Разработка приложения для восстановления удаленных файлов на NTFS диске

Лабораторная работа

Ревизия: 0.1

Последняя модификация: Vlad Kovtun

Дата последней модификации: 17.10.2010 14:06:00

История изменений

12.10.2010 – Версия 0.1. Первичный документ. Влад Ковтун

Содержание

Лабораторная работа 11. Разработка приложения для восстановления удаленных файлов на NTFS диске

Вопросы

- Постановка задачи.
- Методические рекомендации.

Постановка задачи

Цель

1. Разработать консольное (оконное) приложение для восстановления удаленных файлов на NTFS диске для ОС Windows.

Задачи

1. Ознакомиться с особенностями построения файловой системы NTFS [2-6].

2. Ознакомиться с особенностями использования функции Win API `CreatFile` для доступа к оборудованию [1].

3. Ознакомиться с классам `CNTFSDrive` и `CMFTRecord` для работы с NTFS диском и MFT записями [3].

4. Оформить отчет к лабораторной работе. В отчете обязательно указать, каким образом производится тестирование на корректность.

Задание

Опишем основные ключи командной строки, которые следует реализовать для приложения.

Для управления приложением предлагается использовать ключи командной строки:

- /vl – выдается список всех томов с указанием типа файловой системы, пронумерованный от 0.
- /vn:<volume number> – номер тома NTFS, который следует проанализировать на наличие удаленных файлов.
- /o:<filename> – полный путь к файлу, либо имя файла, который будет хранить список NTFS томов либо список удаленных файлов на заданном томе. Если файл не указан, то выводится весь список на экран. В списке указывается: номер записи в MFT и все ее содержимое (значения всех атрибутов, кроме тела файла).
- /irn:<record number> - номер записи в таблице MFT для файла, который следует восстановить.
- /uf:<filename> - имя файла или полный путь к нему, куда следует восстановить удаленный файл.
- /? - вывод информации о допустимых ключах командной строки.

Пример

Получение списка NTFS томов, результат выводится в файл `ntfsdrives.txt`.

```
C:/>ntfsundelete.exe /vl /o:ntfsdrives.txt
```

Получение списка файлов (MFT записей) на первом NTFS томе, результат выводится в файл `fileslist.txt`.

```
C:/>ntfsundelete.exe /vn:1 /o:fileslist.txt
```

Восстановление удаленного (создание копии существующего файла) расположенного в 128 MFT записи (с именем `logs.dat`) на 0-ом NTFS томе в файл с именем Получение списка файлов (MFT записей) на первом NTFS томе, результат выводится в файл `logs_copy.dat`.

```
C:/>ntfsundelete.exe /vn:0 /irn:128 /uf:logs_copy.dat
```

Вывод

Во время работы приложения, рекомендуется выводить информацию о статусе приложения, а также о корректности его работы на консоль.

Допускается перенаправление вывода консоли в текстовый файл, например:

```
C:/>ntfsundelete.exe /vn:1 >log.txt
```

Требования

- Архитектура приложения строится по модульному принципу.
- За основу принимается стандартная библиотека C++ (в случае разработки на C++).
- Рекомендуется использовать защищенные ресурсы и указатели.
- Обязательным является обработка исключений.
- Исходный код обязан быть комментирован. Для C++ следует использовать нотацию doxygen [8]. Для C# следует использовать нотацию Microsoft Visual Studio [7].

Методические указания для самостоятельной работы

Тестирование

Тестирование приложения осуществляется в несколько этапов:

- Разработка Unit Test в рамках проекта. Данный подход позволит проверить корректность реализации алгоритма.
- Воспользоваться тестовыми пакетными файлами для проверки корректности функционирования.

Теоретические сведения

NTFS

NTFS является быстрой файловой системой с поддержкой журналирования, что серьезно влияет на ее сложность. Как известно журналируемая файловая система может восстанавливаться после неожиданных (внезапных) разрушений. Что касается журналирования, то эта функция будет детально описана ниже. NTFS позволяет реализовать несколько функций в ОС Windows, которые не поддерживаются FAT, например: безопасность, большие хранилища, и т.д.

