

# ВОССТАНОВЛЕНИЕ ДАННЫХ НА NTFS-РАЗДЕЛАХ

## ЧАСТЬ 2

*Сегодня мы продолжим говорить о восстановлении данных, сосредоточив свое внимание на загрузочных секторах, таблицах разделов, динамических дисках и всех сопутствующих им служебных структурах.*



**КРИС КАСПЕРСКИ**

### Master boot record – базовые концепции

Первые жесткие диски были небольшого (даже по тем временам) размера и форматировались практически так же, как и дискеты, однако их объемы стремительно росли, и MS-DOS уже не могла их целиком адресовать. Для преодоления этого ограничения был введен механизм разделов (partitions), разбивающий один физический диск на несколько логических, каждый из которых имел свою собственную файловую систему и форматировался независимо от других. За счет чего это достигается?

В первом секторе физического диска (цилиндр 0/голов-

ка 0/сектор 1) хранится специальная структура данных – master boot record (главная загрузочная запись) или сокращенно MBR. Она состоит из двух основных частей – первичного загрузчика (master boot code) и таблицы разделов (partition table), описывающей схему разбиения и геометрию каждого из логических дисков. В конце сектора по смещению 1FE находится сигнатура 55h AAh, по которой BIOS определяет признак «загрузочности» сектора. Даже если вы не хотите дробить свой винчестер на части и форматироваете его как один диск, присутствие master boot record обязательно.



- таблица разделов
- раздел

Рисунок 1. Схематичное представление разбитого диска

При старте компьютера BIOS выбирает загрузочный винчестер (обычно Primary Master, но порядок загрузки в большинстве BIOS можно изменять, а самые продвинутые из них при удержании ESC во время прохождения post (процесса начального тестирования оборудования) даже выводят интерактивное меню), считывает первый сектор в память по адресу 0000h:7C00h, проверяет наличие сигнатуры 55h AAh в его конце, и если такая сигнатура действительно обнаруживается, передает управление на 0000h:7C00h. В противном случае анализируется следующее загрузочное устройство, а если таковое отсутствует, выдается ругательное сообщение.

Первичный загрузчик, получив управление, сканирует partition table (которая уже загружена в память!), находит активный раздел (Boot Indicator === 80h), извлекает номер стартового сектора раздела, также называемого boot-сектором, загружает его в память по адресу 0000h:7C00h (предварительно переместив свое тело в другое место, чтобы избежать затирания), убеждается в наличии сигнатуры 55h AAh, передавая управление по 0000h:7C00h, в противном случае выдается ругательное сообщение, и после нажатия на клавишу компьютер перезагружается. Некоторые загрузчики поддерживают несколько активных разделов, последовательно перебирая их один за другим, но это уже отсебятина разработчиков, выходящих за стандартные спецификации Microsoft, что, впрочем, никого не смущает.

Если первичный загрузчик поврежден, то BIOS не сможет запустить операционную систему с такого диска, однако при подключении его «вторым» (или загрузке с дискеты) все логические диски будут доступны. Как минимум они должны быть «видны», т.е. команды C:, D:, E: выполняются нормально, правда, работоспособность команды dir уже не гарантируется. Во-первых, для этого файловая система соответствующего раздела должна быть известна загруженной операционной системе и не повреждена, а во-вторых, должен быть цел boot-сектор (но об этом позже).

Partition Table, которую анализирует master boot code, а чуть позже – драйвер логических дисков операционной системы, состоит из четырех 10h-записей, расположенных по смещению 1BEh, 1CEh, 1DEh, 1EEh байт от начала диска соответственно. Каждая из них описывает свой логический раздел, задавая его стартовый и конечный сектора, записанные в CHS-формате (да! даже если диск работает в LBA-режиме, патриции все равно адресуются через CHS!). Поле относительного смещения раздела, отсчитываемое от начала таблицы разделов, является вспомогательным, и его избыточность очевидна. То же самое относится и к полю с общим количеством секторов на диске – как будто это нельзя вычислить на основе стартового и конечного секторов! Одни операционные системы и загрузчики игнорируют вспомогательные поля, другие же их активно используют, поэтому они должны соответствовать действительности.

Поле идентификатора диска содержит уникальную 32-разрядную последовательность, помогающую операционной системе отличить один смонтированный диск от другого и автоматически копирующую в следующий ключ реестра: HKLM\SYSTEM\MountedDevices. На самом деле Windows свободно обходится и без него, поэтому содержимое этого поля не критично.

Поле Boot ID содержит идентификатор файловой системы, установленной на разделе, который в случае NTFS равен 07h. За динамическими дисками согласно фирменной спецификации закреплён идентификатор 42h. На самом деле это справедливо лишь для тех из них, что получены путем обновления (update) обычного раздела до динамического. Сведения об остальных динамических дисках в таблице разделов не хранятся, а содержатся в последнем мегабайте физического диска в LDM-базе, и для стандартных дисковых менеджеров они не видны. При установке операционной системы семейства Windows 9x или UNIX на винчестер, содержащий динамические диски, они могут быть необратимо утрачены, поскольку согласно таблице разделов занятое ими пространство отмечено как свободное. Тем не менее загрузочный логический диск (независимо от того, динамический он или нет) в обязательном порядке должен присутствовать в partition table, иначе BIOS не сможет его загрузить.

