unformat для NTFS

крис касперски

- Я у вас тут винчестер недавно купил. Так вот, он сдох!
- Гарантия какая?
- Пожизненная.
- Раз сдох, значит, гарантия кончилась.
разговор продавца с покупателем

случилось самое страшное: вы потеряли весь NTFS-раздел целиком. случайно отформатировали или пережили разрушительный дисковый сбой. где-то там остались миллиарды байт бесценных данных теперь уже недоступных операционной системе. как вернуть информацию к жизни? в этой статье автор делится советами ручного и автоматического восстановления.

введение

До сих пор мы рассматривали лишь незначительные дисковые сбои и легкие разрушения данных типа ошибочно удаленных файлов. Теперь настал черед тяжелых повреждений при которых прежнее содержимое тома становится недоступно операционной системе. Причиной может быть и непредумышленное форматирование, и искажение главной файловой таблицы, и… Но вы не падайте духом! Из любых переделок NTFS выходит с минимальными потерями и во всех этих случаях возможно полное, стопроцентное восстановление данных, если к делу подойти с головой и прямыми руками.

Проще всего начать с форматирования. Для экспериментов нам потребуется format.com, входящий в комплект штатной поставки Windows NT/2000/XP и дисковый раздел, не содержащий ничего полезного. Хорошо бы обзавестись виртуальной машиной — **Virtual PC** или **VMWare**, эмулирующей жесткий диск и ускоряющей процедуру форматирования в сотни (!) раз.

"Живой" винчестер лучше не трогать (во всяком случае до тех пор, пока вы не научитесь его восстанавливать). Если нет виртуальной машины (до сих пор не поставили Осла? не пользуетесь IRC?), попробуйте приобрести ZIP-привод (который, кстати говоря, намного надежнее оптических накопителей) и форматируйте дискеты под NTFS — благо штатный форматер это позволяет. С гибкими дисками дело обстоит сложнее. По мнению Microsoft их емкости недостаточно для размещения всех структур данных, хотя, простейший расчет разбивает это утверждение в пух и прах, что утилита NTFSflp от Марка Руссиновича (mark@sysinternals.com), собственно говоря, и демонстрирует. Статья "NTFS Support for Floppy Disks" (http://www.sysinternals.com/ntw2k/freeware/ntfsfloppy.shtml) подробно описает как обхитрить систему, заставив ее отформатировать гибкий диск под NTFS (для этого вам потребуется soft-ice).

Рисунок 1 форматирование виртуального диска в среде VMWare

что происходит при форматировании

Форматирование диска — это сложная многостадийная операция, намного более сложная и намного более многостадийная, чем это может показаться на первый взгляд. Кто писал свой собственный форматер дискет (а в конце восьмидесятых—начале девяностых его писали практически все) — тот поймет. Свои исследования мы начнем с изучения NTFS-тома, отформатированного под NTFS (техника восстановления NTFS-томов, отформатированных под FAT16/32 дана в одноименной врезке).

При выполнении команды "`format X: /U /FS:NTFS`" в файловой системе диска X: происходят следующие изменения (форматирование диска GUI-утилитой, вызываемой из контекстного меню "проводника" осуществляется по аналогичной схеме):

- формируется boot-сектор в формате NTFS;
- генерируется новый серийный номер диска и записывается в boot-сектор по смещению 48h байт от его начала;