NTFS начинается с MFT (Master File Table), которая выглядит как FAT (File Allocation Table) в файловой системе FAT. Загрузочный сектор NTFS указывает на эту таблицу. После старта системы, ОС читает загрузочный сектор, отыскивает MFT и загружает ее перед каждой файловой операцией. В отличие от FAT, MFT может находиться в любом месте диска. По аналогии с FAT, существует две копии MFT для каждой NTFS файловой системы. Обычно, первая находится в начале NTFS диска, а вторая MFT, называемая копией или зеркалом, находится в середине NTFS диска.

На рис. 1 представлена структура NTFS диска (раздела) после форматирования.

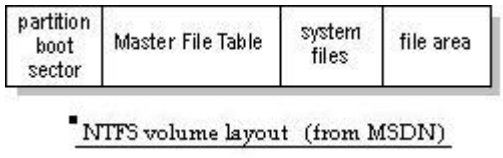

Рис. 1. Структура NTFS раздела

MFT

Все файлы в NTFS представлены записью в MFT. Далее под файлом будем понимать системные файлы, директории, обычные файлы, скрытые файлы и т.д. Даже MFT и ее копия представлены записью в самой себе. Первая и вторая записи в MFT являются записями, описывающими саму MFT и ее копию, соответственно. Первые 16 записей в MFT являются зарезервированными для системных файлов. Каждая запись в MFT обладают одинаковым размером. Запись MFT обычно занимает до 1024 байт и может изменяться в этом диапазоне. Размер MFT записи в байтах указан в загрузочном секторе (т.е. в BPB). На рис. 2 представлена организация MFT и системных файлов.