Четырех записей partition table, обеспечивающих всего четыре логических диска, явно не хватало, но расширять таблицу разделов было уже некуда – последняя запись упиралась в конец сектора, а использовать следующий сектор разработчикам не хотелось, поскольку его активно использовали многие вирусы и нестандартные драйвера, к тому же это все равно не решало проблемы, а лишь оттягивало конец. Тогда инженеры нашли другое решение, предложив концепцию расширенных разделов (Extended partition). Если boot ID некоторого раздела равен 05h или 0Fh, он трактуется как «виртуальный физический диск»<sup>1</sup> со своей собственной partition table, расположенной в его начале, на которую и указывает стартовый сектор расширенного раздела. Короче говоря, таблица разделов получается вложенной, и уровень вложения ограничен разве что свободным местом жесткого диска и количеством стековой памяти загрузчика (при условии, что он использует рекурсивный алгоритм сканирования). Таблица разделов как бы размазывается вдоль винчестера. Большинство утилит резервирования сохраняют лишь первый сектор, чего явно недостаточно (впрочем, первый сектор гибнет намного чаще других, так что даже плохая политика резервирования лучше, чем совсем ничего).

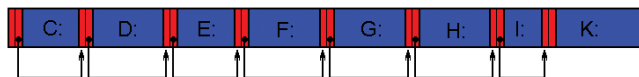


Рисунок 2. Расширенная таблица разделов

Штатные утилиты разбивки (FDISK.EXE, Disk Manager) в каждой таблице разделов создают один основной и один расширенный раздел. То есть при разбиении винчестера на четыре логических диска на нем образуются четыре partition table (см. листинг 4), хотя в данном случае можно

<sup>1</sup> Термин автора.

было бы обойтись и одной. Штатный загрузчик FDISK требует, чтобы активный раздел находился в первом секторе partition table, «благодаря» чему операционная система может грузиться только с диска C:. Нестандартные менеджеры, анализирующие всю цепочку разделов, позволяют загрузаться с любого из разделов. Самые честные из них создают в первой partition table еще один раздел (благо если диск был разбит FDISK, свободное место там всегда есть), назначают его активным и помещают в него свое тело. Другие же внедряются непосредственно в MBR, замещая первичный загрузчик, что создает очевидные проблемы совместимости.

Таблица 1. Формат MBR

Смещение	Размер	Назначение
0x000	перемен.	Код загрузчика
1x1BB	4h	Идентификатор диска
0x1BE	10h	Partition 1
0x1CE	10h	Partition 2
0x1DE	10h	Partition 3
0x1EE	10h	Partition 4
0x1FE	0x2	Признак таблицы разделов – сигнатура 55h Aah

Таблица 2. Формат partition

Смещение	Размер	Назначение
000 1BE 1CE 1DE 1EE	byte	Флаг активного загрузочного раздела. (Boot Indicator) 80h – загрузочный раздел, 00h – не загрузочный.
001 1BF 1CF 1DF 1EF		Стартовая головка раздела.
002 1C0 1D0 1E0 1F0	byte	Стартовый сектор раздела (биты 0 – 5). Старшие биты стартового цилиндра (биты 6-7).
003 1C1 1D1 1E1 1F1	byte	Младшие биты стартового цилиндра (биты 0-7).
004 1C2 1D2 1E2 1F2	byte	Идентификатор системы (Boot ID), см. таблицу. 4.
005 1C3 1D3 1E3 1F3	byte	Конечная головка раздела.
006 1C4 1D4 1E4 1F4	byte	Конечный сектор раздела (биты 0 – 5). Старшие биты конечного цилиндра (биты 6-7).
007 1C5 1D5 1E5 1F5		Младшие биты конечного цилиндра (биты 0-7).
008 1C6 1D6 1E6 1F6	dword	Смещение раздела относительно начала таблицы разделов в секторах.
00C 1CA 1DA 1EA 1FA	dword	Количество секторов раздела.

Таблица 3. Возможные значения Boot ID

Boot ID	Тип раздела
00h	Раздел свободен
0x01	FAT12 (менее чем 32.680 секторов в томе или 16 Мб)
0x04	FAT16 (32.680-65.535 секторов или 16-33 Мб)
0x05	Расширенный раздел (extended partition)
0x06	BIGDOS FAT16 раздел (33 Мб – 4 Гб)
0x07	NTFS-раздел
0x0B	FAT32-раздел
0x0C	FAT32-раздел с поддержкой расширенной BIOS INT 13h
0x0E	BIGDOS FAT16-раздел с поддержкой расширенной BIOS INT 13h
0x0F	Расширенный раздел с поддержкой расширенной BIOS int 13h
0x12	EISA-раздел
0x42	Динамический диск
0x86	legacy FT FAT16-раздел
0x87	legacy FT NTFS-раздел
0x8B	Legacy FT volume formatted with FAT32 *
0x8C	Legacy FT volume using BIOS INT 13h extensions formatted with FAT32

Если таблица разделов повреждена, логические диски, скорее всего, будут полностью недоступны – они не отображаются в «Моем Компьютере», не появились в панели «Driver» файлового менеджера FAR, а команда C: вызывает ошибку. Искажение таблицы разделов не приводит к немедленному изменению объема уже отформатированных томов (т.к. он хранится в boot-секторе и картах свободного пространства), но при последующем переформатировании произойдет затирание данных из соседнего раздела или же текущий раздел окажется усечен. Кстати говоря, если расширенный раздел указывает сам на себя или на один из предшествующих разделов в цепочке, все известные мне операционные системы наглухо зависнут еще на этапе загрузки, даже если диск подключен «вторым». Чтобы испра-

вить ситуацию, необходимо запустить редактор диска или другую утилиту, а для этого необходимо загрузить операционную систему! Существует несколько путей выхода из этой, казалось бы, неразрешимой проблемы. Самое простое – горячее подключение диска на ходу с последующей работой с ним через BIOS или порты ввода/вывода. Если ни диск, ни материнская плата не умеют (а для IDE-устройств подключение «на лету» представляется довольно жестким испытанием!), вы сможете запустить доктора и работать с диском на физическом уровне. Другой «хакерский» путь, – пропатчить MS-DOS, изменив сигнатуру 55h AAh на что-нибудь еще, тогда она не сможет распознать таблицу разделов и, стало быть, не станет ее анализировать. Как вариант можно записать в boot-сектор дискеты специально подготовленную программу, которая обнуляет MBR или искажает сигнатуру, расположенную в его конце. Просто загрузитесь с нее и все!