- рассчитывается новая контрольная сумма boot-сектора и записывается по смещению 50h от его начала (подробнее см. первую статью этого цикла: "загрузочный сектор — базовые концепции");
- создается новый $MFT-файл, содержащий сведения обо всех файлах на диске, и как правило, размещаемый поверх старого $MFT-файла; исключения здесь крайне редки — разве что прежний $MFT был заблаговременно перемещен дефрагментатором, или при форматировании назначен новый размер кластера. Во всех остальных случаях первые ~24 файловых записи (FILE Record) мрут безвозвратно. В них находится — непосредственно сам $MFT, $MFTMirr, корневой каталог, /$LogFile – файл транзакций, /$BITMAP – карта свободного пространства, /$Secure – дескрипторы безопасности и другие служебные файлы;
- инициируется $MFT:$DATA — назначается новая длина ($MFT:$30.AllocatedSize, $MFT:$30.RealSize, $MFT:$80.AllocatedSize, $MFT:$80.RealSize, $MFT:$80.CompressionSize, $MFT:$80.InitializedSize, $MFT:$80.LastVCN), дата/время создания/последней модификации ($MFT:$10.FileCreationTime, $MFT:$10.FileAlertedTime, $MFT:$10.FileReadTime, $MFT:$30.FileCreationTime, $MFT:$30.FileAlertedTime, $MFT:$30.MFTChangeTime, $MFT:$30.FileReadTime) и, самое главное, создается новый список отрезков (data-runs), необратимо затирающий старый, а это значит, что собирать фрагментированный $MFT нам придется по частям;
- создается новый /$MFT:$BITMAP, отвечающий за занятость файловых записей в MFT — все старые записи помечаются как свободные, однако, их фактического удаления при этом не происходит (поле FileRecord.flags остается нетронутым), благодаря чему процедура восстановления заметно упрощается. Чаще всего $MFT:$BITMAP располагается на том же самом месте, что и старый (т. е. между boot-сектором и MFT), забивая прежнее содержимое нулями, однако, с помощью утилиты chkdsk его можно восстановить;
- создается новый /$BITMAP-файл, отвечающий за распределение дискового пространства (свободные и занятые кластера), опять-таки затирающий старый, но восстанавливаемый chkdsk'ом;
- создается новый файл журнала транзакций — /$LogFile, в структуру которого мы углубляться все равно не будем, хотя в NTFS LINUX Project она описана достаточно подробно, но восстанавливать транзакции — это уж слишком;
- в заголовок файловой записи $MFT заноситься новый LogFile Sequence Number или сокращенного LSN;
- $MFT назначается новый номер последовательности обновления (Update Sequence Number);
- создается новое зеркало $MFTMirr, необратимое затирающее старое (в текущих системах оно расположено посередине NTFS-раздела), в результате чего возникает резонный вопрос — какой прок от зеркала, которое ничего не отражает?;
- создаются новые /$Volume, /$AttrDef и другие служебные файлы, играющие сугубо вспомогательную роль и легко восстанавливаемые chkdsk'ом (хотя и /$Volume и присутствует в зеркальной копии MFT, его ценность явно преувеличена);
- осуществляется проверка целостности поверхности и все обнаруженные плохие кластеры заносятся в файл /$BadClus;
- формируется новый корневой каталог;
- если до форматирования тома на нем присутствовал /System Volume Information-файл, то он обновляется, в противном случае новый /System Volume Information создается только после перезагрузки;

На самом деле, процесс форматирования протекает намного сложнее, но не будем в него углубляться — мы же не форматер пишем! Интересующееся могут взять в руки IDA Pro и расковырять format.com самостоятельно. Подсказка — format.com содержит лишь высокоуровневую надстройку, опирающуюся на библиотеки ifsutil.dll, untfs.dll и… непосредственно сам драйвер файловой системы. Так что дизассемблировать придется много. Чтобы упросить себе работу, можно пронаблюдать за процессом форматирования с помощью шпионских средств — утилит Марка Руссиновича FileMon и DiskMon, бесплатные копии которых можно скачать с www.sysinternals.com. Так же не забывайте о точках останова на основные native-API функции такие как NtFsControlFile, NtDeviceIoControlFile и т. д. Да будет soft-ice вам в помощь!