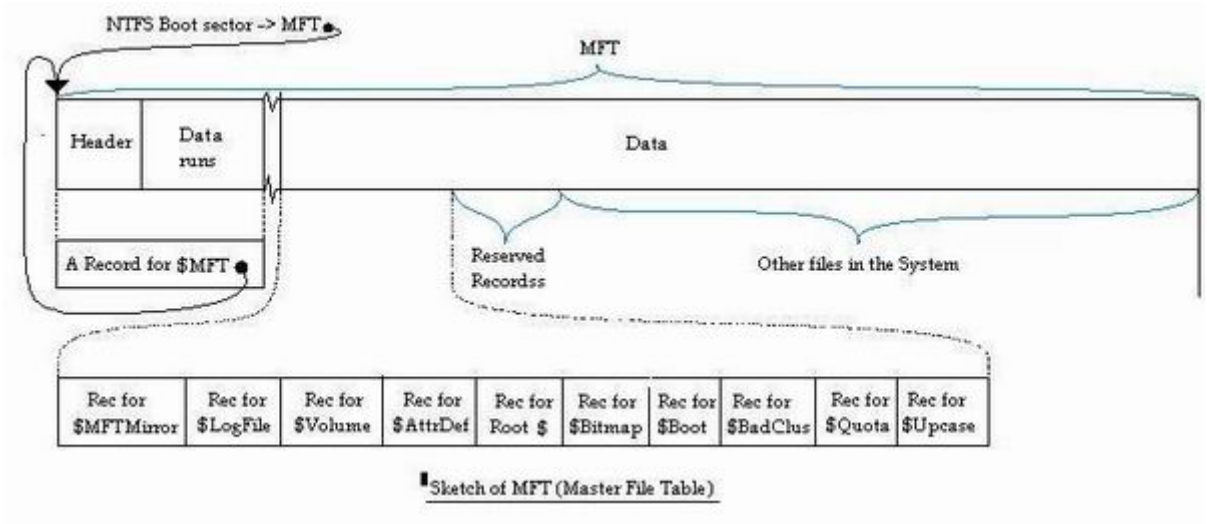

Рис. 2. Организация MFT и системных файлов

- **$LogFile:** Файл журнала позволяет восстанавливать файловую систему. Файл журнала является системным файлом, что может быть найден ранее процесса загрузки ОС и использован для восстановления раздела. Все транзакции заносятся в журнал непосредственно перед их выполнением над разделом. В случае сбоя системы в процессе выполнения транзакции, та часть транзакции, которая была выполнена успешно, будет возвращена к прежнему состоянию либо откачена. Операция отката возвращает файловую систему к предыдущему, ранее известному устойчивому состоянию, как будто бы транзакция никогда и не выполнялась.
- **$Volume:** Имя раздела, версия NTFS, и прочая информация о разделе.
- **$AttrDef:** Таблица атрибутов имен, номеров и описаний. MFT атрибуты определяются в этом разделе и могут быть расширены.
- **$:** Запись используется для корневого каталога. Все каталоги и файлы перечисляются именно здесь.
- **$Bitmap:** Битовая карта-образ заполнения блоков в файловой системе. Каждый бит представляет блок в файловой системе, который означает данный блок свободен либо занят.
- **$Boot:** Запись содержит программу начальной загрузки для раздела, если таковой является загрузочным.
- **$BadClus:** Запись содержит информацию о размещении поврежденных секторов в разделе.
- **$Quota:** Дисковая квота для каждого пользователя ОС в разделе.
- **$Upcase:** Используется для конвертации символов в нижнем регистре в соответствующие символы Unicode в верхнем регистре, что позволяет эффективно осуществлять поиск в файловой системе.

Для NTFS, вся информация организована в виде атрибутов заголовка и части данных. Даже системные файлы строятся на основе атрибутов. Разница между обычным файлом и/каталогом от системного файла состоит в названии файла и его функциональности. Системные файлы начинаются со знака ($). На рис. 3 представлена вырезка из соответствующей записи MFT. Каждая запись состоит из атрибутов. Перед выполнением каких-либо действий над файлом, первым делом, следует найти соответствующую ему запись в MFT. Затем будут получены атрибуты данной записи, и лишь после этого будут возможны какие-либо действия над этим файлом.

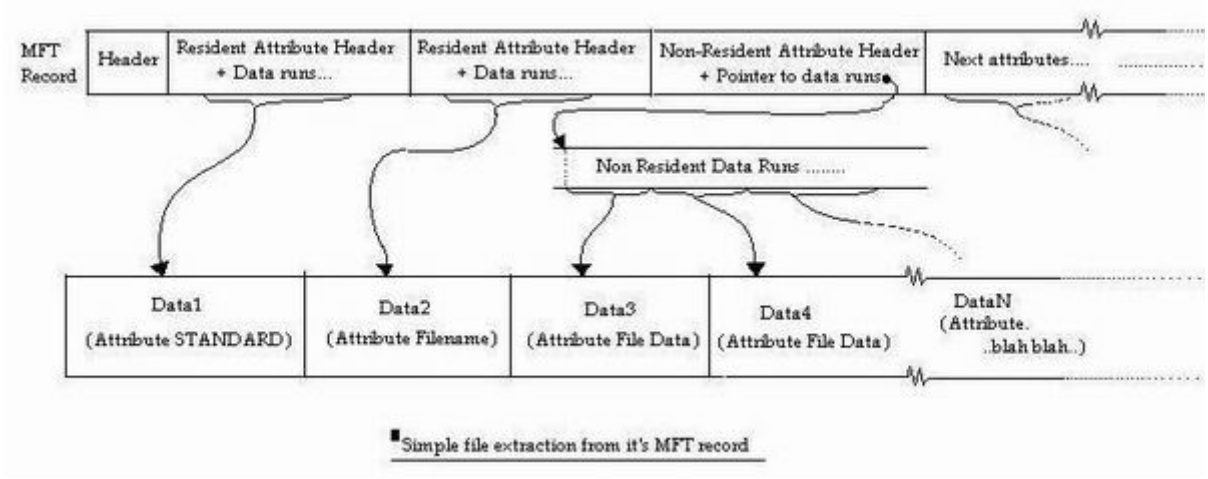

Рис. 3. Пример записи таблицы MFT

Любой атрибут в записи MFT принадлежит одной из категорий:

- **Resident attribute**: Эта категория содержит атрибуты, которые содержатся внутри самой записи. Когда файл очень маленький, он может быть размещен в саму запись. Более того, когда система читает эту запись, она также читает и данные. Для этого файла, ОС не требуется дополнительного обращения к диску для чтения записи.
- **Non-resident attribute**: Атрибуты размером превышающем размер записи MFT принадлежат к этой категории. Категория содержит актуальные данные атрибута, которые могут быть фрагментированы: размещены в разных частях диска. Заголовок атрибутов содержит указатель на начала размещения данных, а не сами данные. Однако после чтения данной записи, ОС следует снова обращаться к диску для получения данных.