Листинг 1. Пример таблицы разделов, сформированный программой FDISK

```

Sector Inspector                                Copyright Microsoft Corporation 2003
=====
Target - \\.\PHYSICALDRIVE0
          1867 Cylinders
          255 Heads
           63 Sectors Per Track
          512 BytesPerSector
           12 MediaType

LBN 0 [C 0, H 0, S 1]
=====
Master Boot Record
=====
| B | FS TYPE | START | END | RELATIVE | TOTAL |
| F | (hex) | C H S | C H S |          |        |
=====
| * | 07 | 0 1 | 1 764 254 63 | 63 | 12289662 |
| | 0f | 765 0 | 1 1023 254 63 | 12289725 | 17687565 |
| | 00 | 0 0 | 0 0 0 0 | 0 | 0 |
| | 00 | 0 0 | 0 0 0 0 | 0 | 0 |
=====
LBN 12289725 [C 765, H 0, S 1]
=====
Extended Boot Record
=====
| B | FS TYPE | START | END | RELATIVE | TOTAL |
| F | (hex) | C H S | C H S |          |        |
=====
| | 07 | 765 1 | 1 1023 254 63 | 63 | 8193087 |
| | 05 | 1023 0 | 1 1023 254 63 | 8193150 | 4096575 |
| | 00 | 0 0 | 0 0 0 0 | 0 | 0 |
| | 00 | 0 0 | 0 0 0 0 | 0 | 0 |
=====
LBN 20482875 [C 1275, H 0, S 1]
=====
Extended Boot Record
=====
| B | FS TYPE | START | END | RELATIVE | TOTAL |
| F | (hex) | C H S | C H S |          |        |
=====
| | 07 | 1023 1 | 1 1023 254 63 | 63 | 4096512 |
| | 05 | 1023 0 | 1 1023 254 63 | 12289725 | 5397840 |
| | 00 | 0 0 | 0 0 0 0 | 0 | 0 |
| | 00 | 0 0 | 0 0 0 0 | 0 | 0 |
=====
LBN 24579450 [C 1530, H 0, S 1]
=====
Extended Boot Record
=====
| B | FS TYPE | START | END | RELATIVE | TOTAL |
| F | (hex) | C H S | C H S |          |        |
=====
| | 07 | 1023 1 | 1 1023 254 63 | 63 | 5397777 |
| | 00 | 0 0 | 0 0 0 0 | 0 | 0 |
| | 00 | 0 0 | 0 0 0 0 | 0 | 0 |
| | 00 | 0 0 | 0 0 0 0 | 0 | 0 |
=====

```

## Master boot record – техника восстановления

Существует множество утилит для автоматического восстановления master boot code и partition table (Get Data Back, Easy Recovery, Active Data Recovery Software и т. д.). До некоторого времени они вполне успешно справлялись со своей задачей, восстанавливая даже полностью уничтоженные таблицы разделов, однако с появлением емких вин-



тов, преодолевших барьер в 2 Гб с помощью всевозможных расширений, они стали часто путаться, и потому доверять им нельзя. Если не хотите потерять свои данные – восстанавливайте MBR самостоятельно (тем более что это достаточно простая операция, не требующая особой квалификации). Восстановление значительно упрощается, если в вашем распоряжении имеется копия таблицы разделов, снятая Sector Inspector или подобными ей утилитами. Однако чаще всего ее все-таки нет...

Если операционная система отказывается загружаться, а на экране появляется ругательство от BIOS типа «Disk Boot failure, Non-System disk or disk error... Press Enter to restart», это указывает на разрушение сигнатуры 55h AAh, обычно сопровождаемое смертью первичного загрузчика. Внимание! Очень важно отличать сообщение BIOS от сообщений первичного загрузчика и boot-сектора. Зайдите в BIOS Setup и отключите все загрузочные устройства, оставив активным только диск A: с вытасненной дискетой. А теперь перезагрузитесь и запомните, какое сообщение появится на экране. Это и будет ругательством BIOS.

Восстановить сигнатуру 55h AAh можно в любом дисковом редакторе. Когда будете это делать, убедитесь, что в начале диска присутствуют осмысленный master boot code (если вы испытываете затруднение с дизассемблированием в уме, воспользуйтесь IDA PRO или HIEW). Вы не умеете дизассемблировать? Тогда попробуйте оценить степень «нормальности» первичного загрузчика визуально (однако для этого опять-таки требуется опыт работы с кодом). В начале более или менее стандартного загрузчика расположено приблизительно 100h байт машинного кода, в котором обнаруживаются последовательности: 00 7C, 1B 7C, BE 07, CD 13, CD 18, CD 10, 55 AA, а затем идут характерные текстовые сообщения: «Invalid partition table, Error loading operating system, Missing operating system...» ну или подобные им. Если загрузчик поврежден, но сигнатура 55 AA цела, то попытка загрузки с такого диска обернется неизменным зависанием.

Восстановить «слетевший» или искореженный первичный загрузчик можно с помощью утилиты FDISK.EXE, запущенной с ключом /MBR, записывающей в главную загрузоч-

ную запись первого диска стандартный master boot code, или командой FIXMBR консоли аварийного восстановления в Windows 2000 (недокументированный ключ /CMBR, появившийся в MS-DOS 7.0, позволяет выбирать любой из подключенных дисков). Внимание! Если вы использовали нестандартный загрузчик (такой, например, как LILO), то после перезаписи MBR сможете загружаться только с основного раздела, а для запуска операционных систем из других разделов вам придется переустановить свой мультизагрузочный менеджер (вообще-то такой менеджер можно написать и самостоятельно, при наличии HIEW, а лучше использовать транслятор ассемблера – работа не займет и получаса).

