую он сообщает BIOS компьютера. Производится приведение к формату, когда количество цилиндров не превышает 1024, а количество секторов на дорожку равно 63. Приведённое количество дорожек при этом зависит от BIOS компьютера и объёма жёсткого диска и может быть равно 16, 64 , 128 или 255, последние версии BIOS как правило используют приведение к 255 (0-254) дорожкам независимо от объёма накопителя. При обращении к диску переданные в LBA-трансляции номер цилиндра, головки и сектора пересчитываются в абсолютный номер сектора, и именно он передаётся BIOS компьютера контроллеру жёсткого диска для выполнения операции. При объёме накопителя свыше 8 Гбайт количество цилиндров получается более 1024, потому на компьютерах, которые не поддерживают работу расширенного режима Int 13h, без установки специального программного обеспечения пространство за границами 8 Гбайт недоступно (независимо от режима трансляции). Иногда, впрочем, помогает обновление BIOS компьютера.

LARGE, или ECHS (Extended CHS) - при этом режиме трансляции производится приведение числа цилиндров к значению менее 1024 за счёт кратного увеличения количества дорожек. Приведённое количество дорожек строго кратно реальному количество секторов на дорожку не изменяется. Впрочем, у большинства современных накопителей количество секторов на дорожку (по уверениям контроллера накопителя) равно 63. Максимально доступный объём на компьютерах, которые не поддерживают работу расширенного режима Int 13h, зависит от реального количества дорожек, но не более 8 Гбайт (например, если диск имеет 16 дорожек, то в LARGE трансляции их может быть 16, 32, 48… 240, но не 255, т.к. 255 не кратно 16, а максимальный доступный объём - $1024 * 240 * 63 * 512 / (1024^3) = 7.38$ Гб).

При использовании трансляции LBA или LARGE за счёт округления до целых значений несколько секторов в конце накопителя могут "выпасть" из описанного пространства и стать недоступными. Впрочем, потери обычно невелики.

Что же касательно SCSI-накопителей, то они всегда работают в режиме LBA-трансляции. Вернее, контроллер SCSI представляет геометрию накопителя в соответствии с требованиями LBA-трансляции, а сам при обращении к диску передаёт ему абсолютный адрес сектора. Естественно, что именно на BIOS SCSI-контроллера возлагается обязанность производить необходимые пересчёты.

Вернёмся к MBR. Как уже сказано, он содержит 4 блока данных об элементах таблицы разделов. Это означает, что максимальное количество разделов, которое может быть описано в MBR, равно четырём. Однако это отнюдь не означает, что максимальное число разделов, на которые может быть разделён накопитель, равно четырём. Для преодоления этого барьера был введён специальный тип раздела с кодом 05h - расширенный раздел (Extended Partition).

Расширенный раздел сильно отличается от всех остальных типов разделов. Во-первых, он описывает не раздел, а область пространства накопителя, в которой расположены другие разделы. При этом количество находящихся в нём разделов теоретически не ограничено. Правда, те разделы, которые расположены в этой области, несколько "ограничены в правах", самым существенным ограничением является то, что они не могут быть активными (вернее, можно сделать так, что из такого раздела ОС будет загружена, но штатные средства большинства существующих ОС этого не позволяют, придётся использовать специальные средства). Во-вторых, в MBR должна присутствовать только одна запись о расширенном разделе. Вернее, их можно сделать и больше (хоть все