Рисунок 2 исследования процесса форматирования с помощью шпионских средств

автоматическое восстановление отформатированного диска

Форматирование не уничтожает файловые записи пользовательских файлов и они могут быть полностью восстановлены. Спасением данных занимается множество утилит (R-Studio, EasyRecovery, GetDataBack и т. д.), однако, прямых наследников unformat'a среди них как-то не наблюдается. Unformat восстанавливал весь том целиком, а эти всего лишь "вытягивают" отдельные уцелевшие файлы/каталоги, переписывая их на новый носитель. Да где же нам его взять?! Запись на оптические накопители отпадает сразу — во-первых, для сохранения 80 — 120 Гбайтного жесткого диска в лучшем случае потребуется грузовик (на чем еще вы перевезете такое количество болванок?), во-вторых, непосредственная запись CD-R/RW не всегда возможна, ведь при крахе системы восстанавливающие утилиты приходится загружать с CD-ROM, а в большинстве компьютеров установлен только один оптический привод, и, в-третьих, ни одна известная мне утилита не позволяет "разрезать" большие файлы на несколько маленьких. Можно, конечно, перегнать данные по локальной сети (а она есть?) или установить дополнительный винчестер (корпус компьютера не опечатан, имеется свободные каналы контроллера и лишняя наличность в кармане). Но не слишком ли это хлопотно? Тем не менее, для "вытягивая" пары сотен особо ценных файлов такая методика вполне подходит.

Продемонстрируем технику автоматического восстановления данных на примере утилиты R-Studio от компании R-TT Inc (www.r-tt.com). Это довольно мощный и в тоже время простой в управлении инструмент, на который можно положиться. После запуска утилиты мы сразу попадаем в окно "Drive View", где перечислены все физические устройства и логические разделы. Находим среди них "свой" и, нажав правую клавишу мыши, говорим "Scan".

Нас запрашивают: начальный сектор для сканирования (start), по умолчанию равный 0 — оставляем его без изменений. Размер сканируемой области (size), по умолчанию развертывается на весь раздел. Это гарантирует, что сканер обнаружит все уцелевшие файловые записи, хотя сам поиск займет значительное время. Можно ли ускорить этот процесс? Давайте возьмем ручку и подсчитаем. Допустим, восстанавливаемый раздел содержит сто тысяч файлов. Типичный размер файловой записи составляет 1 Кбайт. При условии, что $MFT не фрагментирован, достаточно просканировать всего ~100 Мбайт от начала раздела. Если эта величина не превышает 10% полной емкости тома (размер пространства зарезервированного под MFT) и диск никогда не заполнялся более чем на 90%, то, скорее всего, все так и есть. В противном случае, $MFT наверняка фрагментирован и живописно разбросан по всему диску. Впрочем, в случае ошибки мы ничем не рискуем. Вводим N Кбайт, где N – предполагаемое кол-во файлов (каталог также считается файлом) и выполняем сканирование — если один или несколько файлов останутся необнаруженными, возвращаемся к настройкам по умолчанию и повторяем процедуру сканирования вновь (если количество имеющихся файлов заранее неизвестно, вводим 10% от емкости тома). В поле File System выбираем файловую систему NTFS, сбрасывая галочки напротив двух остальных (FAT и Ext2FS), т. к. они нам ни к чему, и нажимам "Scan".

Рисунок 3 R-Studio осуществляет поиск уцелевших файловых записей

В процессе сканирования будут найдены все уцелевшие файлы (как удаленные, так и нет) и восстановлена структура директорий по корневой каталог включительно. Постойте! Как же так?! Ведь при форматировании он был уничтожен и сформирован заново! Хе-хе. Файловую систему NTFS можно уничтожить только динамитом. Как уже отмечалось, в отличии от FAT, в NTFS каталоги являются лишь вспомогательной структурой данных, проиндексированной для ускорения отображения их содержимого. Всякая файловая запись независимо от своего происхождения, содержит ссылку на материнский каталог, представляющую собой номер записи в MFT. А запись корневого каталога всегда располагается по одному и тому же месту!

Удаленные файловые записи могут ссылаться на уже уничтоженные каталоги. R-Studio складывает их в $$$FolderXXX, где XXX – порядковый номер директории. Иерархия подкаталогов в большинстве случаев успешно восстанавливается.

Просмотр виртуального дерева обнаруженных файлов осуществляется нажатием кнопки <F5> или через одноименный пункт контекстного меню. Выбрав файл (или даже целый каталог с кучей подкаталогов) жмем <F2> или залезаем в предварительный просмотр/редактирование (пункт edit/view контекстного меню). Это достаточно мощный

инструмент, отображающий внутреннее содержимое восстанавливаемого файла со всеми его атрибутами, списками отрезков и т. д. в "очеловеченном" формате, хотя до NT Exploder'a ему ох как далеко. При желании можно восстановить все файлы за раз ("Recovery All"), или выбрать восстановление по маске (Mask).