Как уже говорилось, некоторые загрузчики изменяют схему трансляции адресов жесткого диска, и со штатным загрузчиком такой диск будет полностью неработоспособен. Попробуйте переустановить загрузчик с дистрибутивных дисков – быть может, это поможет. В противном случае ничего не остается, как писать свой собственный загрузчик, определять текущую геометрию диска и соответствующим образом транслировать секторные адреса. Это довольно сложная задача, требующая серьезной подготовки, и здесь ее лучше не обсуждать.

Если загрузчик говорит «Invalid partition table», это еще не значит, что таблица разделов повреждена, просто ни один из основных разделов не назначен активным. Такое случается при использовании нестандартных загрузчиков, загружающих операционную систему из расширенного раздела. После выполнения команды FDISK /MBR или при установке операционной системы, автоматически заменяющей первичный загрузчик своим собственным, он не обнаружит в пределах досягаемости ни одного активного раздела, и, естественно, разразится многоэтажным ругательством. Такое поведение, в частности, характерно для Windows 98. Для решения проблемы либо восстановите прежний загрузчик, либо установите операционную систему на первичный раздел и, запустив FDISK, сделайте его активным.

Загрузитесь с системной дискеты (другого винчестера, CD-диска) и посмотрите, видны ли ваши логические диски или нет. Если да, то смело переходите к следующему пунк-

## Проблема нулевой дорожки

Главная загрузочная запись жестко держит за собой первый сектор, и если он вдруг окажется разрушенным, работа с таким диском станет невозможной. До недавнего времени проблема решалась посекторным копированием винчестера с переносом данных на здоровый жесткий диск с идентичной геометрией с последующим восстановлением MBR.

Сейчас ситуация изменилась. Современные винчестеры поддерживают возможность принудительного замещения плохих секторов из резервного фонда (а некоторые делают это автоматически), поэтому проблема нулевой дорожки, преследующая нас еще со времен гибких дисков и 8-разрядных машин, наконец перестала существовать.

Механизм замещения секторов все еще не стандартизирован и осуществляется утилитами, предоставляемыми производителем конкретной модели винчестера. Чаще всего они распространяются бесплатно и могут быть свободно найдены в сети.

```

0x0000  eb 2e 49 50 41 52 54 20-63 6f 64 65 20 30 30 39  м.ИPART code 009
0x0010  20 2d 20 49 6f 6d 65 67-61 20 43 6f 72 70 6f 72  - Iomega Corpor
0x0020  61 74 69 6f 6e 20 2d 20-31 31 2f 32 33 2f 39 30  ation - 11/23/90
0x0030  fa fc 8c c8 8e d0 bc 00-7c 8e d8 8e c0 b9 00 02  .MMIO[0][0]O[0]
0x0040  bf 00 7e be 00 7c f3 a4-e9 00 02 fb bd 00 7e 8b  7. . . | едн. . . .Л
0x0050  fd be be 01 b9 04 00 80-3a 80 74 0b 83 c6 10 e2  nd . . . .A:At.T|P
0x0060  f6 8b b5 b2 01 eb 51 56-83 c6 10 49 e3 0b 80 3a  ылл. . . .mQVT|Iy.A:
0x0070  80 75 f5 8b b5 b0 01 eb-3f 5e 56 8a 12 8a 72 01  АуИл. . . .m^VKr.
0x0080  8a 44 02 8a 6a 03 bb 00-7c be 05 00 56 b8 01 02  KJ.Kj. . . .Vq.
0x0090  cd 13 73 0e 33 c0 cd 13-5e 4e 75 f0 8b b5 b4 01  ==!s.3|!^NuЕл. . .
0x00a0  eb 16 5e be fe 7d 81 3c-55 aa 74 06 8b b5 b6 01  м.^.^]БсUkt.л.л.
0x00b0  eb 06 5e c3 f5 e9 48 fd-e8 1b 00 8b b5 b8 01 e8  м.^.^нннн. .л.л.ш
0x00c0  14 00 b4 00 cd 16 33 c0-8e c0 26 c7 06 72 04 34  9. . . .-3|0.л.л.r.4
0x00d0  12 ea 00 ff 00 f0 03 f5-ac 3c 00 74 0b 56 b4 0e  :лЕ.Е.им.с.т.V.
0x00e0  bb 07 00 cd 10 5e eb f0-c3 49 6e 76 61 6c 69 64  9. . . .^ME|Invalid
0x00f0  20 70 61 72 74 69 74 69-6f 6e 20 74 61 62 6c 65  .partition table
0x0100  00 44 69 73 6b 20 69 73-20 6e f6 74 20 62 6f 6f  .Disk is not boo
0x0110  74 61 62 6c 65 00 45 72-72 6f 72 20 6c 6f 61 64  .table.Error load
0x0120  69 6e 67 20 6f 70 65 72-61 74 69 6e 67 20 73 79  .ing operating sy
0x0130  73 74 65 6d 00 4d 69 73-73 69 6e 67 20 6f 70 65  .stem.Missing ope
0x0140  72 61 74 69 6e 67 20 73-79 73 74 65 6d 00 0d 0a  .rating system...
0x0150  52 65 70 6c 61 63 65 20-61 6e 64 20 73 74 62 69  .Replace and stri
0x0160  6b 65 20 61 6e 79 20 6b-65 79 20 77 68 65 6e 20  .ke any key when
0x0170  72 65 61 64 79 0d 0a 00-00 00 00 00 00 00 00 00  .ready.....
0x0180  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .
0x0190  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .
0x01a0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 45 06  . . . . .E.
0x01b0  e9 00 01 01 16 01 35 01-4e 01 6a 72 a5 d5 00 00  .щ. . . .5.N.jre.
0x01c0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .
0x01d0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .
0x01e0  00 00 00 00 00 00 00 00-00 00 00 00 00 80 01  . . . . .A.
0x01f0  01 00 06 3f 20 5f 20 00-00 00 e0 ff 02 00 55 aa  . . . . .P. . . .

