сага о FSCTL_LOCK_VOLUME
или дисковые тома под замком

крис касперски, ака мыщъх, a.k.a. nezumi, a.k.a. souriz, a.k.a. elraton. no-email

набьем наш кальян травой, надербаненной на просторах MSDN, и раскурим малоизвестную, но очень интересную и актуальную тему блокировки дисковых томов как от непреднамеренного доступа (что полезно, когда винчестер находится в спячке), так и от жестокого вандализма (вирусных атак, например), чего можно добиться как штатными средствами операционной системы, так и мыщъх'иной утилитой, прилагаемой к журналу.

введение

Эта история началась много лет назад, когда "дятел" (винчестер типа IBM-DTLA-307015), используемый в качестве вспомогательного накопителя на базовой мыщъх'иной машине, в результате естественной выработки подшипника начал свистеть, шуметь и вибрировать всем корпусом. Мыщъх недовольно повел ущами (которым так хотелось тишины!!!) и, недолго думая, установил в настройках электропитания отключение диска через 3 минуты. Основой диск (Баракуда от Seagate) работал бесшумно и, поскольку к нему постоянно происходило обращение, никогда не отключался, а вот "Дятел" (куда складировались редко используемые вещи) по мыщъх'иному замыслу должен был скоро уснуть и долго не просыпаться (благо тачка перезагружается раз в месяц, а то и реже).

…положенные три минуты давно истекли, а "Дятел" все никак не отключался и продолжал шуметь еще добрых десять минут, прежде чем мыщъх не обнаружил, что это он конфликтует с Process Explorer'ом Марка Руссиновича, после закрытия которого, "Дятел", лениво зевнув, отправился на покой. Вибрации прекратились и воцарилась долгожданная тишина.

Мыщъх блаженствовал, но… через некоторое время "Дятел" проснулся вновь, раскручивая свой шпиндель с визгом свиньи, заживо сливаемой в унитаз. А… это агент Windows подсуетился и сообщил, что на таком-то диске осталось мало места и не мешало бы его почистить, но вместо этого разъяренный мыщъх завалил самого агента!!! Несколько часов блаженствующей тишины и… вновь грохот пробуждающегося Дятла!

Какая с… служебная программа потревожила его на этот раз?! Ах, MS Photo Editor, который при запуске зачем-то перечитывает все диски… А DVD-Decryptor при грабеже фильмов автоматически пытается записать их на диск с наибольшим количеством свободного пространства, сканируя все диски, включая спящие. Тысячи программ нахально лезут туда, куда их не просят и далеко не каждую из них можно от этого отучить (да и времени на этой уйдет)…

Обозначенная проблема довольно актуальна (особенно на компьютерах, где воткнуто большое количество дисков, но основная нагрузка ложиться на один из них, а остальные используются в резервных целях, например). И чтобы ее решить, мыщъх стал искать пути блокировки дисковых томов, что в конечном счете оказалось полезным не только для усыпления дисков, но и защиты данных от вирусных/хакерских атак. Даже если злоумышленник захватит администратора (вместе со всеми правами), он все равно не сможет ни открывать файлы, ни даже просматривать содержимое заблокированного тома.