На рис. 4 изображен метод получения данных из non-resident attributes посредством чтения данных.

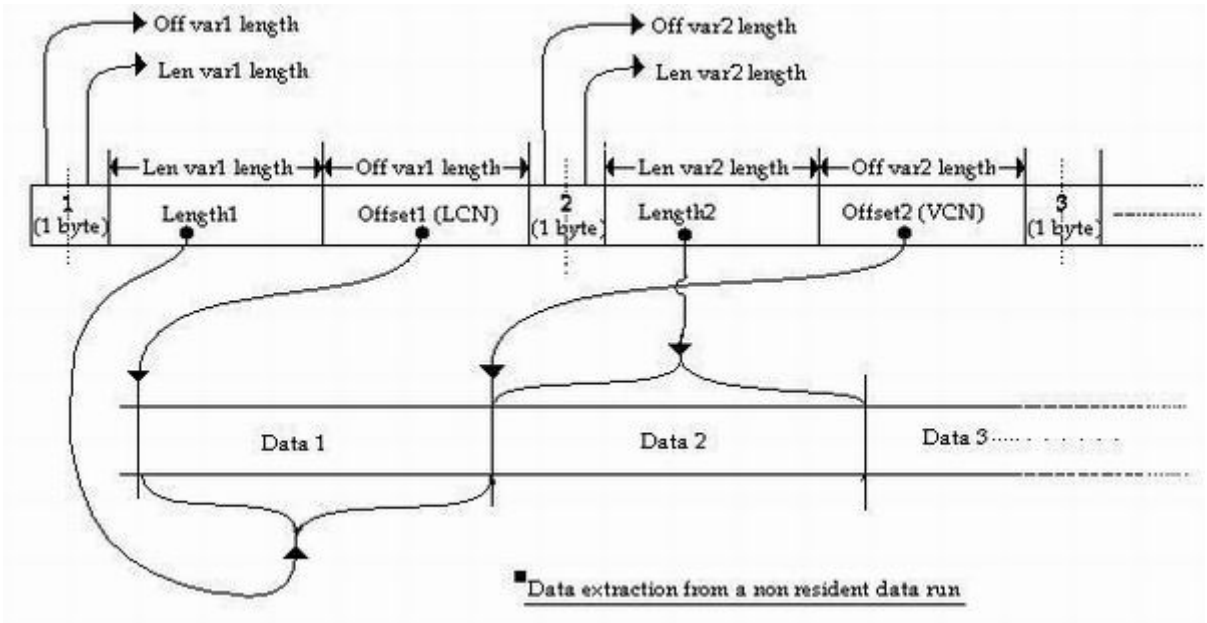

Рис. 4. Получение данных non-resident attributes

Типы атрибутов

Так, после чтения и вычленения данных с диска, их следует обработать в соответствии с типом атрибутов. Существует много доступных атрибутов, остановимся на описании лишь некоторых из них:

- **Standard Information attribute:** Этот атрибут содержит DOS File Permissions (file attributes) время файла. Windows 2000 вводит четыре новых поля: Quota, Security, File Size and Logging information.
- **Filename attribute:** Как и следует из названия, данный атрибут описывает имя файла, кроме того он описывает DOS File Permissions (file attributes), время файла, длинное/короткое имя в формате Unicode, выделенный размер и реальный размер файла. Этот атрибут повторяется для длинных и коротких имен файлов. Длинные имена файлов могут содержать до 255 символов Unicode. Короткие имена файлов (поддерживаются MS-DOS) могут содержать 8 + 3 символов, без учета регистра.
- **Security Descriptor attribute:** Содержит ACLs и SIDs для обеспечения безопасности. Показывает информацию о том, кто имеет право получить доступ к файлу, кто владеет файлом и т.д. Начиная с Window 2000 этот атрибут не используется. Начиная с Windows 2000, NTFS хранит всем дескрипторы безопасности stores в файле метаданных $Secure, разделяя дескрипторы между файлами и каталогами, что обладают одинаковые настройки. Предыдущие версии NTFS хранили информацию о дескрипторе безопасности с каждым файлом и каталогом.
- **Attribute List attribute:** Определяет расположение дополнительных MFT записей, в случае необходимости.
- **Bitmap attribute:** Этот атрибут файла является последовательностью битов, каждый из которых представляет статус вхождения. Обеспечивает отображение используемых записей MFT или каталога. Наиболее часто используется для больших каталогов.
- **Extended Attributes and Extended Attribute Information:** Расширенные атрибуты активно не используются, однако присутствуют для обратной совместимости с OS/2 приложениями.
- **Index root and Index allocation attribute:** Каталоги и файлы ничем не отличаются в NTFS. Если каталог достаточно небольшой, индекс к файлам, на который он указывает, может разместиться в MFT записи в атрибуте называемом **Index Root attribute**. Если присутствует достаточное число вхождений, NTFS будет создавать новое вхождение с non-resident атрибутом называемом **Index buffer**. В некоторых каталогах, атрибут index buffers содержит так называемые «b+ tree», которые являются структурой данных спроектированной для минимизации числа сравнений необходимых для поиска вхождения определенного файла. «b+ tree» хранят информацию (или индексы к информации) в отсортированном виде. С точки в каталоге, NTFS хранит отсортированные группы вхождений и указателей на вхождения, что произошли при сортировке. Это обладает большим преимуществами перед хранением вхождений в произвольном порядке их появления. Например, если вы хотите сортировать список вхождений в каталоге, ваш запрос будет удовлетворен быстро, потому как порядок хранения расположен в index buffer. Если вы хотите найти некоторую запись, поиск произойдет быстро, потому как, дерево располагает элементы «в ширину», а не «в глубину», что позволяет минимизировать число обращений для достижения необходимого узла дерева

Как восстанавливать файл после удаления?

Это процесс достаточно простой в NTFS, когда вы знаете структуру файла и путь для чтения ее. Когда вы получаете запись произвольного файла из MFT, в его заголовке, вы сможете увидеть поле флагов (`wFlags`). Если этот флаг установлен в единицу, это значит, что файл используется, иначе - удален. В удаленном файле этот бит игнорируется, если вы продолжите читать данные далее, вы сможете получить его содержимое. Вам необходимо создать новый файл и записать туда содержимое удаленного файла. Файл - восстановлен. Вам не нужен специальный механизм для чтения любых удаленных файлов, до тех пор, пока они не будут перезаписаны.

Разработка инструмента

Основную функциональность приложения составляют 2 класса:

- `CMFTRecord` класс: позволяет читать информацию о файле из MFT записи, а так же получить его атрибуты.
- `CNTFSDrive` класс: загружает таблицу MFT и манипулирует файлом в соответствии с запросами пользователя.

Ниже, на рис. 5 представлена диаграмма классов.