```

Листинг 2. Внешний вид типичного MBR-сектора

ту, в противном случае соберитесь с духом и приготовьтесь немного поработать руками и головой.

Восстановление основного раздела, созданного FDISK или Disk Manager, в большинстве случаев осуществляется элементарно, а остальные, как правило, восстанавливать и не требуется, поскольку именно MBR гибнет чаще всего, а расширенные патриции, рассредоточенные по диску, умирают разве что при явном удалении разделов средствами FDISK/Disk Manager.

Адрес стартового сектора первого логического диска всегда равен 0/1/1 (Cylinder/Head/Sector), относительный (Relative) сектор – количеству головок жесткого диска, уменьшенных на единицу (сведения о геометрии диска можно почерпнуть из любого дискового редактора, в том числе и Sector Inspector). Конечный сектор определить несколько сложнее. Если загрузочный сектор цел (см. «Загрузочный сектор – техника восстановления»), то узнать количество секторов в разделе патриции (total sectors) можно и из поля BootRecord.NumberSectors, увеличив его значение на единицу. Тогда конечный цилиндр будет равен:

```
LastCyl := TotalSectors / (Heads * SecPerTrack)
```

где Heads – количество головок на физическом диске, а SecPerTrack – количество секторов на трек.

Конечная головка равна:

```
LastHead := (Total Sector - 1  
              (LastCyl * Heads * SecPerTrack)) / SecPerTrack
```

а конечный сектор равен:

```
LastSec := (Total Sector - 1  
            (LastCyl * Heads * SecPerTrack)) % SecPerTrack
```

Пропишите полученные значения в MBR и посмотрите, не находится ли за вычисленным концом раздела следующий раздел? Это должна быть либо расширенная таблица

разделов, либо boot-сектор. Если это так, создайте еще одну запись в partition table, заполнив его соответствующим образом.

А если boot-сектор отсутствует и не может быть восстановлен, реально ли восстановить таблицу разделов или нет? Да, можно. Необходимо лишь найти boot или partition следующих разделов, в чем вам поможет контекстный поиск. Ищите сектора, содержащие сигнатуру 55h AAh в конце. Отличить boot от partition очень просто (в boot-секторе по смещению два байта от его начала расположен идентификатор производителя (NTFS, MSWIN4.1 и т. д.). Логично, что размер текущего раздела на один сектор меньше, а зная размер и геометрию диска, можно рассчитать и конечный цилиндр/головку/сектор.

Только учтите, что Windows хранит копию boot-сектора, которая в зависимости от версии может быть расположена либо в середине раздела, либо в его конце. Другие копии могут находиться в архивных файлах и файле подкачки. Кстати говоря, посмотрите, не содержится ли среди них ничего удобоваримого. Как отличить копию сектора от оригинала? Если это подлинник, вслед за ним пойдут служебные структуры файловой системы (в частности, для NTFS таблица MFT, каждая запись которой начинается с легко узнаваемой строки FILE\*). Собственно говоря, поскольку служебные структуры файловой системы обычно располагаются на более или менее предсказуемом смещении относительно начала раздела, то отталкиваясь от их «географического» расположения, мы можем установить размеры каждого из логических дисков, даже если все boot/partition уничтожены.

Что произойдет, если границы разделов окажутся определенными неверно? Если мы переборщим, увеличив размер раздела сверх необходимого, все будет нормально работать, поскольку карта свободного пространства хранится в специальной структуре (у NTFS это файл \$bitmap, а у FAT13/32 – непосредственно сама FAT-таблица) и «зап-

## Типы динамических дисков, поддерживаемые Windows 2000

- Простые (simple): практически ничем не отличаются от обычных разделов, за исключением того, что при переразбиении диска отпадает необходимость в перезагрузке. Базовый тип для всех остальных динамических дисков.  
Избыточность: нет.  
Эффективность: низкая.
- Составные (spanned): состоят из одного или нескольких simple-дисков, находящихся в различных разделах или даже устройствах, представленные как один логический диск. Данные на simple-диски пишутся последовательно (классический линейный RAID).  
Избыточность: нет.  
Эффективность: низкая.
- Чередующиеся (stripped): то же самое, что и spanned, но данные записываются параллельно на все simple-диски при условии, что они расположены на различных каналах IDE-контроллера. Это значительно увеличивает скорость обмена данными, короче говоря, классический RAID уровня 0.

Избыточность: нет.

Эффективность: средняя.

- Зеркальные (mirrored): два simple-диска, расположенные на разных устройствах, данные дублируются на оба носителя (RAID уровня 1).

Избыточность: средняя.

Эффективность: средняя.

- Чередующиеся с контролем четности (stripped with parity): соответствует массиву RAID уровня 5. Состоит из трёх или более дисков. Представляет собой striped volume с контролем ошибок. Данные пишутся на два диска, в два блока, а на третий диск и в третий блок записывается ECC, код коррекции ошибок, с помощью которого по информации любого из блоков можно восстановить содержимое второго блока.

Избыточность: высокая.

Эффективность: высокая.

- Зеркальный с чередованием (mirrored striped): соответствует массиву RAID 1+0.

Избыточность: средняя.

Эффективность: высокая.

редельные» сектора будут добавлены только после переформатирования раздела. Если все, что нам нужно, – это скопировать данные с восстанавливаемого диска на другой носитель, то возиться с подгонкой параметров partition table не нужно! Распахните ее на весь физический диск и дело с концом!

