

ВОССТАНОВЛЕНИЕ NTFS – UNDELETE СВОИМИ РУКАМИ

Меня смущает то, что сырые продукты создаются на интуиции и на гениальности их создателей. Но ведь за ними остается выжженная земля! Если OS/360 оставила за собой шлейф идей, людей, что оставляет за собой Windows?

Алексей Бабий
«Из жизни первобытных программистов»

Продолжая говорить о NTFS, сегодня мы рассмотрим технику восстановления удаленных файлов с помощью простейшего дискового редактора (типа Disk Probe) и утилиты chkdsk, а также дадим несколько советов по поводу создания собственного инструментария, который может быть запрограммирован на любом языке, имеющем доступ к win32 API.

КРИС КАСПЕРСКИ

Надежность NTFS – это одно, а ошибочно удаленные файлы – совсем другое. Файловая система, даже такая мощная, как NTFS, бессильна защитить пользователя от себя самого. Но вот предусмотреть «откат» последних выполненных действий она вполне может (тем более что транзакции и журналирование в NTFS уже реализованы). До совершенства остается всего лишь шаг. Увы! Microsoft топчется на месте, все никак не решаясь его сделать (задел, оставленный для будущих версий?). «Защита» от непреднамеренного удаления реализована исключительно на интерфейсном уровне, а это не только неудобно, но и ненадежно. Хорошо, если удаленный файл сохранился в «Корзине», но что делать, если там его нет? Эта статья рассказывает о методах ручного восстановления файлов, в том числе и с отсутствующей файловой записью, когда «покойника» приходится собирать по кластерам.

Внутри FILE_DISPOSITION_INFORMATION

IRP_MJ_SET_INFORMATION/ FILE_DISPOSITION_INFORMATION – это пакет, посылаемый драйверу при удалении файла (имейте это в виду при дизассемблировании). Что-

бы уметь восстанавливать удаленные файлы, необходимо отчетливо представлять, что происходит в процессе удаления файла с NTFS-раздела, а происходит при этом следующее:

- корректируется файл /\$MFT:\$BITMAP, каждый бит которого определяет «занятость» соответствующей файловой записи (FILE Record) в MFT («0» – запись не используется);
- корректируется файл /\$BITMAP, каждый бит которого определяет «занятость» соответствующего кластера («0» – кластер не используется);
- файловые записи, соответствующие файлу, помечаются как удаленные (поле FLAG, находящееся по смещению 16h от начала FILE Record сбрасывается в ноль);
- ссылка на файл удаляется из двоичного дерева индексов (технические подробности этого животрепещущего процесса здесь опускаются, поскольку восстановить таблицу индексов вручную сможет только гуру, да и зачем? в NTFS индексы играют вспомогательную роль – проще переиндексировать директорию заново, чем восстанавливать сбалансированное B*tree-дерево);

- обновляется атрибут \$STANDARD_INFORMATION каталога, хранившего удаляемый файл (время последнего доступа и т. д.);
- в /\$LogFile обновляется Sequence Number (изменения, происходящие в журнале транзакций мы не рассматриваем);
- Update Sequence Number следующих файловых записей увеличивается на единицу: сам удаляемый файл, текущий каталог, /\$MAF, /\$MFT:\$BITMAP, /\$BITMAP, /\$BOOT, /\$TRACKING.LOG.

Каталоги удаляются практически точно так же, как и файлы (с точки зрения файловой системы, каталог тоже файл, только особый – с двоичным B*tree-деревом индексов внутри).

Ни в том, ни в другом случае физического удаления файла не происходит, и он может быть легко восстановлен до тех пор, пока не будет затерта принадлежащая ему FILE Record, хранящая резидентное тело файла или список отрезков (run-list) нерезидентного содержимого. Утрата FILE Record очень неприятна, поскольку в этом случае файл придется собирать по кусочкам руками, и чем сильнее он фрагментирован – тем сложнее эта задача. В отличие от FAT, NTFS не затирает первого символа именем файла, чем значительно упрощает свое восстановление.