Рисунок 4 восстановленная структура директорий

Хваленный Easy Recovery от Data Recovery Software вопреки своему названию простотой управления отнюдь не отличается и имеет довольно специфические особенности поведения. С настройками по умолчанию никаких файлов на отформатированном разделе он не увидит и чтобы заставить этого зверюгу заработать в Advanced Options (дополнительные опции) необходимо указать Ignore MFT (игнорировать MFT), но и в этом случае качество восстановления будет оставлять желать лучшего.

Рисунок 5 красивый интерфейс Easy Recovery — компенсация за низкое качество восстановления

ручное восстановление отформатированного диска

Нашей целью будет ручное восстановление всего отформатированного раздела без использования дополнительных носителей информации и дорогостоящих утилит от сторонних производителей. Все что потребуется — это любой редактор диска (предпочтительнее всего конечно же NT Explorer от Runtime Software, но на худой конец сойдет и бесплатный Disk Probe/Sector Inspector от Microsoft) и chkdsk.

Очевидно, что в процессе форматирования происходит необратимое разрушение большого количества ключевых структур данных, восстанавливать которые вручную было бы слишком затруднительно. Да это, собственно, и не нужно! Идея состоит в том, чтобы вернуть разделу потерянные файловые записи, а все остальные ремонтные работы поручить chkdsk'у — пускай старается.

Дизассемблирование показывает, что единственной структурой данных, без которой не может работать chkdsk, является атрибут $DATA файла $MFT. А раз так, все, что нам надо — воссоздать прежний $MFT:$DATA, разместив его поверх старых файловых записей. В простейшем случае (если $MFT:$DATA не фрагментирован) это достигается спекулятивным увеличением его длины. А как ее увеличить?

Запускаем NT Explorer, переходим в начало MFT (Goto → Mft), щелкаем по $MFT файлу, находим атрибут $DATA (80h) и увеличиваем поля Allocated Size/Real Size/Compressed Size на требуемую величину, параллельно с этим корректируя список отрезков (он же run-list). Поле Last VCN трогать не нужно — chkdsk исправит его и сам. Как определить длину не фрагментированного MFT-файла? Она равна разнице номеров первого и последнего секторов в начале которых присутствует сигнатура "FILE", умноженная на 512 байт (исключая сектора, принадлежащие $MFTMirr) Известные мне дисковые редакторы не поддерживают поиска последнего вхождения, поэтому соответствующую утилиту приходится писать самостоятельно. Впрочем, точную длину MFT определять совершенно необязательно и вполне допустимо взять ее с запасом — лишнее все равно отсеет chkdsk. Действуйте по принципу — лучше перебрать, чем недобрать.

Рисунок 6 ручное восстановление MFT. Подчеркнуты поля, подлежащие изменению

Коварный NT Explorer не позволяет редактировать поля в естественном режиме отображения, заставляя нас переключаться в HEX-mode и искать смещения всех значений самостоятельно. Найти заголовок атрибута $DATA очень просто — в его начале расположена последовательность 80 00 00 00 xx 00 00 00 01. В NTFS версии 3.0 она находится по смещению F8h от начала сектора. Поле Real Size во всех версиях NTFS располагается по смещению 30h относительно заголовка, а поля Allocated Size и Initialized Size соответственно по смещениям 28h/38h байт, причем значение Allocates Size должно быть кратно размеру кластера. Кстати, о кластерах. Убедитесь, что при переформатировании диска размер кластера не изменился, в

противном случае у вас ничего не получится. Как восстановить исходный размер кластера? Да очень просто — набраться мужества и переформатировать восстанавливаемый диск с ключом /A:x, где x – размер кластера. А как его определить? Возьмем любой файл с известным содержимым и проанализируем его run-list. Пускаем контекстный поиск по всему диску, находим файл, запоминаем (записываем на бумажке) его стартовый сектор, после чего открываем закрепленный за ним FILE Record, декодируем run-list и вычисляем номер первого кластера. Делим номер сектора на номер кластера и получаем искомую величину.