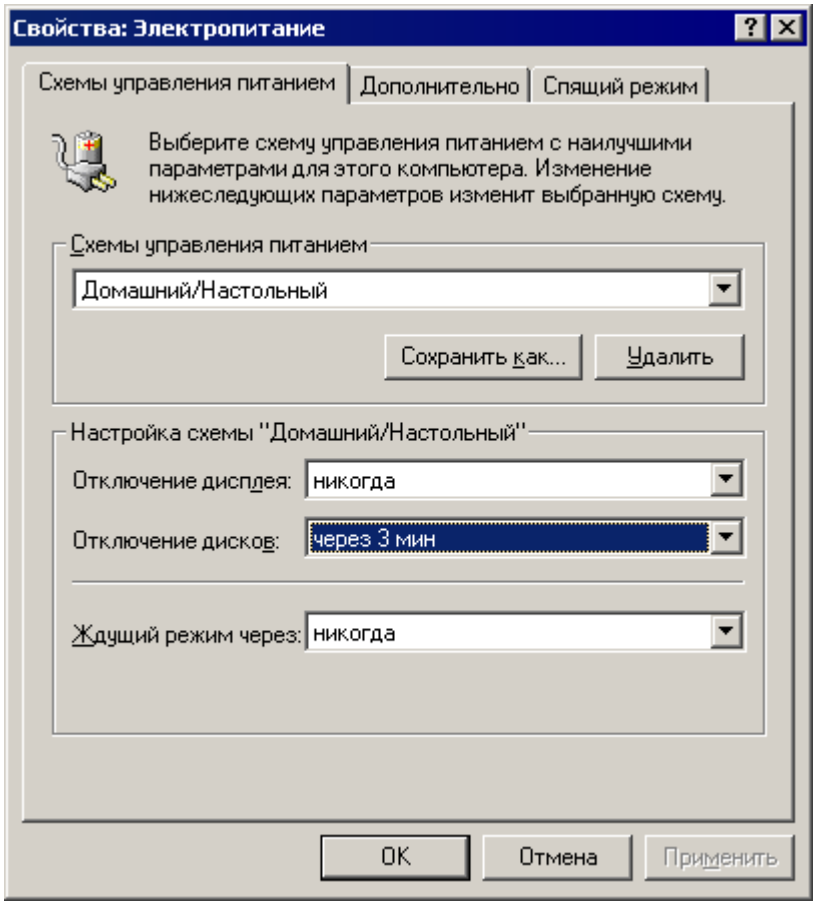

Рисунок 1 отключение неиспользуемых дисков через Менеджер Электропитания — бесполезная операция, поскольку, обращение к дискам (пробуждающие их ото сна) происходит даже тогда, когда мы меньше всего этого ожидаем

в дебрях MSDN

Всякий, кто стакивался с Windows (а кто ж с ней не сталкивался?!) знает, что chkdsk.exe блокирует дисковый том на время его "лечения". Еще бы! Попытка исправления ошибок, сопровождаемая записью на диск со стороны других приложений, рискует превратить базовые структуры данных в кашу! Весь вопрос в том — _как_ _именно_ chkdsk.exe осуществляет блокировку? Он же ведь, зараза такая, ничего не говорит и исходных текстов нет (на самом деле, уже давно как есть, утечки сырцов Windows совершались неоднократно и сейчас ими забиты все файло-обменные сети, но на тот момент у мыщъх'а их не было, так что приходилось плясать от печки).

Ну нет, исходных текстов — и не надо. Зато есть дизассемблер и MSDN. Распотрошив chkdsk.exe в IDA Pro и натравив на него монитор IOCTL-запросов, собранный на базе отладчика soft-ice (точка останова на API-функцию DeviveIoControl с записью всех аргументов в лог-файл), мыщъх определил, что блокировка дискового тома осуществляется путем посылки ему вполне документированной команды FSCTL_LOCK_VOLUME, подробно описанной в SDK вместе с другими командами, среди которых значится и FSCTL_DISMOUNT_VOLUME, с которой случилась одна забавная и в тоже время поучительная история.

Когда мыщъх (перед тем как выполнять ресерч chkdsk'а) спросил на форуме RSDN: какой IOCTL-код можно использовать для блокировки доступа к тому, то получил однозначный ответ: курить SDK в направлении на FSCTL_DISMOUNT_VOLUME название которой недвусмысленно свидетельствует, что она предназначена для размонтирования дисков, а в мире UNIX это равносильно потере доступа к ним на логическом уровне вплоть до последующего монтирования. Логично? Логично! Во только…

…Windows это не вам UNIX! Это совсем другой зверь. Вспомните, когда вы в Windows последний раз вручную монтировали дискету или CD-ROM. Не помните? Вот, и я не помню. Windows, в отличии от UNIX, _всегда_ монтирует диски автоматически! А SDK полезно читать

последовательно — от корки до корки. Несмотря на то, что в описании команды FSCTL_DISMOUNT_VOLUME недвусмысленно сказано, что "*a dismounted volume has the following properties: а) there are no open files. б) the operating system does not "know" about the volume;*" (размонтированный том обладает следующими свойствами: а) на нем отсутствуют открытые файлы; б) операционная система "забывает" о размонтированном томе" — казалось бы, то, что нам нужно! Ан нет, ниже следует убийственная приписка (кстати, напечатанная тем же самым шрифтом): "*the operating system tries to mount an unmounted volume as soon as any attempt is made to access it. For example, a call to GetLogicalDrives triggers the operating system to mount any unmounted volumes*" ("операционная система пытается смонтировать демонтированный том при первой же попытке доступа к нему, в частности, вызов функции *GetLogicalDrives* заставит операционную систему смонтировать все демонтированные тома").

Другими словами, как только мы нажмем "ALT-F1"/"ALT-F2" в FAR'е или попытаемся просмотреть содержимое тома, операционная система автоматически смонтирует его и никакой блокировки не получится! Ну и зачем раздавать такие советы?! Мыщъх полдня убил на эксперименты с FSCTL_DISMOUNT_VOLUME пока не понял, что его обманули. Вот и верь после этого людям…