Автоматическое восстановление файла

Утилиты, восстанавливающие удаленные файлы, не входят в комплект штатной поставки Windows NT, и их приходится приобретать отдельно (а ведь в MS-DOS такая утилита была!). Опасаясь угробить файловую систему окончательно, большинство из них избегает прямой записи на диск – вместо этого вам предлагается считать удаленный файл и переписать его в другое место (но только не на сам восстанавливаемый раздел). Не слишком-то удачное решение! А если на остальных дисках свободного места нет или восстанавливаемый диск имеет всего лишь один логический раздел?

Предположим, вам необходимо восстановить базу данных в несколько гигабайт. Можно, конечно, подключить второй винчестер, скопировать ее туда, а затем обратно, но сколько же это займет времени, не говоря уже о том, что сервер лучше не выключать, а горячую замену поддерживают далеко не все жесткие диски! Другой недостаток подобных утилит – слишком медленная работа. Вместо того чтобы найти один-единственный файл, имя которого нам известно, они проводят полномасштабные маневры, сканируя весь раздел целиком. При работе с большими дисками на это уходит от одного до нескольких часов впустую потраченного времени.

С другой стороны, утилиты, вносящие изменения непосредственно в саму NTFS, рискуют серьезно повредить дисковый том, после чего ему не помогут даже профессионалы. Настоящие хакеры не доверяют никакому коду, кроме своего собственного, особенно если исходные тексты недоступны, а документация туманна и двусмысленна. Различные версии NTFS отличаются друг от друга. Последние радикальные изменения произошли в Windows XP (NTFS версии 3.1) – массив последовательности обновления

(Update Sequence Number-n-Array) переместился на шесть байтов вперед, а его место было отдано под выравнивание и поле номера текущей файловой записи (Number of this MFT Record). Восстанавливающая утилита должна не только поддерживать вашу версию файловой системы, но и безошибочно отличать ее от всех остальных (при обновлении Windows 2000 до Windows XP обновления файловой системы не происходит вплоть до переформатирования диска). Попробуй потом объясни начальству, «это не я, это она все испортила!».

Наконец в момент удаления файла утилит для его восстановления может просто не оказаться под рукой (что поделаешь – закон подлости!), и тогда приходится рассчитывать только на свои силы.

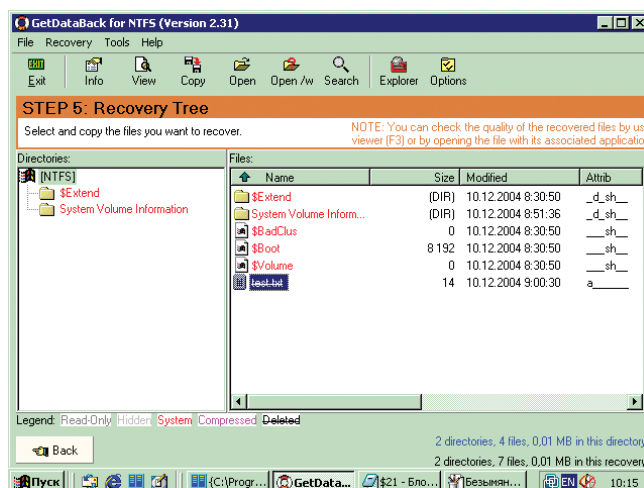


Рисунок 1. Утилита GetDataBack за восстановлением удаленных файлов

Ручное восстановление файла по FILE Record

Начнем с простейшего. Файл только что удален, и принадлежащая ему FILE Record еще не затерта. Как найти его на диске? Существует два способа – «теоретический» и «практический». Теоретический исключительно надежен, но требует дополнительных телодвижений, которых можно избежать, приняв ряд практических допущений.

Теоретически: извлекаем из boot-сектора указатель на MFT, извлекаем из нее первую запись (она описывает \$MFT), находим атрибут \$DATA (80h), декодируем список отрезков (data runs) и последовательно читаем все записи в MFT, анализируя содержимое атрибута \$FILE_NAME (30h) – имя файла (кстати, таких атрибутов у файла может быть несколько). Этот же атрибут хранит ссылку на материнскую директорию – если несколько одноименных файлов удалены из различных директорий, мы должны разобраться, какой из них наш.