Естественно, такой способ восстановления подходит только для первого раздела диска, а для всех последующих нам потребуется определить стартовый сектор. Это определение должно быть очень точным, поскольку все структуры файловой системы адресуются от начала логического диска и ошибка в один-единственный сектор делает весь этот тонкий механизм полностью неработоспособным. К счастью, некоторые из структур ссылаются сами на себя, давая нам ключ к разгадке. В частности, файлы \$mft/\$mftmifф содержат номер своего первого кластера. Стоит нам найти первую запись FILE\*, как мы узнаем, на каком именно секторе мы сейчас находимся (конечно, при условии, что сумеем определить количество секторов на кластер, но это уже другая тема – см. раздел «Загрузочный сектор – базовые концепции»).

## Динамические диски

Динамические диски, впервые появившиеся в Windows 2000, – это все тот же программный RAID, призванный преодолеть ограничения стандартных механизмов разбиения с учетом ошибок своего прямого предшественника программного RAID Windows NT, хранящего конфигурационную информацию в системном реестре, что, во-первых, препятствовало его перемещению с машины на машину, а во-вторых, делало очень уязвимым к порче реестра.

По умолчанию Windows создает базовые диски (см. расшифровку терминов в таблице 5), но всякий базовый диск в любой момент времени может быть обновлен до динамического (это даже не потребует перезагрузки). Динамические диски не пользуются таблицей разделов, а потому и не имеют проблем, связанных с ограничением CHS-разрядности и позволяют создавать тома практически неограниченного размера. Однако динамические диски, созданные путем обновления основных разделов, все-таки остаются в partition table, при этом их Boot ID меняется на 42h. Если эта информация окажется удалена, система откажется подключать такой динамический диск. Кстати говоря, Windows может быть установлена только на обновленный динамический диск, поскольку BIOS может загружать систему лишь с тех разделов, которые перечислены в partition table, а динамические диски, созданные «на лету», в нее как раз и не попадают.

Схема разбиения динамических дисков содержится в Базе Менеджера Логических Дисков – Logical Disk Manager Database или сокращенно LDM. Это протоколируемая (journalled) база данных, поддерживающая транзакции и устойчивая к сбоям. Если в процессе манипуляции с томами неожиданно исчезнет питание, при последующем включении компьютера будет выполнен откат в предыдущее состояние. При переносе винчестера, содержащего один или несколько динамических дисков, на другую систему, они автоматически распознаются и монтируются, как обыкновенные диски.

LMD-база хранится в последнем мегабайте жесткого диска, а для дисков, полученных путем обновления базового раздела до динамического, – в последнем мегабайте этого самого раздела, идентификатор системы Boot ID соответствующей записи в Partition Table принимает значение 42h. Так происходит потому, что при стандартном разбиении винчестера в его конце просто не остается свободного места и операционной системе приходится сохранять эту информацию непосредственно в самом обновляемом диске (естественно, для этого в нем должен быть свободен по меньшей мере 1 Мб).

Сразу же за таблицей разделов по адресу 0/0/2 расположен приватный заголовок PRIVHEAD, содержащий в себе ссылки на основные структуры LDM (см. рис. 3). Если PRIVHEAD погибнет, Windows не сможет обнаружить и смонтировать динамические диски. А гибнет он удручающе часто. Подавляющее большинство загрузочных вирусов и дисковых менеджеров считают сектор 0/0/2 свободным и используют его для хранения своего тела, необратимо затирая прежнее содержимое. Осознавая значимость PRIVHEAD, разработчики из Microsoft сохранили его аж в двух копиях, одна из которых хранится в хвосте LDM, а другая – в последнем секторе физического диска. Благодаря такой избыточности, PRIVHEAD практически никогда не приходится восстанавливать вручную, но если это все-таки потребуется сделать, обратитесь к проекту LINUX-NTFS за подробным описанием его структуры (<http://linux-ntfs.sourceforge.net>), здесь же оно по соображениям экономии места не приводится.

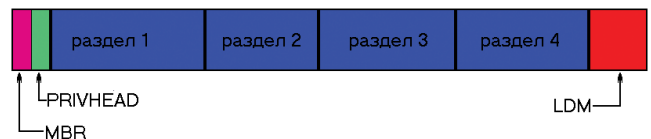


Рисунок 3. LDM-база и ее дислокация

Внутреннее устройство LDM-базы недокументировано и буквально пышет мощностью и сложностью. Наверху иерархии расположено оглавление базы – структура TOCBLOCK (Table Of Content Block), состоящая из двух секций config и log (вероятно, в будущем их список будет расширен). Секция config содержит информацию о текущем разбиении динамических дисков и сведения о томах, а log хранит журнал изменений схемы разбивки. Это очень мощное средство в борьбе с энтропией! Если удалить один или несколько динамических разделов, информация о старом разбиении сохранится в журнале и утраченные тома могут быть с легкостью восстановлены! Будучи очень важной структурой, оглавление диска защищено от случайного разрушения тремя копиями, одна из которых вплотную примыкает к настоящему TOCBLOCK, расположенному в начале LDM-базы, а две другие находятся в конце диска, между копиями PRIVHEAD.

Внутренняя секция config состоит из заголовка (VMDB) и одного или нескольких VBLKs – специальных 128-байтовых структур данных, каждая из которых описывает соответствующий ей том, контейнер, раздел, диск или группу дисков. VMDB-заголовок не имеет копии и нигде не дублируется, однако все его изменения протоколируются в журнале (KLOG) и потому могут быть восстановлены.



Строение VMDb и VBLs подробно документировано «LMD Documentation», находящейся на сайте LINUX-NTFS и потому описывать его здесь нет никакой необходимости (оно слишком громоздко, к тому же крайне маловероятно, что кому-то потребуется восстанавливать секцию config руками).