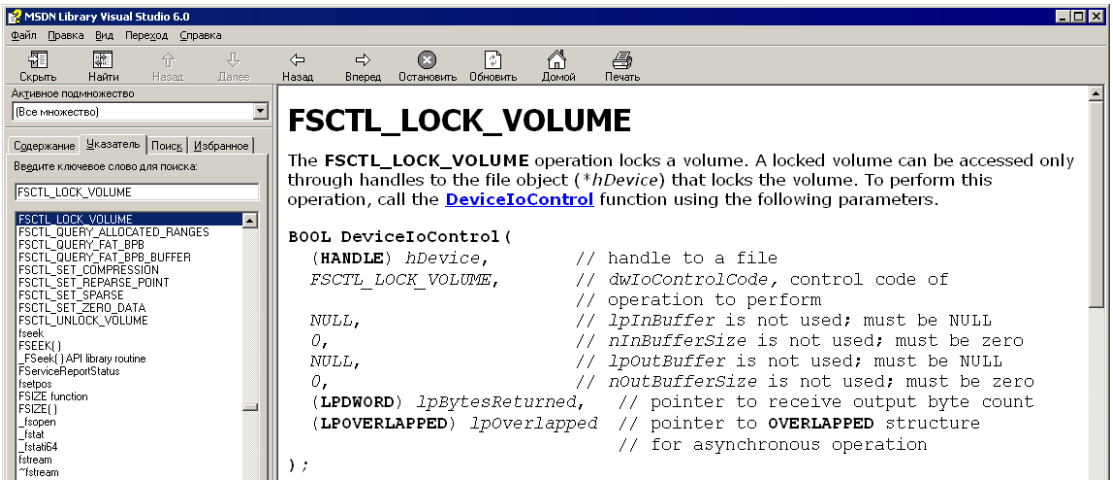

Рисунок 2 FSCTL_LOCK_VOLUME на MSDN

Самое смешное, что когда мыщъх снова вылез на форум с уже готовым решением ("голый" вызов FSCTL_LOCK_VOLUME, рекомендованный в описании команды FSCTL_DISMOUNT_VOLUME самой MS), его тут же закидали тухлыми яйцами с криками, что использовать FSCTL_LOCK_VOLUME без FSCTL_DISMOUNT_VOLUME ни в коем случае нельзя! Дескать? при этом операционная система не сбрасывает дисковые буфера и если случайно забыть смонтировать том перед завершением работы Windows, диску придет капец.

Что ж, открываем SDK и читаем: "*the system flushes all cached data to the volume before locking it. For example, any data held in a lazy-write cache is written to the volume*" ("операционная система сбрасывает все дисковые буфера на том перед тем как заблокирует его, в частности, все данные, находящиеся в кэш'е "ленивой записи" выгружаются на том"). Короче, на счет потенциальных разрушений можно не волноваться. Вызывать FSCTL_DISMOUNT_VOLUME до блокировки тома — бесполезно, а после — невозможно (DeviceIoControl возвратит ошибку, в полном соответствии со спецификациями Microsoft).

Так что MSDN рулит, а все "советчики" с форумов идут в баню.

скелет программы

Вдоволь накурившись SDK, пишем следующий код, "скелет" которого приведен на листинге 1. Обработка ошибок с прочими второстепенными деталями для упрощения понимания опущена:

```
HANDLE hDevice; DWORD lpBytesReturned;
hDevice = CreateFile (buf,GENERIC_READ|GENERIC_WRITE,
                      FILE_SHARE_READ|FILE_SHARE_WRITE,0,OPEN_EXISTING,0,0);
DeviceIoControl(hDevice,FSCTL_LOCK_VOLUME,NULL,0, NULL,0,&lpBytesReturned,0);
```

Листинг 1 скелет программы, блокирующей дисковые тома

Прежде, чем отправлять тому команду FSCTL_LOCK_VOLUME его необходимо открыть API-функцией CreateFile с флагами FILE_SHARE_READ|FILE_SHARE_WRITE, OPEN_EXISTING. Ну, с OPEN_EXISTING все понятно. Мы же не собираемся создавать новый дисковый том (имя которого, кстати говоря, записывается в формате "\\.\X:"), а открываем уже существующий, но вот флаги FILE_SHARE_READ|FILE_SHARE_WRITE способы легко ввести в заблуждение начинающих программистов, создавая впечатление, что заблокированный том будет доступен для совместной записи/чтения со стороны других приложений. Ничего подобного! Том будет заблокирован намертво! А эти флаги относятся к самому устройству, но не команде FSCTL_LOCK_VOLUME.

Перед блокировкой тома необходимо закрыть все открытые файлы и директории (т. е. не просматривать никакой каталог блокируемого тома в FAR'e), а так же заручиться правами администратора (например, прибегнув к помощи службы "runas"), в противном случае DeviceIoControl возвратит ошибку. Стоп! Заблокировать том, мы заблокируем, а как его потом разблокировать обратно? Возвращаемся к MSDN, читаем: "*a locked volume remains locked until one of the following occurs: а) the application invokes the FSCTL_UNLOCK_VOLUME DeviceIoControl operation to unlock the volume; б) The handle closes, either directly through CloseHandle, or indirectly when a process terminates*" ("том будет оставаться заблокированным пока не совершится одно из следующих действий: а) приложение, заблокировавшее том, не вызовет команду FSCTL_UNLOCK_VOLUME; б) дескриптор тома не будет закрыт вызовом CloseHandle или путем завершения процесса").