четыре), но как поведёт себя ОС, встретив такое, предсказать трудно. Например MS-DOS 6.20 просто игнорирует все расширенные разделы, кроме первого в списке, как будто их вообще нет. В третьих, в отличие от остальных типов разделов в том секторе, который прописан в структуре как сектор начала раздела, содержится отнюдь не BR. Там находится фактически еще один MBR, который имеет сигнатуру и таблицу разделов, но обычно не содержит программного кода (обычно сектор, содержащий таблицу разделов, но не содержащий кода начальной загрузки, называют Abstract MBR). Впрочем, поскольку там нет активных разделов, то и код ни к чему. В таблице разделов такого сектора имеется обычно одна или две записи. Первая описывает обычный раздел (Partition), причём этот раздел должен полностью находиться внутри пространства Extended Partition. Если обычный раздел занимает не всё пространство, в таблице разделов появляется второй элемент, который описывает оставшееся пространство как Extended Partition. В следующем секторе точно также описывается один раздел и, если место осталось, еще запись об Extended Partition. И так продолжается до тех пор, пока пространство не закончится. Фактически все записи о расширенных разделах представляют собой связанную цепь (Extended Partition Chain), в которой от дискового пространства отщипываются кусочки на обычные разделы, пока место не кончится. Ошибка в любом элементе этой цепи приведёт к её рассыпанию, в результате все записи после разрыва не будут найдены ОС, а занимаемое ими пространство ОС будет считать незанятым.

При заполнении цепи обычно ОС придерживается нескольких правил. Во-первых, описанный в очередном "звене" цепи обычный раздел не должен располагаться в середине, поскольку тогда для описания получившихся двух кусков незанятого пространства потребуется в PT этого элемента ввести две записи о двух разных расширенных разделах а, как я говорил ранее, ОС обычно игнорируют все такие записи кроме первой, и в результате часть дискового пространства выпадет из разбиения. Во-вторых, как правило запись об обычном разделе делается так, чтобы она занимала начальную область расширенного раздела, а следующий элемент цепи разделов - остаток.

Какие же проблемы могут возникнуть с содержимым MBR? Во-первых, физическое или логическое разрушение, т.е. повреждение поверхности или иная механическая проблема либо разрушение сервометки, что не даёт возможности прочитать этот сектор с диска. Однако подобные случаи выходят за рамки нашего рассмотрения. И вторая, наиболее часто встречающаяся проблема - это разрушение всей или части информации, содержащейся в секторе, в результате чего разделы либо не могут быть найдены операционной системой, либо их параметры определяются неверно.

Самый лёгкий случай - это разрушение сигнатуры. При этом ОС считает, что в секторе содержится некая случайная информация, "мусор", а сам накопитель вообще не поделён на разделы и никакой информации на нём нет. Для восстановления достаточно всего лишь любым средством прямого доступа к секторам диска (наиболее популярен DISKEDIT из пакета NORTON UTILITIES) восстановить сигнатуру. Большинство ОС, правда, нужно перезагрузить, поскольку ОС как правило при старте считывают информацию о разбиении диска на разделы и далее в процессе работы её изменения не учитывают.

Более сложный случай - это разрушение кода. При этом теряется возможность произвести загрузку операционной системы с накопителя, а попытка загрузки как правило заканчивается "зависанием" компьютера. В то же время если загрузиться с другого накопителя (другой жесткий диск, дискета, загрузочный CD-ROM и т.п.), то вся информация на накопителе доступна для использования. В этом случае рекомендуется использовать штатные средства восстановления кода загрузчика, которые имеются в каждой ОС. Например, в ОС Windows 9x для этой цели используется программа FDISK.EXE, запускаемая с ключом /MBR.

Разрушение кода - не столь редкий случай, как может показаться. Как правило, подобная неприятность происходит, когда на один накопитель последовательно устанавливаются разные ОС в один или разные разделы. Любая ОС при установке желает иметь в MBR собственный код, но далеко не все заботятся о сохранении того кода, который был в

секторе ранее. Например ОС Windows 9x переписывают код загрузчика, не ставя в известность пользователя и безвозвратно уничтожая старое содержимое. Впрочем, такой случай как раз не очень страшен, поскольку все загрузчики ОС очень похожи. Проблема возникает, если для разбиения накопителя на разделы использовались программные средства третьих фирм, такие как EZ-drive, ODM или например SpeedStore, для которых как раз понятия MBR и MBS не эквивалентны. Из-за другой идеологии загрузки (которую мы рассмотрим позднее) код, находящийся в MBS, выполняет другую функцию, а именно поиск и загрузку части кода MBR, располагающейся в других секторах диска. Замена кода приводит к тому, что эта функция утрачивается, и соответственно возникают проблемы различного характера. Пользователь должен быть весьма осторожен, если использует подобные программные средства - восстановление кода в таких случаях может оказаться весьма непростым делом.