Практически: в девяти из десяти случаев \$MFT-файл не фрагментирован и располагается практически в самом начале диска. Имена файлов хранятся по смещению EAh от начала сектора, в начале которого расположена сигнатура «FILE*» («FILE0» – в NTFS 3.1). Поэтому мы просто запускаем любой дисковый редактор (например, Disk Probe из комплекта Support Tools от Microsoft), вводим имя восстанавливаемого файла в юникоде и ищем его по смещению EAh (в NTFS 3.1 – F0h) от начала сектора.

Когда же искомое вхождение будет найдено, смотрим: находится ли в начале сектора сигнатура «FILE*»/«FILE0», и если нет – продолжаем поиск. Двухбайтовое поле по смещению 16h от начала сектора содержит флаги записи: 00h – запись не используется или была удалена, 01h – запись используется и описывает каталог, 02h – запись используется и описывает директорию. Встречаются и другие значения (04h, 08h... что они обозначают – неизвестно, может быть, вы сможете пролить свет на этот вопрос?).

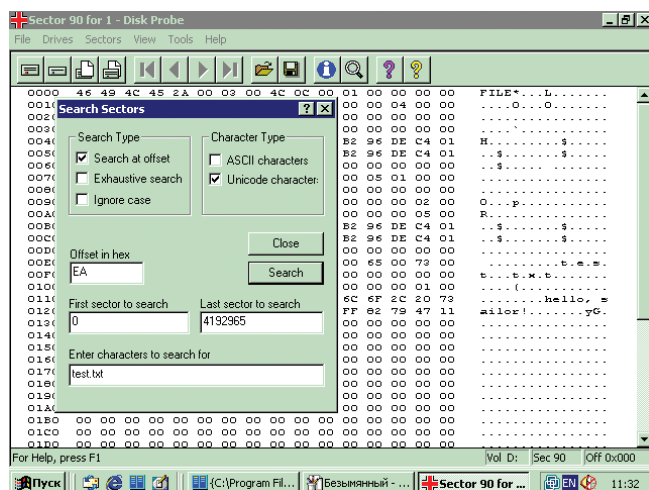


Рисунок 2. Ручное восстановление файла с помощью Disk Probe

```
C:\chkdsk D: /F
Тип файловой системы: NTFS.
Проверка файлов завершена.
Проверка индексов завершена.
Восстановление потерянных файлов.
Восстановление потерянного файла test.txt в файле каталога 5
Замена неправильного идентификатора безопасности для файла 29
Проверка дескрипторов безопасности завершена.
Исправление ошибок в атрибуте BITMAP основной таблицы файлов.
Windows сделала изменения в файловой системе.
```

```
1068290 КБ всего на диске.
 20 КБ в 2 файлах.
  4 КБ в 9 индексах.
  0 КБ в поврежденных секторах.
7894 КБ используется системой.
7392 КБ занято под файл журнала.
1060372 КБ свободно на диске.
```

```
Размер кластера:      2048 байт.
Всего кластеров на диске: 534145.
530186 кластеров на диске.
```

Рисунок 3. Восстановление удаленного файла при помощи chkdsk

Исправляем 00h на 01h, записываем изменения и... Ничего не выходит?! А что вы хотели! Ведь помимо этого необходимо еще, во-первых, сообщить файлу /\$MFT:\$BITMAP, что данная MFT-запись вновь используется, во-вторых, отобрать у файла /\$BITMAP номера кластеров, принадлежащие восстанавливаемому файлу, в-третьих, перестроить двоичное дерево индексов, хранящее содержимое каталога. Первые два пункта не проблема, но вот над последним придется попыхтеть. Или... просто запустить chkdsk с ключом /F. Он самостоятельно найдет «потерянный» файл и внесет все необходимые изменения в файловую систему. От нас потребуется только установить флаг по смещению 16h в единицу, а остальное – его забота. После этих нехитрых манипуляций файл оказывается в своем родном каталоге. Восстановленный!