Небольшое пояснение по поводу пункта (а): определенный артикль "the application" подразумевает вполне определенное приложение (в данном случае, приложение, вызвавшее FSCTL_LOCK_VOLUME), никакое другое приложение не может разблокировать том посылкой команды FSCTL_LOCK_VOLUME (тогда бы в документации использовался неопределенный артикль "an application", т. е. "any application"). Вот такая, значит, лингвистика. И эксперименты ее полностью подтверждают. То есть, если мы блокируем том, то остальные приложения, даже обладающие правами администратора, не смогут до него добраться, во всяком случае на логическом уровне. В принципе, обладая правами администратора, нетрудно прочитать/записать содержимое диска на физическом уровне, открыв устройство \\.\PHYSICALDRIVE2, например, но это уже перебор. Нормальная малварь так не поступает, не говоря уже о пользователе типа "жена".

Теперь по поводу пункта (б). Как только наше приложение завершится, все заблокированные тома будут автоматически разблокированы и потому ничего другого не остается, кроме как резидентно болтаться в памяти, взаимодействуя с пользователем посредством командной строки или еще как.

Обычно, приложения, не создающие окон (т. е. не имеющие пользовательского интерфейса) и работающие в фоновом режиме, оформляются в виде служб (они же системные сервисы), но мыщъх'у программировать службу было лень и он поступил проще: создал консольное приложение, а при его сборке указал линкеру, что это GUI, в результате чего удалось избежать создания консольного окна, загрязняющего своим присутствием "Рабочий Стол".

Уничтожение процесса через "Диспетчер Задач" приводит к автоматической разблокировке всех заблокированных томов, но это некрасиво и потому было решено создать именованный семафор "nezumi_lock_mutex", дожидаясь его освобождения с помощью API-функции WaitForSingleObject, вгоняющий процесс в глубокий сон, чтобы он не кушал процессорное время. Повторный запуск программы с ключом "-release" освобождает семафор, передавая управление WaitForSingleObject, пробуждая процесс ото сна и тут же завершающий его, с закрытием дескрипторов всех заблокированных томов, что, как уже говорилось выше, приводит к их разблокировке.

Достаточно простой, удобный и необычный способ блокировки дисков от (не)преднамеренного обращения, а главное — надежный! Поскольку, большинство из нас хранит архивные копии прямо на жестком диске (это удобнее, чем стример или CD/DVD-R/RW), то имеет смысл заблокировать архивный том, снимая блокировку только на время обновления архивов.