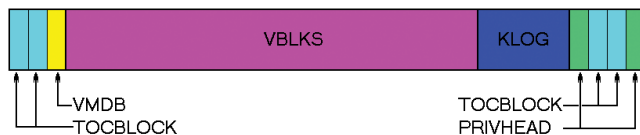


Рисунок 4. Внутри LDM

Для просмотра LDM-базы и архивирования ее содержимого можно воспользоваться утилитой LDM-dump Марка Руссиновича, бесплатную копию которой можно скачать с его сайта (<http://www.sysinternals.com/files/ldmdump.zip>). Как вариант – можно зарезервировать последний мегабайт физического диска и последний мегабайт всех патриций, чей Boot ID равен 42h любым подходящим дисковым редактором (например, Sector Inspector) и сохранить эту информацию на надежном носителе (Zip, CD-R/RW), не забывая также зарезервировать и TOCBLOCK.

При восстановлении удаленных динамических дисков следует учитывать, что, во-первых, журнал изменений на интерфейсном уровне недоступен и выполнить откат легальными средствами операционной системы невозможно. Во-вторых, boot-сектор удаляемых дисков автоматически очищается и восстанавливать его приходится вручную, о чем мы чуть позже и поговорим.

Если размер и тип удаленного динамического диска вам известен (на NTFS-дисках его можно извлечь из копии boot-сектора), просто зайдите в «Менеджер Управления Диска» и воссоздайте его заново, от предложения отформатировать раздел любезно откажитесь и восстановите очищенный boot-сектор по методике, описанной ниже.

Как видно, Microsoft тщательно позаботилась о своих пользователях и занималась проектированием структуры динамических дисков на свежую голову, что для нее вообще говоря нехарактерно.

Таблица 4. Терминологическое соответствие динамических и обычных дисков

Базовые (Basic) разделы	Динамические (Dynamic) разделы
Основной раздел/Primary partition	Простой том/Simple volume
Системный и загрузочный раздел/System and boot partitions	Системный и загрузочный том/System and boot volumes
Активный раздел/Active partition	Активный том/Active volume
Расширенный раздел/Extended partition	Том и свободное пространство/Volume and unallocated space
Логический диск/Logical drive	Простой том/Simple volume
Набор томов/Volume set	Составной том/Spanned volume
Чередующийся набор/Stripe set	Чередующийся том/Stripe set

## Загрузочный сектор – базовые концепции

Первый сектор логического диска носит название загрузочного (boot). Он содержит самозагрузочный код (bootstrap code) и важнейшие сведения о геометрии диска, без которых раздел просто не будет смонтирован! Структура boot-сектора определяется архитектурными особенностями конкретной файловой системы, в частности NTFS boot sector выглядит так:

Таблица 5. Строение NTFS boot-сектора

Смещение	Размер	Назначение
0x00	3 bytes	Инструкция перехода
0x03	8 байт	OEM ID – идентификатор
0x0B	25 bytes	BPB
0x24	48 bytes	Extended BPB
0x54	426 bytes	Bootstrap Code
0x01FE	WORD	55 AA

В начале всякого сектора расположена трехбайтовая машинная команда перехода на bootstrap code (обычно EB 52 90, хотя возможны и вариации). Так происходит потому, что при загрузке boot-сектора в память управление передается на его первый байт, а bootstrap код по туманному историческому соображению был отодвинут в конец сектора (для NTFS верхняя граница составляет 54h байт), вот и приходится прыгать блохой!

С третьего по одиннадцатый байты (считая от нуля) хранится идентификатор производителя, определяющий тип и версию используемой файловой системы (например, «MSDOS5.0» для FAT16, «MSWIN4.0»/«MSWIN4.1» для FAT32 и «NTFS» для NTFS). Если это поле окажется искажено, драйвер не сможет смонтировать диск и даже может посчитать его не отформатированным! (примечание: с FAT-дисками Windows 2000 будет работать даже с запарченным OEM ID, чего не скажешь про NTFS).

Следом за идентификатором расположен 25-байтовый блок параметров BIOS (BIOS Parameter Block или сокращенно BPB), хранящий сведения о геометрии диска (число цилиндров, головок, секторов, размер сектора, количество секторов в кластере и т. д.). Если эта информация окажется утеряна или искажена, нормальное функционирование драйвера файловой системы станет невозможным. Причем если данное число цилиндров/головок/секторов дублирует информацию, содержащуюся в MBR, а при ее утере элементарно восстанавливается описанным выше способом, то размер кластера определить не так-то просто! Позже мы обсудим этот вопрос более подробно, пока же ограничимся следующей табличкой, сообщающей размер кластера NTFS-томов, выбираемой штатной утилитой форматирования по умолчанию (см. таблицу 7).

При выборе размера кластера вручную, Windows 2000 поддерживает следующий модельный ряд: 1, 2, 4, 8, 16, 32, 64 и 128 секторов. Чем больше размер кластера, тем меньше фрагментация и выше предельно адресуемый дисковый объем, но вместе с тем и выше потери от грануляции. Впрочем, ручное задание размеров кластера встречается достаточно редко.

Таблица 6. Размер кластера, выбираемый Windows 2000 по умолчанию

Размер диска	Размер кластера
< 512 Мб	1 сектор
< 1 Гб	2 сектора
< 2 Гб	4 сектора
> 2 Гб	8 секторов

К блоку параметров BIOS вплотную примыкает его продолжение – extended BPB, хранящий номер первого кластера MFT, ее размер в кластерах, номер кластера с зеркалом MFT и некоторую другую информацию. В отличие от FAT16/32, MFT может располагаться в любом месте диска (для борьбы с BAD-секторами это актуально). При нормальном развитии событий MFT располагается практически в самом начале диска (где-то в районе 4 кластера) и если только она не была перемещена, ее легко найти глобаль-

ным поиском (строка «FILE\*» по смещению 0 от начала сектора). При разрушении или некорректном заполнении extend PBP драйвер файловой системы отказывается монтировать раздел, объявляя его неотформатированным.