И наиболее тяжёлый и неприятный случай - это разрушение самой таблица разделов. Впрочем, как известно, беда никогда не приходит одна, и чаще всего разрушаются все три компонента MBR, но именно разрушение PT приводит к наиболее тяжёлым последствиям, поскольку при этом теряется возможность доступа к хранящейся на накопителе информации. PT может быть разрушена полностью, а может и частично - т.е. часть элементов разрушена, а остальные целы. Бывают случаи, когда PT, находящаяся в MBR, цела, а разрушена запись о разделах в одном из звеньев цепи Extended Partition. Однако поскольку структура MBR и структура абсолютно идентичны (за исключением того что в элементах Extended Partition отсутствует код), методика восстановления в обоих случаях одна и та же.

Существует достаточное количество программ, умеющих восстанавливать разрушенные PT. Качество их работы различно и зависит главным образом от того, насколько сложным было разбиение диска на разделы, все ли типы разделов, которые были на диске, известны программе и нет ли кроме разрушения PT еще каких повреждений информации в других секторах, особенно в BR разделов. Однако мы не ищем лёгких путей, и если у Вас над душой не висит начальник с криками "скорее, быстрее, прыжками распечатай мне бланк, я в отпуск опаздываю!", Вы получите гораздо большее удовольствие, если самостоятельно, без всяких программ-автоматов, пользуясь только своими знаниями, сможете восстановить всю информацию. Вы даже можете просто провести все необходимые исследования, получить все цифры, которые нужно записать в PT, а потом запустить программу восстановления и после её работы убедиться, что всё посчитали верно. А может, даже и поправить результаты её работы, если она отработала не на 100%.

Итак, для работы нам потребуется: загрузочная дискета с любым DOS, на которую скопирован файл DISKEDIT.EXE, карандаш, бумага, калькулятор (впрочем, калькулятор есть в DISKEDIT) и немного мозгов. Желательно, конечно, чтобы и DOS, и DISKEDIT были посвежее. Я использую DOS 7.10 от Windows 98 SE и DISKEDIT из пакета NORTON UTILITIES 2002. Никакие драйверы нам на этом этапе не нужны, ну кроме тех случаев, когда накопитель подключен к старому SCSI-контроллеру и без загрузки драйвера просто не виден. Можно загрузить драйвер мышки - будет немного удобнее. Теперь главная мелочь - кроме указанных файлов на дискете не должно быть ничего! Это важно - если на дискете нет конфигурационного файла, DISKEDIT.EXE запустится в режиме только чтения (Read-Only) и никакие данные на диске не будут изменены, пока мы этого явно не попросим. А дискету вообще бы закрыть от записи.

Загружаемся с дискеты. Запускаем DISKEDIT.EXE. После загрузки нажимаем клавиши Alt-D (или через меню Object - Drive). Выводится окно с доступными дисковыми устройствами. Сперва укажем, что нам нужны физические устройства (Physical disks), а потом выберем нужный диск (допустим, Hard Disk 1) и нажмем ОК. При этом в качестве диапазона просмотра будут выбраны все сектора диска от первого до последнего. Это нам и нужно.

Сначала мы попросим программу выполнить просмотр диска и найти все сектора, которые могут быть элементами цепи Extended Partition или BR. И хотя при этом

DISKEDIT будет просто искать сектора, имеющие сигнатуру, а не анализировать содержимое (это мы берём на себя) - результаты могут здорово облегчить работу. Правда придётся запастись терпением - процесс это не быстрый, да к тому же каждый найденный сектор нужно будет брать на карандаш, но овчинка выделки стоит.