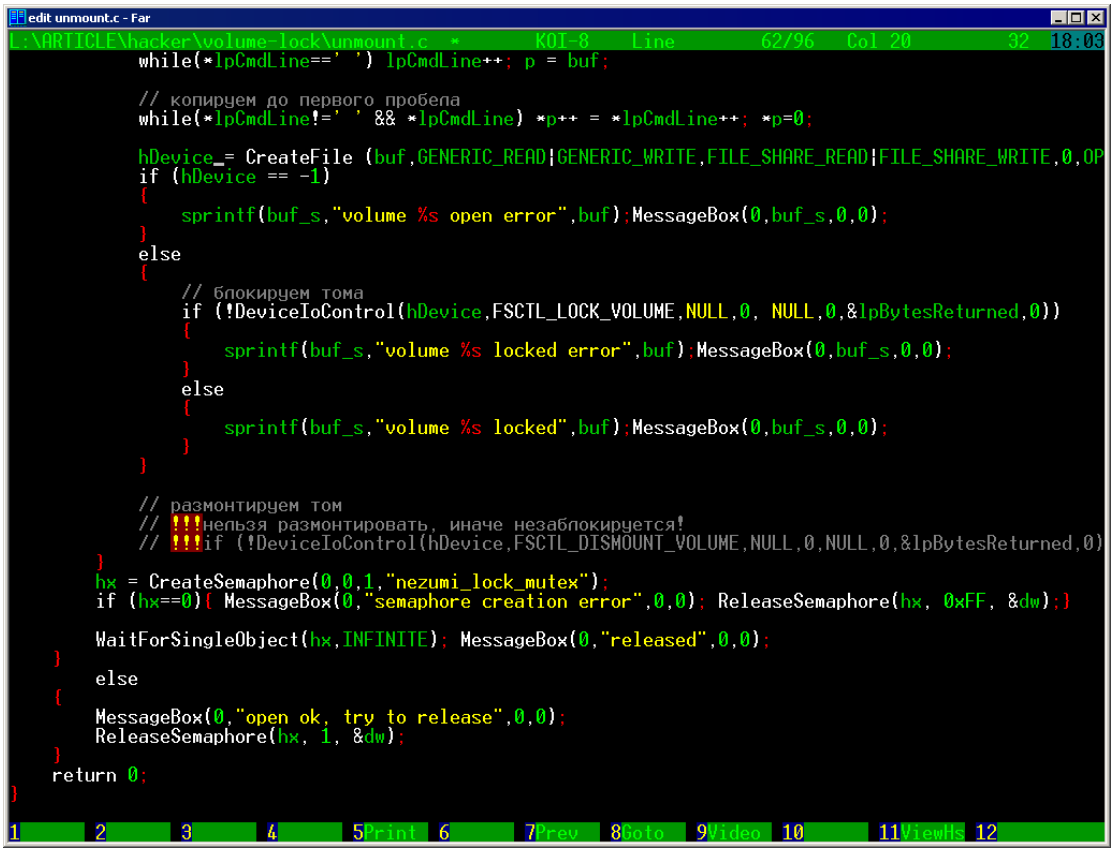

Рисунок 3 мыщъх'иная программа для блокировки дисков

как пользоваться программой

Поскольку, в комплект штатной поставки операционной системы не входит утилиты для блокировки дисковых томов, мыщъх наскоро написал свою собственную, обозвав ее unmount.c, исходные тексты которой прилагаются к статье (как видно из названия, она создавалась еще в те далекие времена, когда мыщъх верил в могущество команды FSCTL_DISMOUNT_VOLUME).

Процедура сборки компилятором MS Visual C++ выглядит так:

```
$cl.exe /c unmount.c
$link unmount.obj /SUBSYSTEM:WINDOWS USER32.LIB
```

Листинг 2 сборка утилиты компилятором MS Visual C++ из командной строки

Внимание: если просто скопировать исходный текст в окно IDE и сказать "Build" мы получим пулеметную очередь ошибок, а все потому, что по умолчанию IDE настроена на компиляцию Си++ программ, а эта написана на чистом Си со всеми его вольностями в преобразованиях типов.

Для самых ленивых предлагается уже готовый к исполнению unmount.exe, который следует запускать из командной строки со списком дисков, которые необходимо заблокировать, например: "unmount.exe \\.\X: \\.\Y: \\.\Z:", а разблокировать ранее заблокированные тома — "unmount.exe -release". Выборочная разблокировка отдельных томов в программе непредусмотрена (желающие могут добавить ее сами). Так же, в некоторых оболочках типа FAR'a во избежание возможной блокировки командой строки, ожидающей завершения приложения (которое в данном случае завершаться не собирается) рекомендуется команду "start", поддерживаемую всеми версиями Windows, линейки NT.