Разгребая кластерные обломки

С нерезидентными файлами, хранящими свое тело вне MFT, ситуация обстоит не так плачевно, хотя проблем тоже хватает. Порядок размещения файла на диске хранится в run-list внутри файловой записи в MFT (теперь уже затертой), и потому возможен лишь контекстный поиск по содержимому. Запускаем диск-редактор, вводим последовательность, заведомо содержащуюся в удаленном файле, но не встречающуюся во всех остальных, и нажимаем «search». Для ускорения поиска можно искать только в свободном дисковом пространстве (за это отвечает файл /\$BITMAP). Известные мне редакторы пренебрегают этой возможностью (а зря!), однако утилиту «продвинутого» поиска несложно написать и самостоятельно.

Нефрагментированные файлы восстанавливаются элементарно. Просто выделяем группу секторов и записываем ее на диск (только ни в коем случае не на сам восстанавливаемый том!). Единственная проблема – как определить оригинальную длину? Некоторые типы файлов допускают присутствие «мусора» в своем хвосте (и тогда нам остается следовать правилу «лучше перебрать, чем недобрать»), а некоторые нет!

Если конец не удастся определить визуально (например, pdf-файлы завершаются сигнатурой «%%EOF»), проанализируйте заголовок файла – среди прочей полезной информации обычно там присутствует и его размер. Тут все зависит от структуры конкретного файла, и универсальных рекомендаций дать невозможно.

Если файл фрагментирован – ситуация практически безнадежна. Чтобы собрать разрозненные цепочки кластеров воедино, необходимо хорошо знать содержимое удаленного файла. В этом смысле NTFS восстанавливается намного хуже, чем FAT. Последовательность фрагментов файла, хранящаяся в File Allocation Table в виде однонаправленного списка, очень живуча. Если список не поврежден, достаточно лишь найти его первый элемент (а сделать это проще простого, поскольку он будет указывать на заголовок файла с вполне предсказуемым содержимым). Даже если список «разрубить» на несколько частей, они продолжают жить собственной жизнью и нам останется лишь подобрать комбинацию, как их правильно склеить воедино. Список гибнет лишь при полном затирании FAT, что случается, прямо скажем, нечасто. В NTFS же порядок фрагментов файла хранится в крохотных списках отрезков, и их гибель – обычное дело, после чего мы остаемся один на один с миллионом беспорядочно разбросанных кластеров. Текстовые файлы восстанавливаются без труда, но что делать, если это электронная таблица, графическое изображение или архив? Без знания стратегии выделения дискового пространства тут никуда. Порядок, в котором драйвер файловой системы находит подходящие свободные фрагменты, не определен и варьируется в зависимости от множества обстоятельств, однако кое-какие закономерности в нем все же присутствуют.

Анализируя списки отрезков сильно фрагментированных дисков, мне удалось установить следующее: сначала заполняются самые большие «дыры», двигаясь от конца MFT-зоны к концу диска. Затем драйвер файловой системы возвращается назад и начинает заполнять дыры поменьше.

ше, и так продолжается до тех пор, пока файл не оказывается на диске целиком. Последними заполняются дыры размером в один кластер.

Просматривая карту диска, представленную файлом /\$BITMAP, мы можем в точности восстановить порядок размещения фрагментов удаленного файла, наскоро собрав их воедино. Во всяком случае, теоретически. Практически же на этом пути нас ждут коварные препятствия. С момента создания восстанавливаемого файла карта свободного дискового пространства могла капитально преобразиться. Всякое удаление файлов высвобождает одну или несколько дыр, хаотично перемешивающихся с дырами восстанавливаемого файла, искажая картину. Как этому противостоять? Сканируем MFT в поисках записей, помеченных как удаленные, но еще не затертых. Декодируем run-list и вычеркиваем соответствующие им фрагменты из списка кандидатов на восстановление. Это существенно сужает круг поиска, хотя количество комбинаций, в которые можно собрать фрагментированный файл, по-прежнему остается велико. Но это еще что...