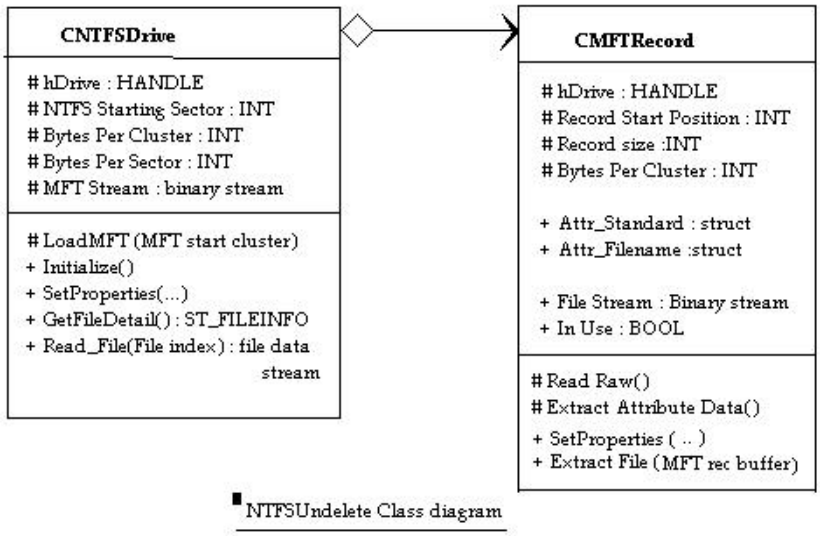

Рис. 5. Диаграмма классов

На рис. 6 представлено взаимодействие методов класса `CNTFSDrive` с методами и данными класса `CMFTRecord` class для получения файла.

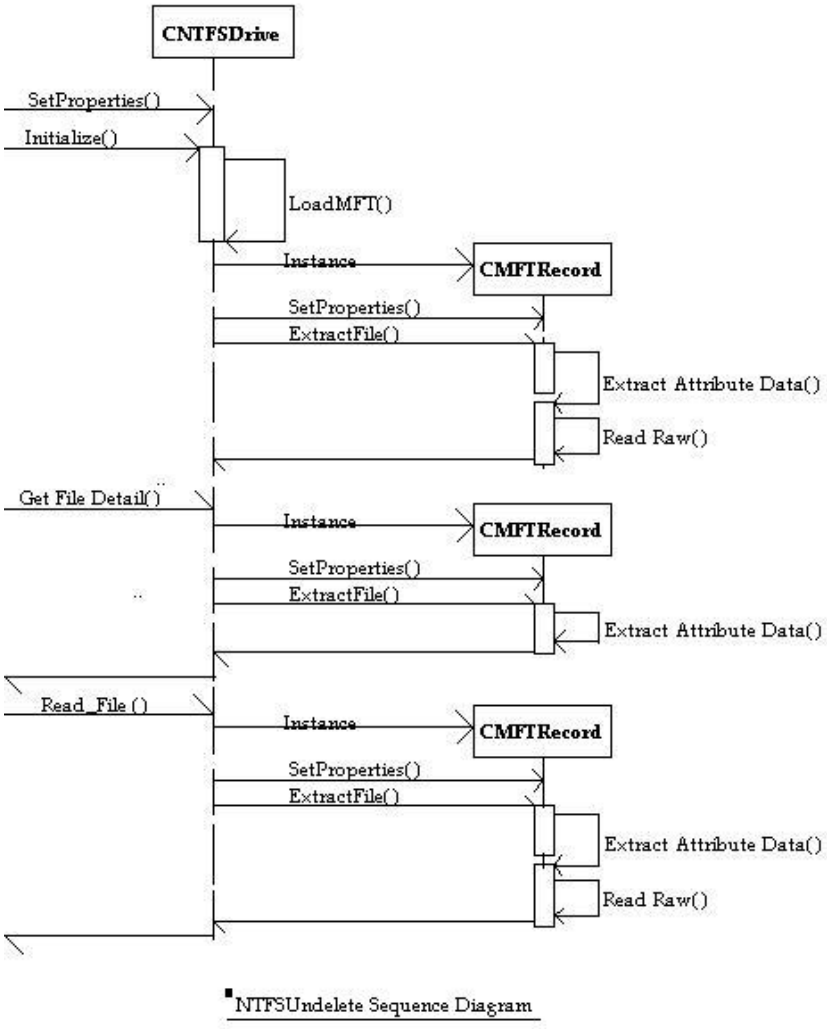

Рис. 6. Диаграмма последовательностей

Использование класса CNTFSDrive

Данный класс следует проинициализировать начальным сектором NTFS диска (его файловой системы), другими словами загрузочным сектором NTFS (если он таковым обладает). Незначительная часть приложения позаимствована из [2], которая отвечает за чтение таблицы разделов, начального сектора.