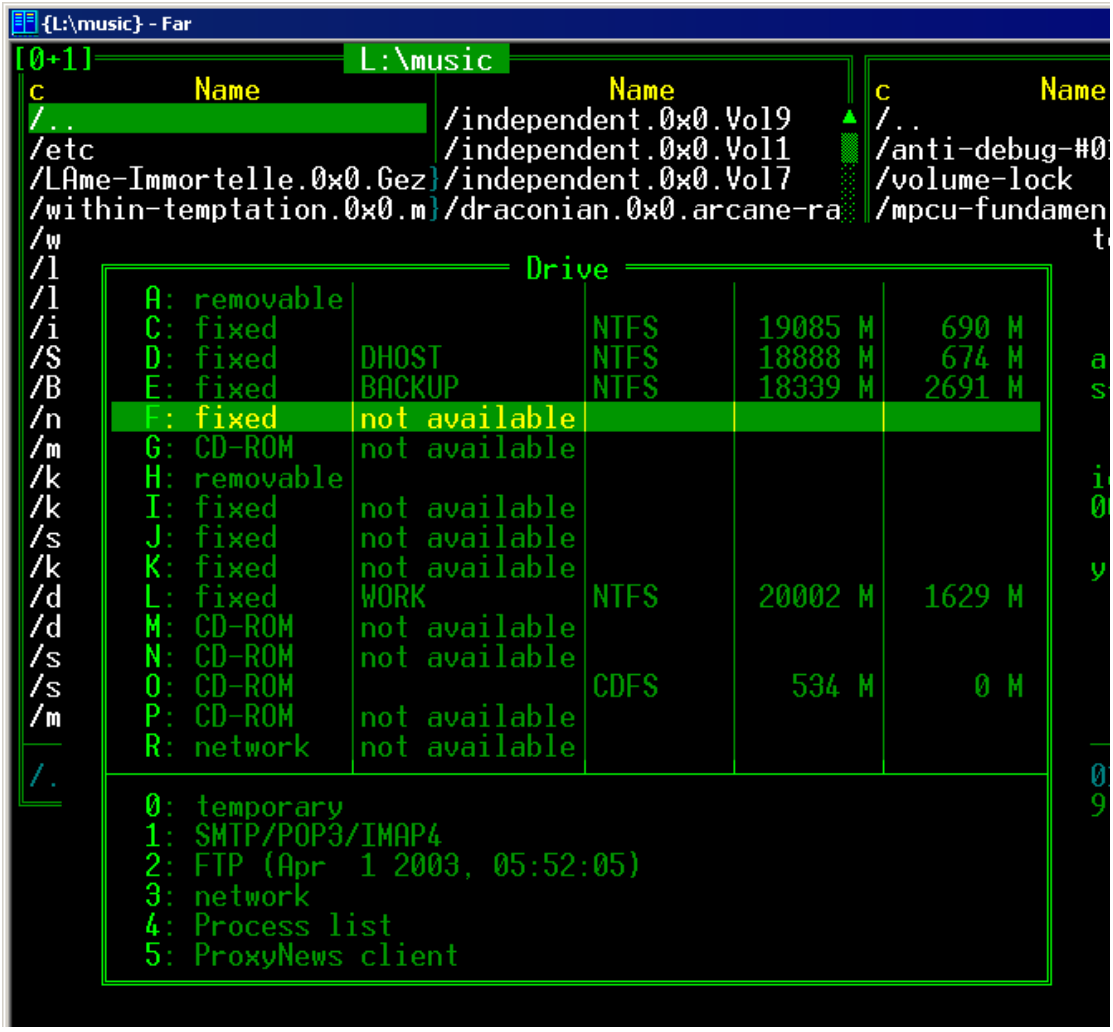

Рисунок 4 заблокированные жесткие диски видятся в FAR'е как недоступные (not available)

Примечание: программа содержит несколько некритичных ошибок, которые мыщъх так и не удосужился исправить. В частности, если запустить unmount не под администратором, то семафор будет успешно создан, а вот при попытке блокировки томов возникнет ошибка, но семафор останется не уничтожен и повторный запуск утилиты под администратором ни к чему не приведет, пока пользователь (от имени которого создался семафор) не вызовет unmount.exe с ключом "-release". Впрочем, все это мелочи. Главное — руководящая идея!

размонтирование дисковых томов

Альтернативный метод защиты дисков от (не)преднамеренных обращений к данным заключается в удалении точки монтирования, что легко сделать с помощью штатной утилиты MOUNTVOL, входящий в комплект поставки Windows, кажется, еще начиная с W2K.

При запуске без ключей она выдает список точек монтирования, который выглядит приблизительно так:

```
\\?\Volume{98fc8062-566a-11d9-82a2-806d6172696f}\
        C:\

\\?\Volume{98fc8063-566a-11d9-82a2-806d6172696f}\
        D:\

\\?\Volume{98fc8064-566a-11d9-82a2-806d6172696f}\
        E:\

\\?\Volume{98fc8065-566a-11d9-82a2-806d6172696f}\
        L:\
```

```
\\?\Volume{98fc8066-566a-11d9-82a2-806d6172696f}\
       I:\

\\?\Volume{98fc8067-566a-11d9-82a2-806d6172696f}\
       J:\

\\?\Volume{98fc8068-566a-11d9-82a2-806d6172696f}\
       K:\

\\?\Volume{98fc8069-566a-11d9-82a2-806d6172696f}\
       F:\
```

Листинг 3 имена томов и точки монтирования

Длинная строка циферок, начинающаяся с "\\?\Volume" — это и есть имя тома (не путать с меткой диска!), ниже идет точка монтирования, обычно представляющая собой букву, однако, операционные системы семейства NT позволяют монтировать диски на каталоги любого другого тома (при условии, что этот каталог пустой), позволяя создавать файловые иерархии, характерные для UNIX-систем.

Допустим, мы хотим размонтировать диск "A:\", что мы должны для этого сделать? Во-первых, приобрести права администратора, а, во-вторых, отдать команду:

```
$MOUNTVOL.EXE A: /D
```

Листинг 4 удаление точки монтирования диска

Теперь диск A:\ "волшебным" образом исчезает из системы и больше не отображается ни в "Проводнике", ни в FAR'e (см. рис. 5), создавая впечатление, что его _вообще_ нет (а он ведь есть). И потому малвари или другому программному обеспечению до него теперь как просто не добраться"