Самое «интересное» начинается, когда на диск одновременно записываются несколько файлов (например, скачиваемых с помощью ReGet из Интернета) или файл постепенно увеличивает свой размер (набираете дипломную работу в Word?), а в это время на диск записываются другие файлы. Когда к существующему файлу дописывается

крошечная порция данных, файловая система находит наименьшую дыру, затем следующую наименьшую дыру и т. д., вплоть до тех пор, пока маленькие дыры не исчерпаются, и тогда наступает черед дыр побольше. Как следствие, файл выходит сильно фрагментированным – это раз. Файл заполняется не от больших дыр к меньшим, а наоборот (т.е. происходит инверсия стратегии размещения) – это два. Маленькие фрагменты одного файла перемешиваются с маленькими фрагментами других файлов – это три.

Хуже всего поддаются восстановлению документы, созданные в MS Office, и вот почему: приложение создает большое количество резервных копий редактируемого файла как в текущем каталоге, так и в каталоге %TEMP%. Вот и разберись, какой фрагмент какому файлу принадлежит!

Проще всего восстанавливаются ZIP-архивы. Для этого вам даже не потребуется запускать дисковый редактор. Откройте временный файл на запись, сделайте seek на размер свободного дискового пространства, закройте файл. А теперь обработайте его утилитой pkzipfix.exe (или запустите стандартный pkzip.exe с ключом Fix). В «исправленном» файле волшебным образом появятся все уцелевшие ZIP-архивы! Внутренняя структура ZIP-архива такова, что pkzipfix легко распознает даже переупорядоченные блоки, поэтому высокая степень фрагментации ему не помеха.

Дефрагментация тоже происходит интересно. Стандартное API-дефрагментации в силу малопонятных ограниче-

Фрагментация и ее исследование

Существуют по меньшей мере две методики исследования стратегии выделения дискового пространства: статическая и динамическая. В первом случае мы просто запускаем дисковый редактор (предпочтительно Disk Explorer от Runtime Software) и анализируем run-list уже существующих файлов, записанных в различное время и различными способами (можно, например, скопировать файл с одного места в другое или попеременно увеличивать размер нескольких файлов – стратегии выделения свободного пространства в обоих случаях будут различны). Статический подход полезен тем, что дает бесценный статистический результат для всего тома целиком, однако, определяет лишь конечный результат, но не путь, которым он был достигнут.

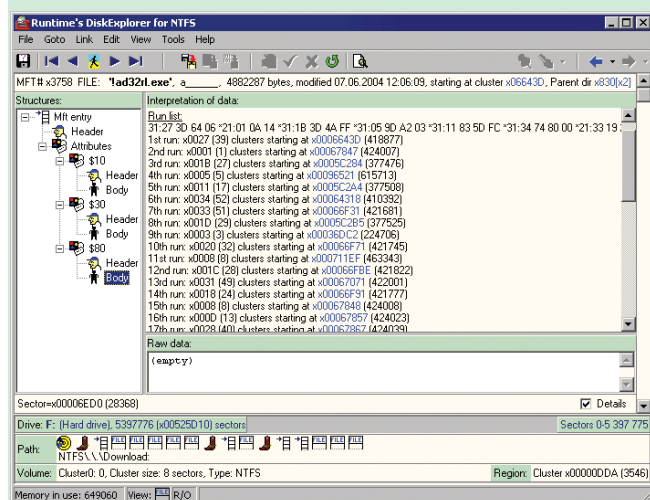


Рисунок 4. Статистический анализ стратегии выделения дискового пространства, выполняемый при помощи Disk Explorer от Runtime Software

#	Time	Request	Device	Result	Other
3346	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3347	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3349	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3350	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3352	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3353	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3355	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3356	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3358	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3359	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3361	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3362	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3364	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3365	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3367	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3368	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3370	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3371	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3373	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3374	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3376	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3377	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3379	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3380	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128
3382	22:11:36	IRP_MJ_WRITE	\\Device\\Harddisk1\\Partition4	SUCCESS	Sector: 13198024 Length: 128

Рисунок 5. Динамический анализ стратегии выделения дискового пространства, выполняемый при помощи дискового монитора Марка Руссиновича

Дисковый монитор Марка Руссиновича (<http://www.sysinternals.com>) позволяет заглянуть в «святыню» файловой системы и увидеть, как именно она выделяет дисковое пространство для файлов. Особенно интересно запускать его параллельно с дефрагментатором и chkdsk – тайное сразу становится явным.