Итак, Tools - Find Object - Partition/Boot. Поехали. Каждый раз, когда попадается сектор с сигнатурой, поиск останавливается, на экран выводится дамп сектора, а в правом нижнем углу - номер сектора. Именно эти адреса и нужно брать на карандаш. Впрочем, если при разбиении диска на разделы не использовались особо изощрённые методы, все интересующие нас сектора будут располагаться на нулевой или первой стороне в первом секторе, т.е. Side 0 или 1, Sector 1. Остальные сектора, например какой-нибудь Cyl 12, Side 4, Sector 52 можно смело проигнорировать - это случайно. Правда, мы получаем абсолютный номер сектора, но это не страшно, для "правильных" секторов номер будет нацело делиться на количество секторов на дорожку, обычно 63 (другие значения встречаются сейчас гораздо реже - 17, 26, 40, 56, поэтому далее по тексту везде где я буду говорить о числе секторов 63, имейте в виду, что на Вашем конкретном накопителе возможно придётся использовать другое число). А эту цифру мы можем посмотреть через меню (Info - Drive Info). К сожалению, количество сторон и цилиндров там может оказаться неправильным (не соответствующим используемой трансляции), но и это не страшно. Записав номер очередного сектора, продолжаем поиск (можно через меню Tools - Find Again, можно просто Ctrl-G). И так пока не получим сообщение что объект не найден. В этот момент у нас на руках (вернее, на бумаге) все номера секторов, в которых присутствует сигнатура.

Теперь обрабатываем список, отсеивая явно случайные номера (это которые не делятся на 63), и особо выделяя пары номеров, которые различаются на 63. Эти пары - не что иное как пара из элемента Extended Partition и BR описанного в нём раздела.

Теперь отложим на минутку листок с цифрами и попытаемся вспомнить, какого размера разделы были на диске. Так, крупными мазками, 600 мегабайт, 12 гигабайт… запишем всё что помним. Если сумма не равна объёму накопителя - либо что-то забылось, либо неверно вспомнилось, либо было пространство, не принадлежавшее ни одному разделу (а что, бывает… знаю не один случай, когда десятигигабайтный жёсткий диск разбивался на компьютере, материнская плата которого не понимала более восьми гигабайт, а при апгрейде это как-то не вспомнилось… вот так 2 гигабайта и зависли). На этом этапе желательно вспомнить ещё и типы файловых систем в каждом из разделов.

Теперь возьмём оба листка и попытаемся совместить полученные данные. Мегабайт - две тысячи секторов, гигабайт - два миллиона… приблизительно. Но обычно удаётся совершенно однозначно наложить одно на другое. Полезно бывает нарисовать длинный прямоугольник, расставить на нём границы, соответствующие найденным секторам, и поделить на кусочки, соответствующие размерам разделов. Даже если однозначности нет - не беда. Разберёмся. При совмещении данных рекомендую помнить, что некоторые BR и элементы цепи Extended Partition могут быть разрушены (и соответственно не будут найдены), причём наиболее часто разрушаются BR первого (по положению на накопителе) раздела и BR активного раздела.

Впрочем, на данном этапе, пожалуй, всё… и не потому, что дальше некуда, а по другой причине - для дальнейшей работы по восстановлению требуется анализ других структур, которые располагаются уже в "найденных" разделах - это BR, FAT/MFT, каталоги и пр… мы их пока не рассматривали. Впрочем, иногда и найденной и вспомненной информации достаточно. В конце статьи приведен пример такого восстановления (пока не написан).

Иногда знание структуры разделов применяют совершенно для других целей - например, для создания своего собственного разбиения диска на разделы. Пример такого применения также есть в конце статьи.

При создании нестандартного разбиения диска на разделы, кроме требований, описанных ранее (один расширенный раздел и пр.), рекомендую также учитывать то, в каком порядке ОС MS-DOS и Windows назначают разделам буквы логических дисков

(Q51978 - Order in Which MS-DOS and Windows Assign Drive Letters). Буквы присваиваются, начиная с C: (A: и B: зарезервированы для дисководов гибких дисков, возможно виртуальных). Порядок подключения таков:

1. Раздел, с которого загружается ОС (при загрузке с жёсткого диска).
2. Первые первичные разделы остальных жёстких дисков в порядке их нумерования (инициализации) BIOS компьютера.
3. Разделы в Extended Partition жёстких дисков в порядке их нумерования (инициализации) BIOS компьютера, в порядке их записи в Partition Table дисков.
4. Остальные первичные разделы дисков, в порядке их записи в Partition Table по порядку их нумерования (инициализации) BIOS компьютера.
5. Устройства, формируемые драйверами, запускаемыми в файлах config.sys и autoexec.bat, в порядке их формирования и инициализации, если формируемому устройству не назначается в явной форме определённая буква или диапазон букв.
6. Для ОС, которые могут опознавать и подключать накопители, не инициализируемые BIOS компьютера (не описанные в установках CMOS накопители) - разделы этих накопителей в соответствии с правилами 3 и 4 в порядке инициализации накопителей операционной системой.

Разделы не известных ОС типов не инициализируются и буквы им не присваиваются.
Следует помнить, что ОС семейства Windows NT имеют штатные средства переопределения букв логических дисков.

Пример 1. Ручное разбиение на разделы.

- Параметры накопителя в LBA-трансляции (взяты из BIOS - Autodetect Hard Disk) - Cylinders 1216, Heads 255, Sectors 63, Capacity 10 Gb.
- Желаемое разбиение: система 2 Гбайт, данные - 2 Гбайт, игры и дистрибутивы - остальное. Желательно разместить системный раздел в конце диска (по тестам там самая быстрая область), игры - в начале диска (по заверениям специалистов, наиболее часто данные повреждаются именно там).
- Операционная система - Windows 98 SE rus, все разделы - FAT-32.

Исходя из желаемого разбиения, видится следующая схема: сначала расширенный раздел с двумя логическими дисками в нём - 6 и 2 Гбайт, потом первичный активный раздел 2 Гбайт, либо 3 первичных раздела в указанном порядке и с указанными размерами. Последний вариант нам не подходит (неважно по каким соображениям). Особенности работы программы FDISK выбранной ОС не позволяют выполнить разбиение штатно: если сначала мы создадим первичный раздел, он будет находиться в начале накопителя, если же мы сначала создадим расширенный раздел, то программа отказывается создавать первичный.
Принимаем решение провести разбиение с помощью FDISK насколько возможно, а затем доделать вручную. Первый этап: создать расширенный раздел с двумя дисками; второй: вручную добавить запись о первичном разделе.
Первый этап проблем не вызывает: загружаемся с дискеты, создаём extended partition размером 8 Гбайт, и в ней два логических диска - 6 Гбайт и 2 Гбайт. Перезагружаемся с дискеты, убеждаемся, что на диске появились (но недоступны - ведь мы не форматировали разделы!) диски C: и D:. Форматируем их при помощи стандартного FORMAT и в процессе форматирования убеждаемся, что диск C: имеет размер 6 Гбайт, диск D: - 2 Гбайт. Запускаем DISKEDIT и смотрим содержимое MBR. В нём имеется следующая запись:

```
             Starting location     Ending location    Relative Number of
System Boot Side Cylinder Sector Side Cylinder Sector Sectors  Sectors
EXTEND  No  1    0        1      254  972      63     63       15631182
unused  No  0    0        0      0    0        0      0        0
unused  No  0    0        0      0    0        0      0        0
unused  No  0    0        0      0    0        0      0        0
```

Нам нужно добавить запись о первичном разделе. Тип (System) будет 0Ch (FAT32x), признак загрузки установлен, начало раздела по адресу 973/0/1, конец раздела по адресу 1215/254/63, относительный сектор начала раздела 15631245 (973*255*63+0*63+1-1), относительный сектор конца раздела 19535039 (1215*255*63+254*63+63-1), количество секторов 3903795 (19535039-15631245+1). Весьма существенная тонкость - если номер цилиндра более 1023, в соответствующее поле вносится значение 1023 - увы, это максимальное значение, которое можно туда записать - а правильное значение ОС рассчитает исходя из заданного количества секторов.