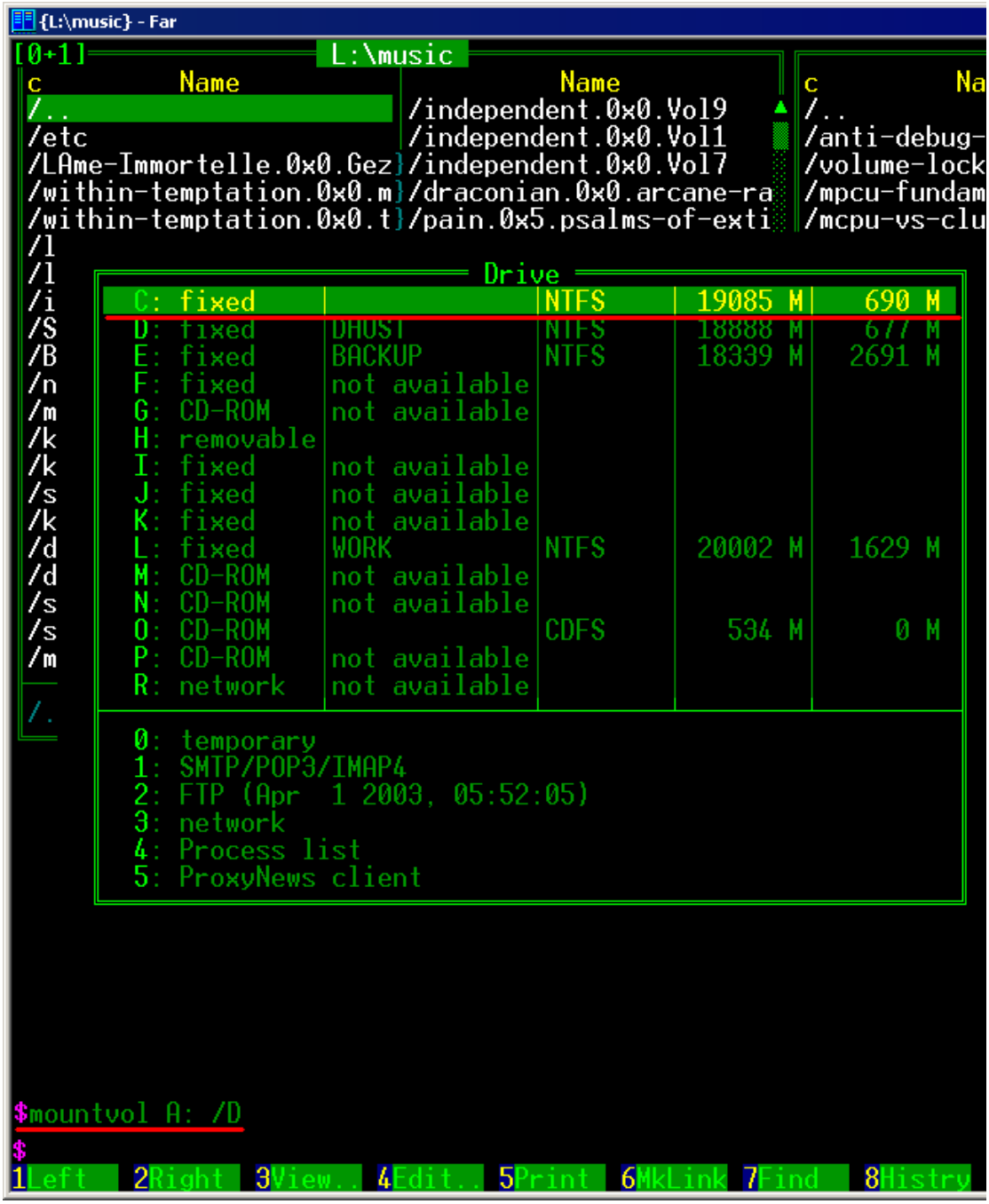

Рисунок 5 куда подевался наш диск "A:\"?

Хорошо, а как смонтировать диск обратно? Нет ничего проще!

```
$REM вызываем MOUNTVOL без ключей, чтобы увидеть имена дисков
$MOUNTVOL.EXE
Возможные значения ИмениТома вместе с текущими точками подключения:

\\?\Volume{98fc8062-566a-11d9-82a2-806d6172696f}\
        C:\

\\?\Volume{98fc8063-566a-11d9-82a2-806d6172696f}\
        D:\

…

\\?\Volume{98fc8060-566a-11d9-82a2-806d6172696f}\
        *** НЕТ ТОЧЕК ПОДКЛЮЧЕНИЯ ***

\\?\Volume{e34f31c2-5654-11d9-9ec4-c140ca76d429}\
```

```
        H:\

REM видим имя \\?\Volume{98fc8060-566a-11d9-82a2-806d6172696f}\
REM без точек подключения. это и есть наш бывший диск A:\
REM сделаем из него диск B:\
$MOUNTVOL B: \\?\Volume{98fc8060-566a-11d9-82a2-806d6172696f}\
```

Листинг 5 восстановление точки подключения

Вот, теперь диск B:\ (бывший A:\) вновь появился в системе (см. рис. 6) и его можно юзать наравне с другими.