Ничто так не постоянно, как временное

Если, несмотря на все усилия, восстановить удаленный файл так и не удастся, попробуйте отыскать его резервную копию. Многие приложения создают такие копии, но не все афишируют их присутствие. Не стоит также забывать о файле подкачки, временных файлах, дампе памяти и других источниках, которые могут хранить фрагменты восстанавливаемого файла (а иногда и весь файл целиком). Даже если они уже были удалены, возможно, принадлежащая им файловая запись еще не была затерта и восстановление не займет много времени.

ний оперирует не единичными кластерами, а блоками! Минимальный размер блока составляет 16 кластеров, причем начало блока должно быть кратно 16 кластерам в файле! Как следствие – количество мелких дыр после дефрагментации только возрастает, а непрерывных областей свободного пространства практически совсем не остается.

Кстати говоря, перемещать внутри MFT-зоны тоже ничего нельзя. «На томе C: свободно 17%, но только 5% доступно для использования дефрагментатора диска. Для эффективной работы дефрагментатор требует по крайней мере 15% доступного свободного места» – знакомое сообщение, не правда ли? «Недоступное» для дефрагментатора место находится внутри MFT-зоны (как мы помним, при форматировании диска под \$MFT-файл резервируется 10% от емкости тома, а затем по мере исчерпания дискового пространства \$MFT-файл усекается наполовину, и освободившееся пространство заселяется пользовательскими файлами).

Таким образом, для гарантированной работы дефрагментатора ему нужно $10\% + 15\% = 25\%$ свободного дискового пространства. Не слишком ли высокая плата за дефрагментацию? Если же у вас свободно свыше 25%, настоятельно рекомендуется создать на диске временный файл и выполнить seek, чтобы заполнить все более или менее крупные дыры, не давая их изуродовать дефрагментатору (естественно, после дефрагментации этот файл нужно удалить).

Кстати говоря, на сжатые файлы ограничение в 16 кластеров не распространяется, поэтому мелкие файлы очень

выгодно держать в сжатом состоянии – это существенно уменьшает фрагментацию тома. Почаще дефрагментируйте свой диск! Это не только увеличит быстродействие, но и упростит восстановление удаленных файлов с затертой FILE Record.

Заключение

Восстановление файлов – операция несложная, но нудная и кропотливая. Если по долгу службы или в силу иных обстоятельств вам приходится заниматься восстановлением постоянно, процесс можно «механизировать», написав несколько простых утилит. Чтобы получить доступ к логическому разделу в Windows NT, достаточно открыть одноименное устройство с помощью функции:

```
CreateFile("\\.\X:", GENERIC_READ, FILE_SHARE_READ, 0,
OPEN_EXISTING, 0, 0),
```

где «X:» – буква логического диска (подробности – в MSDN, или, как нынче модно его называть, – Platform SDK).

Не думайте, что все уже написано задолго до вас! Утилит, пригодных для профессионального восстановления данных, под NTFS до сих пор нет (во всяком случае в открытой продаже). Те же, что есть, страдают массой нелепых ограничений (например, не могут ограничить диапазон секторного поиска только свободным/занятым пространством). Так что дерзайте!

В следующей статье этого цикла мы рассмотрим технику восстановления отформатированных дисков, в том числе и тех, на которые после форматирования что-то писалось.

От лидера индустрии — лидерам программирования

Журнал для разработчиков программного обеспечения

www.microsoft.com/rus/msdn/magazine

Оформить подписку на журнал
можно
в интернет-магазине
издательства
www.ITbook.ru