Переводим DISKEDIT в режим Read-Write (Tools-Configuration) и во второй строке вписываем рассчитанные данные. После ввода и проверки всех значений выходим из DISKEDIT клавишей Esc, а на вопрос, что делать с изменениями, отвечаем - записать (write). Перезагружаем компьютер. Убеждаемся, что на диске теперь три раздела - C: - недоступен, D: - 6 Гбайт, E: - 2 Гбайт. Форматируем диск C: с переносом на него системных файлов, в процессе форматирования убеждаемся, что диск C: имеет размер 2 Гбайт. Обновляем код MBR командой FDISK /MBR, вынимаем дискету, перезагружаем компьютер, убеждаемся, что ОС загрузилась, диски C:, D: и E: доступны и имеют размеры 2, 6 и 2 Гбайт. Запускаем NDD и убеждаемся, что ошибок ни в таблице разделов, ни на дисках нет. Разбиение закончено.

Теперь таблица разделов при просмотре через DISKEDIT выглядит так:

```
             Starting location     Ending location    Relative Number of
System Boot Side Cylinder Sector Side Cylinder Sector Sectors  Sectors
EXTEND  No  1    0        1      254  972      63     63       15631182   ;
Расширенный раздел
FAT32x  Yes 0    973      1      254  1023     63     15631245 3903795    ;
Первичный раздел (C:)
unused  No  0    0        0      0    0        0      0        0
unused  No  0    0        0      0    0        0      0        0
```

Если поставить курсор на строку с записью о расширенном разделе и нажать Enter, то DISKEDIT автоматически перенесёт просмотр в сектор, на который указывает адрес начала раздела (для первичного раздела это будет BR, для расширенного - элемент цепи разделов). В нашем случае мы увидим такое содержимое элемента цепи разделов:

```
             Starting location     Ending location    Relative Number of
System Boot Side Cylinder Sector Side Cylinder Sector Sectors  Sectors
FAT32x  No  2    0        1      254  728      63     63       11711259   ;
Логический раздел (D:)
EXTEND  No  0    729      1      254  972      63     11711322 3919923    ;
Расширенный раздел
unused  No  0    0        0      0    0        0      0        0
unused  No  0    0        0      0    0        0      0        0
```

Продолжим движение по цепи разделов в следующий элемент:

```
             Starting location     Ending location    Relative Number of
System Boot Side Cylinder Sector Side Cylinder Sector Sectors  Sectors
FAT32x  No  1    729      1      254  972      63     63       3919860    ;
Логический раздел (E:)
unused  No  0    0        0      0    0        0      0        0
unused  No  0    0        0      0    0        0      0        0
unused  No  0    0        0      0    0        0      0        0
```

Очередной элемент цепи не содержит записи о расширенном разделе. Цепь закончилась.

Order in Which MS-DOS and Windows Assign Drive Letters

SUMMARY

Microsoft MS-DOS assigns drive letters to the first two physical floppy disk drives and hard disk drives it finds at boot time in a fixed sequence, including multiple partitions and logical drives on the hard disks. You cannot change this sequence.

The drive letters assigned to additional drives installed using DRIVER.SYS and other installable device drivers is dependent upon the order in which the drivers are loaded in the CONFIG.SYS file. These drive letter assignments can be influenced by changing the order of the CONFIG.SYS statements or loading "dummy" drives to "use up" drive letters.

Drive letter assignments can change when you upgrade from one Microsoft MS-DOS version to another or from an original equipment manufacturer (OEM) version of MS-DOS to another version that assigns drive letters differently. (The order in which drive letters are assigned was modified by OEMs in earlier versions of MS-DOS.) This article describes how MS-DOS assigns drive letters; it does not explain how particular OEM MS-DOS versions assign drive letters.