Теперь необходимо сгенерировать новый run-list. В общем случае он будет выглядеть так: 13 XX XX XX YY 00, где XX XX XX – трехбайтовый размер $MFT в кластерах, а YY – стартовый кластер. Стартовый кластер обязательно должен указывать на первый кластер MFT, в противном случае chkdsk не сможет работать. Если новый run-list длиннее нынешнего (а именно так скорее всего и будет) необходимо скорректировать длину атрибутного заголовка (она расположена по смещению 04h от его начала). Проделав эту нехитрую операцию, запустим chkdsk с ключом /F и блаженно откинемся на спинку кресла, созерцая как возрождаются наши милые папки и файлы. Единственное, что не восстанавливается — так это дескрипторы безопасности: всем файлам/папкам назначаются права доступа по умолчанию. В остальном же, с отремонтированным диском вполне можно работать, не опасаясь, что он рухнет окончательно. Файлы, ссылающиеся на несуществующие каталоги складываются в папку Found.xxx. Это "долгожители" пережившие несколько циклов переформатирования, в буквальном смысле вытащенные с потустороннего света.

Сложнее восстановить том, чей MFT сильно фрагментирован. Прежний run-list при форматировании был уничтожен, зеркальная копия так же пострадала. Ничего другого не остается, как собирать все фрагменты руками. Звучит намного страшнее, чем выглядит. В отличии от всех остальных файлов диска, $MFT-файл имеет замечательную сигнатуру FILE, присутствующую в начале каждой файловой записи. Все, что нам нужно — последовательно сканируя раздел от первого кластера до последнего, выписать начало и конец каждого из фрагментов, принадлежащих MFT. Затем из этой цепочки необходимо исключить $MFTMirr. Его легко узнать — он расположен в середине раздела и содержит копии файловых записей $MFT, $MFTMirr, $LogFile и $Volume, причем $MFTMirr ссылается сам на себя. Допустим, наш список выглядит так: 08h – 333h, 669h – 966h, 1013 – 3210h. В грубом приближении ему будет соответствовать следующий run-list: 12 2B 03 08 22 23 03 69 96 22 FD 21 13 10 00. (Подробнее о кодировании/декодировании run-list'ов см. "списки отрезков" в прошлой статье этого цикла).

"В грубом" потому, что мы не знаем в какой последовательности располагались эти отрезки в файле (порядок расположения фрагментов на диске далеко не всегда совпадает с порядком отрезков в run-list'е). Что произойдет, если порядок сборки $MFT-файла окажется нарушен? А вот что — внутри MFT все файловые записи ссылаются друг на друга по своему порядковому номеру, представляющим индекс массива. Эти ссылки необходимы для восстановления структуры директорий, организации hard link'ов и еще кое-чего. Ссылки на материнский каталог дублируются в индексах и восстанавливаются элементарно. Hard link'и мрут безвозвратно (ну разве что попробовать пересобрать $MFT-файл в другом порядке), но они практически нигде и никем не используются, как говорится, было бы что терять. По-настоящему туго приходится сильно фрагментированным файлам, занимающим несколько файловых записей, раскиданных по разным $MFT-фрагментам. Здесь выручает только перестановка фрагментов. К счастью, количество комбинаций обычно бывает невелико и процедура восстановления занимает совсем немного времени. Хорошая новость – начиная с NTFS версии 3.1 (соответствующей Windows XP) в MFT номера файловых записей хранятся в явном виде (четырехбайтовое поле, расположенное по смещению 2Ch от начала FILE Record), что делает задачу восстановления тривиальной.

восстановление после тяжелых разрушений

В результате сбоя содержимое дискового тома может стать недоступно операционной системе и при попытке чтения его оглавления будет выдаваться сообщения в стиле "Файл или папка повреждены. Чтение невозможно", "Нет доступа к диску", "Системе не удается найти логическое устройство" и т.д. Chkdsk говорит, что "Повреждена основная таблица файлов" и прекращает работу. Караул! Верните мой том!

Не паникуйте! Попробуйте запустить NT Explorer и посмотрите, что он покажет. Маловероятно, чтобы содержимое всего тома было утеряно целиком (это же что такое с ним

нужно было сделать?!). Если хотя бы часть файловых записей уцелела, R-Studio/GetDataBack/EasyRecovery их обязательно восстановят!