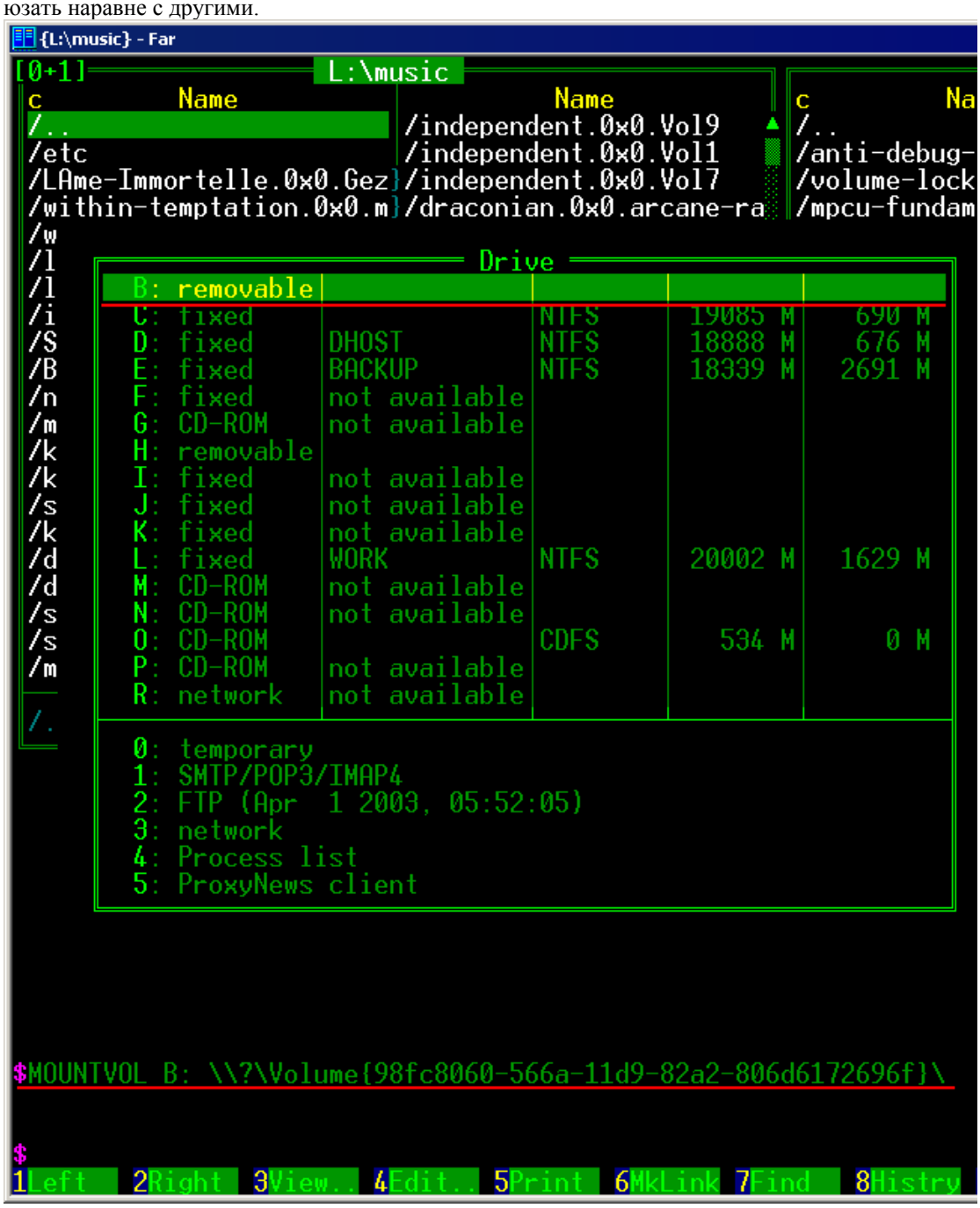

Рисунок 6 здравствуй, диск "B:\"!

Кстати говоря, операции по удалению/восстановлению точки монтирования можно осуществлять и через "Менеджер Дисков" (панель "Администрирование"), там — в графике — оно понагляднее будет, зато командная строка легко автоматизируется путем написания bat-файлов. Но тут на вкус и цвет все фломастеры разные.

По надежности блокировка слегка выигрывает у операции удаления точки монтирования (восстановить точку монтирования может любое приложение, а не только то, которое ее удалило), однако, зловредное программное обеспечение, умеющее работать с точками монтирования мыщъх'у пока что не встречалось. К тому же заблокированные диски остаются "болтаться" в "Проводнике" и FAR'е, мозоля своим присутствием глаза, а вот удаление точек монтирования убирает их из системы, но это опять-таки вопрос вкуса, а вкусы у всех разные.

заключение

Windows — мощная и интересная система, таящая под своим капотом огромные возможности и пока остальные устанавливают сложные (и дорогостоящие) защитные комплексы, мы — хакеры — успешно обходимся своими лапами и хвостом, получая ничуть не худший, а зачастую даже лучший результат!!!