Следом за extend PBP идет Bootstrap Code, который ищет на диске операционный загрузчик (у Windows NT это ntldr), загружает его в память и передает ему управление. Если Bootstrap Code отсутствует, загрузка операционной системы становится невозможной, однако, при подключении восстанавливаемого диска вторым раздел должен быть прекрасно виден. Порча Bootstrap Code вызывает перезагрузку компьютера или его зависание.

И завершает boot-сектор уже известная нам сигнатура 55h AAh, без которой он ни за что не будет признан загрузочным.

Таблица 7. Значение полей NTFS boot-сектора

Смещение	Размер	Назначение
0x00	3 bytes	Инструкция перехода
0x03	8 байт	OEM ID
0x0B	WORD	Байт на сектор (для жестких дисков всегда 512)
0x0D	BYTE	Секторов на кластер
0x0E	WORD	Количество зарезервированных секторов, равно 0
0x10	3 BYTES	Не используется NTFS и всегда должно быть равно 0
0x13	WORD	Не используется NTFS и всегда должно быть равно 0
0x15	BYTE	Медиа-дескриптор для жестких дисков, всегда равен 0xF8
0x16	WORD	Не используется NTFS и всегда должно быть равно 0
0x18	WORD	Количество секторов в треке
0x1A	WORD	Количество головок
0x1C	DWORD	Количество скрытых секторов
0x20	DWORD	Не используется NTFS и всегда должно быть равно 0
0x24	DWORD	Не используется NTFS и всегда должно быть равно 0
0x28	8 байт	Общее количество секторов (total sector)
0x30	8 байт	Логический номер кластера, с которого начинается MTF
0x38	8 байт	Логический номер кластера, с которого начинается зеркало MTF
0x40	DWORD	Количество кластеров на сегмент (File Record Segment)
0x44	DWORD	Количество кластеров на блок индексов (index block)
0x48	8 байт	Серийный номер тома
0x50	DWORD	Контрольная сумма (0 – не подсчитывать).
0x54	426 bytes	Bootstrap Code
0x01FE	WORD	55 AA

## Загрузочный сектор – техника восстановления

Осознавая значимость загрузочного сектора, Windows NT при форматировании диска создает его зеркальную копию (правда, только на NTFS-разделах). Windows NT 4.0 располагает ее посередине логического диска, а Windows 2000 – в последнем секторе раздела. Если partition table цела, просто перейдите в начало следующего раздела и отступите на сектор назад (Windows 2000) или поделите количество секторов логического диска пополам (с округлением в нижнюю сторону) и скажите редактору диска «GO» (Windows NT 4.0).

Если же таблица разделов разрушена, найти копию сектора можно глобальным поиском (ищите строку «NTFS» по смещению 3 от начала сектора). Поскольку положение копии фиксировано и отсчитывается от начала логического диска, мы можем с абсолютной уверенностью определить границы раздела.

Допустим, копия boot найдена в секторе 1 289 724, а поле NumberSectors содержит значение 12 289 661. Тогда конечный сектор раздела равен 1 289 724, а стартовый: 1289724 – 12289661 == 63. Поскольку загрузочный сектор расположен на расстоянии одной головки от partition table,

что соответствует SectorPerTrack-секторам, мы можем восстановить и ее.

В отсутствии копий, boot-сектор приходится реконструировать вручную. Это легко. В поле идентификатора производителя заносится строка «NTFS» (без кавычек, но с четырьмя пробелами на конце). Количество секторов в треке и число головок заполняются исходя из текущей геометрии диска. Количество скрытых секторов (т.е. секторов, расположенных между началом раздела и boot-сектором) равно числу головок. Общее количество секторов в разделе вычисляется на основании его размера (если точный раздел не известен, берите значение с запасом).

Количество секторов в кластере определить сложнее (особенно, если диск отформатирован со значением, отличным от принятого по умолчанию). Но ситуация вовсе не безнадежна. Последовательно сканируя файловые записи в MFT, найдите файл с заранее известной сигнатурой. Пусть для определенности это будет NTOSKRNL.EXE. Откройте его аутентичную копию в HEX-редакторе, найдите уникальную последовательность, гарантировано не встречающуюся ни в каких других файлах и расположенную в пределах первых 512 байт от его начала, после чего найдите ее глобальным поиском на диске. Начальный номер кластера вам известен (он содержится в MFT), логический номер сектора тоже (его нашел дисковый редактор). Теперь остается лишь соотнести эти две величины между собой. Естественно, если дисковый редактор найдет удаленную копию ntoskrnl.exe (или на диске будут присутствовать несколько файлов ntoskrnl.exe), данный метод даст осечку, поэтому полученный результат необходимо уточнить на других файлах.

Логический номер первого кластера MFT равен первому кластеру, в начале которого встретилась строка «FILE\*» (конечно, при том условии, что MFT не был перемещен). Штатно Windows выделяет под MFT 10% от емкости раздела, помещая зеркало в середину. Кроме того, ссылка на зеркало присутствует и в самом MFT. Если же он разрушен, переместитесь в середину диска, немного отступите назад и повторите глобальный поиск строки «FILE\*» (только смотрите, не вылетите в соседний раздел!). Первое же найденное вхождение с высокой степенью вероятности и будет зеркалом.

Количество кластеров на сегмент обычно равно F6h, а на блоке индексов – 01h. Других значений мне встречать не доводилось. Серийный номер тома может быть также любым – он ни на что не влияет.

Для восстановления отсутствующего (искаженного) bootstrap code загрузите консоль восстановления и отдайте команду FIXBOOT.

## Заключение

Как видно, в восстановлении данных нет ничего мифического и для устранения большинства типов разрушений не требуется никакой квалификации. Если же некоторые моменты вам не понятны, перечитайте эту статью с дисковым редактором в руках. До сих пор мы говорили о достаточно простых и хорошо известных вещах. Теперь, как следует расквашившись и освоившись с основными понятиями, мы можем отправляться в самые дебри NTFS, восстановлению структур которой посвящена следующая статья этого цикла.