Анализ показывает, что основной причиной по которой chkdsk отказывается проверять том обычно становится порча файловой записи, описывающей $MFT. Если в процессе обновления $MFT внезапно отключить питание — такой исход событий вполне вероятен, особенно на жестких дисках с емким аппаратным кэшем — они не успевают завершить сохранение секторов на энергии, накопленной в конденсаторах, а вот их младшие собраться с этим справляются. Тоже самое происходит при неудачном перемещении $MFT файла или физическом разрушении первого MFT-сектора. Зеркальная копия $MFT во всех этих случаях остается цела, однако chkdsk по каким-то таинственным причинам не хочет ей пользоваться и вы должны восстановить ее самостоятельно. Просто скопируйте первый сектор $MFTMirr в первый сектор $MFT и громко скажите "ниндзя"! Поклонники Sector Inspector'a могут воспользоваться командным файлов следующего содержания.

```
SECINSPECT.EXE -backup     d:     backup.dsk    XXX    1
SECINSPECT.EXE -restore    d:     backup.dsk    YYY    CONFIRM
```

Листинг 1 командный файл для ручного восстановления $MFT из $MFTMirr, XXX – номер сектора $MFTMirr, YYY – номер сектора $MFT

Теперь можно смело пускать chkdsk. Если же он по-прежнему не работает, значит, либо поврежден загрузочный сектор (а методику его восстановления мы уже обсуждали), либо run-list файла $MFT:$DATA не совпадает с истинным началом MFT (найдите сектор с файловой записью $MFT внутри и исправьте run-list), либо размер кластера, прописанный в boot-секторе отличается от его фактического размера (о том как определять истинный размер кластера мы уже только что говорили).

Если же сбой был настолько серьезен, что вместе с $MFT пострадало и зеркало, задача сводится к восстановлению отформатированного диска. Кстати говоря, при тяжелых разрушениях файловой структуры, когда на диске образуется настоящий кавардак, восстановление тома полезно начинать с… его форматирования. Нет, это не первоапрельская шутка! Утилита format формирует заведомо исправные ключевые структуры, ну а подключение файловых записей — не проблема. Главное — сохраните run-list файла $MFT:$DATA, если, конечно, он еще уцелел. Все остальное — дело техники!

Рисунок 7 безуспешная попытка прочитать поврежденный том

>>>> восстановление NTFS-тома, отформатированного под FAT16/32

При переформатировании диска, операционные системы семейства NT никогда не изменяют тип файловой системы (разве что попросить их это сделать явно), поэтому непреднамеренное форматирование NTFS раздела в FAT16/32 крайне маловероятно. Windows 9x/MS-DOS, напротив, любой диск стремятся отформатировать под FAT16/32, не замечая, что на нем что-то уже находится. Непреднамеренная порча NTFS-разделов при установке Windows 9x/MS-DOS поверх Windows NT – обычное дело, через которое проходит уже второе поколение пользователей.

Стратегия спасения данных во всем похожа на восстановление NTFS-тома, отформатированного под NTFS, с той лишь разницей, что при создании таблицы размещения файлов (file allocation table), первые несколько тысяч файловых записей в MFT затираются безвозвратно (впрочем, их еще можно собрать руками, действуя по методике описанной в разделе "разгребая кластерные обломки" предыдущей статьи этого цикла).

Файлы, с уцелевшей FILE RECORD легко восстанавливаются R-Studio/GetDataBack/EasyRecovery. Для ручного восстановления всего тома его необходимо заново отформатировать под NTFS, предварительно определив количество секторов в кластере. Далее действуем по плану — увеличиваем размер $MFT, пускаем chkdsk и собираем все, что только можно собрать. Поскольку количество файлов, хранящихся на современных дисках, зачастую исчисляется многими миллионами, потеря первой тысячи из них не так уж и страша (если только по закону подлости это не будут самые ценные файлы).