Детально остановимся на использовании класса CNTFSDrive для получения файла из NTFS:

```
CNTFSDrive::ST_FILEINFO stFInfo;
BYTE *pData;
DWORD dwBytes =0;
DWORD dwLen;

m_hDrive = CreateFile("\\\\.\\PhysicalDrive0", GENERIC_READ,
            FILE_SHARE_READ|FILE_SHARE_WRITE, 0,
            OPEN_EXISTING, 0, NULL);

m_cNTFS.SetDriveHandle(pDlg->m_hDrive); // set the physical drive handle

m_cNTFS.SetStartSector(StartingRelativeSector,512); // set the starting
sector of the NTFS


nRet = m_cNTFS.Initialize(); // initialize, ie. read all MFT in to the memory

if(nRet)
{
 ... error...
}

nRet = m_cNTFS.GetFileDetail(100,stFInfo); // get the file detail one by one

if(nRet)
{
 ... error...
}

...

stFInfo,szFilename;    // file name
stFInfo,n64Create;     // Creation time
stFInfo,n64Modify;     // Last Modify time
stFInfo,n64Modfil;     // Last modify of record
stFInfo,n64Access;     // Last Access time
stFInfo,dwAttributes;  // file attribute, eg. Hidden, Archive,
                       // System, ReadOnly, Compressed , etc
stFInfo,n64Size;       // no of clusters used
stFInfo,bDeleted;      // if true then its deleted file

...

// 100 is the file sequence number
nRet = m_cNTFS.Read_File(100,pData,dwLen); // read the file content in to a
buffer

if(nRet)
{
 ...error...
}

// create the same file in a new location

HANDLE hNewFile = CreateFile(stFInfo,szFilename, GENERIC_WRITE,
                0, NULL, CREATE_ALWAYS, stFInfo.dwAttributes, 0);

// save the file data on to the new file
nRet = WriteFile(hNewFile,pData,dwLen,&dwBytes,NULL);
if(!nRet)
{
 ... error...
}

// close the handles ..

CloseHandle(hNewFile);
```

```
CloseHandle(m_hDrive);

// that's it you have extracted a file from the NTFS

...
...
```

Алгоритм

Исходя из архитектуры приложения, опишем алгоритм приложения для восстановления удаленных файлов.

1. Анализ корректности параметров командной строки.

2. В зависимости от режима работы приложения, рассмотрим несколько вариантов дальнейшего развития.

2.1. Получение списка разделов (томов).

2.1.1. С использованием функции Win32 API анализируются все физические устройства.

2.1.2. Производится чтение таблицы разделов физического диска.

2.1.3. Анализируются записи таблицы разделов физического диска и выводится информация о них.

2.2. Получение списка записей MFT (файлов), на заданном NTFS разделе.

2.2.1. Производится чтение загрузочной записи заданного NTFS раздела и проверяется, действительно ли заданный раздел (том) является NTFS.

2.2.2. Производится чтение MFT таблицы, и она целиком загружается в память.

2.2.3. Содержимое MFT таблицы анализируется и выводится в заданный файл отчета, с указанием: номера записи, полного его пути, имени файла, его атрибутов, физического размера, реально выделенного количества кластеров.

2.3. Восстановление заданной MFT записи (файла), другими словами создание нового файла, содержащего данные восстановленного файла.

2.3.1. Производится чтение загрузочной записи заданного NTFS раздела и проверяется, действительно ли заданный раздел (том) является NTFS.

2.3.2. Производится чтение таблицы MFT.

2.3.3. Производится поиск заданной записи таблицы MFT, для дальнейшего создания ее дубликата.

Литература

1. Microsoft Developer Network. URL: http://www.msdn.com

2. Vinoj Kumar. Forensic Inspection of Hard Disk. URL: http://codeguru.com/system/Forensic.html

3. Ramanan T. Undelete a file in NTFS. URL: http://www.codeproject.com/KB/files/NTFSUndelete.aspx

4. NTFS – Home. URL: http://linux-ntfs.sourceforge.net/ntfs/

5. NTFS. URL: http://www.ntfs.com/

6. Кэрриэ Б. Криминалистический анализ файловых систем. URL: http://nrjetix.com/r-and-d/lectures-os

7. MSDN. XML Documentation Comments.

8. Формирование документации к исходному коду с помощью средства doxygen. URL: www.nrjetix.com/r-and-d/lectures