MORE INFORMATION

The following occurs at startup:

1. MS-DOS checks all installed disk devices, assigning the drive letter A to the first physical floppy disk drive that is found.
2. If a second physical floppy disk drive is present, it is assigned drive letter B. If it is not present, a logical drive B is created that uses the first physical floppy disk drive.
3. Regardless of whether a second floppy disk drive is present, MS-DOS then assigns the drive letter C to the primary MS-DOS partition on the first physical hard disk, and then goes on to check for a second hard disk.
4. If a second physical hard disk is found, and a primary partition exists on the second physical drive, the primary MS-DOS partition on the second physical hard drive is assigned the letter D. MS-DOS version 5.0, which supports up to eight physical drives, will continue to search for more physical hard disk drives at this point. For example, if a third physical hard disk is found, and a primary partition exists on the third physical drive, the primary MS-DOS partition on the third physical hard drive is assigned the letter E.
5. MS-DOS returns to the first physical hard disk drive and assigns drive letters to any additional logical drives (in extended MS-DOS partitions) on that drive in sequence.
6. MS-DOS repeats this process for the second physical hard disk drive, if present. MS-DOS 5.0 will repeat this process for up to eight physical hard drives, if present. After all logical drives (in extended MS-DOS partitions) have been assigned drive letters, MS-DOS 5.0 returns to the first physical drive and assigns drive letters to any other primary MS-DOS partitions that exist, then searches other physical drives for additional primary MS-DOS partitions. This support for multiple primary MS-DOS partitions was added to version 5.0 for backward compatibility with the previous OEM MS-DOS versions that support multiple primary partitions.
7. After all logical drives on the hard disk(s) have been assigned drive letters, drive letters are assigned to drives installed using DRIVER.SYS or created using RAMDRIVE.SYS in the order in which the drivers are loaded in the CONFIG.SYS file. Which drive letters are assigned to which devices can be influenced by changing the order of the device drivers or, if necessary, by creating "dummy" drive letters with DRIVER.SYS.

The MS-DOS utility SUBST, networks and programs such as the CD-ROM Extensions which use the MS-DOS network interface can request a specific drive letter be assigned to a block device.

Example 1

Consider as an example a system with one floppy disk drive and one hard disk drive, with two MS-DOS partitions (a primary partition and an extended partition containing a single logical drive) on the hard disk. In this configuration, MS-DOS will assign the floppy disk drive as drives A and B, the primary partition on the hard disk drive as drive C, and the logical drive in the extended partition as drive D.

Example 2

Consider another system with three floppy disk drives, the third drive being installed using DRIVER.SYS, and two hard disk drives, with a primary and an extended partition on each hard disk drive. The extended partition on the first hard disk drive contains two logical drives, and the extended MS-DOS partition on the second hard disk drive

contains one logical drive. A RAM disk is also created using RAMDRIVE.SYS.

In this configuration, MS-DOS will assign the first two floppy disk drives as drives A and B, then assign the primary partitions on the first and second physical hard disk drives as drives C and D, respectively. MS-DOS will then assign the drive letters E and F to the two logical drives in the extended partition on the first physical drive, and G to the logical drive in the extended partition on the second physical drive.

The third floppy disk drive, installed using DRIVER.SYS, and the RAM disk created using RAMDRIVE.SYS, will be assigned the letters H and I in the order in which the DEVICE= statements appear in the CONFIG.SYS file.

Partitioning Schemes

Listed below are some sample partitioning schemes for two 40-megabyte (MB) hard disk drives and their resulting drive letter assignments:

- Drive 1:
  C: 20 MB primary MS-DOS partition
  E: 20 MB logical drive 1 in extended MS-DOS partition

  Drive 2:
  D: 20 MB primary MS-DOS partition
  F: 20 MB logical drive 1 in extended MS-DOS partition
- Drive 1:
  C: 20 MB primary MS-DOS partition
  D: 20 MB logical drive 1 in extended MS-DOS partition

  Drive 2:
  E: 20 MB logical drive 1 in extended MS-DOS partition
  F: 20 MB logical drive 2 in extended MS-DOS partition
- Drive 1:
  C: 10 MB primary MS-DOS partition
  E: 10 MB logical drive 1 in extended MS-DOS partition
  F: 10 MB logical drive 2 in extended MS-DOS partition
  G: 10 MB logical drive 3 in extended MS-DOS partition

  Drive 2:
  D: 10 MB primary MS-DOS partition
  H: 10 MB logical drive 1 in extended MS-DOS partition
  I: 10 MB logical drive 2 in extended MS-DOS partition
  J: 10 MB logical drive 3 in extended MS-DOS partition