>>>> врезка источники угрозы

Почему погибают дисковые разделы? Ниже приводится список наиболее распространенных причини, отсортированный в порядке убывания их "популярности":

- ошибки оператора, вирусы, троянские программы;
- отключение питания/зависание системы во время интенсивных дисковых операций, сопровождаемых обновлением MFT (например, удаление/добавление файлов или каталогов);
- некорректное поведение различных дисковых утилит (Partition Magic, Ahead Nero, Norton Disk Doctor и т. д.);
- физические дефекты оперативной памяти, приводящие к нарушению целостности дискового кэша и как следствие — порче самого диска;
- некорректное поведение привилегированных драйверов, случайно или преднамеренно "залетающих" внутрь служебных структур NTFS-драйвера;

>>>> врезка полезные советы — план превентивных действий по спасению данных

Чтобы предотвратить разрушение тома и упростить задачу восстановления данных, рекомендуется заблаговременно выполнить следующий комплекс мероприятий:

- **переместите $MFT подальше от начала раздела.** Первые секторы раздела, как показывает практика, самое небезопасное место. Во-первых, сюда стремятся вирусы (миф о невозможности прямого доступа к диску под NT всего лишь миф — читайте описание функции CreateFile и инструкцию на ASPI32-драйвер), во-вторых, некоторые утилиты (и в частности Ahead Nero) при некоторых обстоятельствах путают жесткий диск с оптическим накопителем, записывая образ не "туда", а, значит, в первых ~700 Мбайтах физического диска (не логического тома!) не должно быть ничего ценного, в-третьих, если вы вдруг запустите WipeDisk или любую другую затирающую утилиту, первым погибнет именно $MFT, без которого весь дисковый том просто груда мусора, в четвертых… да много разных причин можно найти. Просто переместите $MFT, вам что — трудно? Достаточно взять любой дефрагментатор, распространяющийся в исходных текстах (энтузиасты! ау! присоединяйтесь к проекту http://sourceforge.net/projects/opendefrag/) и слегка доработать его "напильником" под наши нужды. Естественно, валерьянка и резервная копия обязательно должны быть под рукой!
- не допускайте фрагментации $MFT-файла! Не создавайте на диске огромного количества мелких файлов и не заполняйте его более чем на 90%. Стандартный дефрагментатор, входящий в комплект штатной поставки Windows 2000/XP, не позволяет дефрагментировать $MFT и приходится прибегать к сторонним средствам, лучшим из которых на мой взгляд является **O&O Defrag Pro** от одноименной компании (www.oo-software.com). Это действительно профессиональный дефрагментатор, к тому же поддерживающий командную строку;
- периодически создавайте резервную копию файловой записи $MFT — для этого достаточно сохранить один-единственный (!) сектор — первый сектор MFT, номер которого содержится в boot, только не забывайте его периодически обновлять, ведь при добавлении новых файлов/каталогов, MFT планомерно расширяется и старые списки отрезков становятся все менее и менее актуальны;

Рисунок 8 нормальный дисковый том (MFT, выделенный желтым цветом, расположен в начале раздела)

Рисунок 9 "иммунизированный" дисковый том (MFT расположен в середине)

заключение

Наш затянувшийся разговор о восстановлении данных подходит к своему логическому концу, однако, NTFS не стоит на месте, а интенсивно развивается. И хотя до сих пор эти изменения носили чисто косметический характер, в Windows Longhorn все обещает кардинальным образом измениться. Microsoft активно работает над новой файловой системой — Windows File System или сокращенно WinFS, сроки выхода которой, кстати говоря, постоянно переносятся (разработка файловой системы это не шутка!).

По словам вице-президента Microsoft Боба Маглиа, WinFS это все та же NTFS, с прикрученным SQL и XML. Насколько изменятся базовые структуры файловой системы общественности до сих пор неясно. И уж совсем непонятно зачем NTFS понадобился реляционный SQL, когда эти возможности в нее закладывались изначально, просто их не успели доделать. Любой системный программист запросто напишет драйвер, принимающий SQL/XML запросы и транслирующий их в обращения к драйверу текущей файловой системы! Что-либо менять внутри NTFS совсем необязательно. Сдается мне, что это всего лишь очередной маркетинговый трюк, подталкивающих пользователей к переходу на ненавистный мне Longhorn!

С другой стороны, развитие NTFS можно только приветствовать, поскольку оно дает пищу всем специалистам по восстановлению данных, ведь старые утилиты с новой файловой системой скорей всего окажутся несовместимы.