УДК 004.622

ФАЙЛОВАЯ СИСТЕМА NTFS

Якимович С.В.
Научный руководитель – Разоренов Н.А., к.т.н., доцент

Загрузочная запись тома под NTFS (BOOT), содержит основную информацию о томе (логическом диске), такую как расположение MFT, количество секторов на кластер, всего секторов на томе, код загрузчика (NT Loader) и т. д.

Для более удобной работы была объявлена структура (Рис. 1).

```
struct BootRecordNTFS
{
 BYTE dJmp[3];
 CCHAR dSystemID[8];
 struct {
      WORD dBytesPerSector;
      BYTE dSectorsPerCluster;
      WORD ReservedSectors;
      BYTE dUnusedA[5];
      BYTE dMediaID;
      BYTE dUnusedB[2];
      WORD dSectorPerTrack;
      WORD dNumberOfHeads;
      DWORD dHiddenSectors;
      BYTE dUnusedC[4];
      BYTE Signature[4];
      INT64 dTotalSectors;
      INT64 dLCNofMFT;
      INT64 dLCNofMFTMirr;
      DWORD ClusterPerMFT;
      DWORD ClustersPerIndexes;
 }BiosParameterBlock;
 BYTE SerialNumber[8];
 DWORD Checksum;
 BYTE DataCode[426];
 BYTE Signature55AA[2];
};
```

Рисунок 1. – Структура BOOT

Логический диск открывается как файл с помощь функции CreateFile, данные считываются в BootRecordNTFS, делается проверка на количество байт на сектор и имя файловой системы, чтобы программа не пыталась работать с файловой системой, которую она не поддерживает; содержимое структуры представляется пользователю.

Каждый файл на томе NTFS представлен записью в специальном файле, называемом главной файловой таблицей (MFA – master file table). NTFS резервирует первые 16 записей таблицы для специальной информации. Первая запись этой таблицы описывает непосредственно главную файловую таблицу. За ней следует зеркальная запись (mirror record) MFT.

Если первая запись MFT повреждена, то NTFS читает вторую запись для поиска зеркального файла MFT, первая запись которого идентична первой записи MFT. Местоположения сегментов данных MFT и зеркального файла MFT записаны в секторе начальной загрузки.

Каждая запись в MFT начинается с заголовка, за которым следует набор атрибутов.

Для считывания файловой записи используются данные структуры BootRecordNTFS: количество байт на сектор, количество секторов на кластер, номер первого кластера MFT, с помощью которых определяется смещение. С данным смещением с диска считываются запись. Для определения значений параметров заголовка используется структура заголовка файловой записи (Рис 2).

```
struct FileRecordHeader {
    CCHAR SignatureFILE[4];
    WORD OffsetToTheUpdateSequence;
    WORD UpdateSequenceSizeInWords;
    UINT64 LogFileSequenceNumberLSN;
    WORD Sequencenumber;
    WORD HardLinkCount;
    WORD OffsetToTheFirstAttribute;
    WORD Flags;
    DWORD RealSizeOfTheFILErecord;
    DWORD AllocatedSizeOfTheFILErecord;
    UINT64 BaseFILErecord;
    WORD NextattributeID;
    WORD UnusedB;
    DWORD IDOfThisRecord;
    BYTE UpdateSequenceNumber[2];
    BYTE UpdateSequenceArray[4];
    WORD UnusedC;
};
```

Рисунок 2. – Структура заголовка файловой записи

Пользователю предоставляются значения полей и расшифрованные значения флагов. Используя поле структуры, содержащее смешение к первому атрибуту переходим к считыванию значения заголовка атрибута.

Каждый атрибут имеет заголовок, структура которого зависит от того, резидентный атрибут или нет, а также имеет ли он имя. Каждый заголовок атрибута, как и сам атрибут, имеет определенную структуру (Рис. 3-4).

```
struct AttributeHeader {
 DWORD AttributeType;
 DWORD LengthIncludingHeader;
 BYTE NonResidentFlag;
 BYTE NameLength;
 WORD NameOffset;
 WORD Flags;
 WORD AttributeID;
 DWORD LengthOfTheAttribute;
 WORD OffsetToTheAttributeData;
 BYTE IndexedFlag;
 BYTE Padding;
};
```

Рисунок 3. – Структура заголовка резидентного атрибута

```
struct AttributeHeaderNonResident {
 DWORD AttributeType;
 DWORD LengthIncludingHeader;
 BYTE NonResidentFlag;
 BYTE NameLength;
 WORD NameOffset;
 WORD Flags;
 WORD AttributeID;
 UINT64 FirstVCN;
 UINT64 LastVCN;
 WORD DataRunsOffset;
 WORD CompressionUnitSize;
 BYTE Padding[4];
 UINT64 AllocatedSize;
 UINT64 RealSize;
 UINT64 InitializedSize;
};
```

Рисунок 4. – Структура заголовка нерезидентного атрибута

После определения типа атрибута, используется соответствующая структура для его заголовка. Эта структура содержит смещение к телу атрибута. Каждый атрибут идентифицирован кодом типа атрибута и необязательно именем атрибута. Если данных в файле не много, то они хранятся в записи файла MFT и его содержимое может интерпретироваться в зависимости от типа атрибута.

Если данные файла не помещаются в одну запись MFT, то этот факт отражается в заголовке атрибута, который содержит признак того, что этот атрибут является нерезидентным, то есть находится в отрезках вне таблицы MFT. В этом случае тело атрибута содержит адресную информацию каждого отрезка данных.

В записи первого фрагмента первый полубайт содержит информацию о количестве байт, выделенных для хранения номера первого кластера. Второй полубайт содержит количество байт, выделенных для хранения количества кластеров. Далее располагаются само значение количества кластеров и номер кластера. Если файл фрагментирован, то дальнейшие записи о фрагментах файла имеют такую же структуру, но они содержать не номер кластера, а смещение относительно предыдущего фрагмента. Значение ноль в обоих полубайтах означает конец цепочки фрагментов. Считывание RunList (bytePointer – указатель на массив байт, DataRunOffset – поле структуры заголовка, содержащее смещение к RunList) изображено на Рис. 5.

```
PBYTE pointer = bytePointer + attributeHeader.DataRunsOffset;
INT64 firstCluserPrev = 0;
BYTE size = *pointer;
while (size != 0) {
  BYTE ByteCountPerClusterCount = size & 0xf;
  BYTE ByteCountPerFirstCluser = (size >> 4) & 0xf;
  pointer++;
  INT64 cluserCount = 0;
  INT64 temp = 0;
  int i = 0;
  while (i < ByteCountPerClusterCount) {
    temp = *pointer;
    temp <<= 8 * i;
    cluserCount += temp;
    pointer += 1;
    i++;
  }
  INT64 OffsetFromPrev = 0;
  i = 0;
  while (i < ByteCountPerFirstCluser) {
    temp = *pointer;
    temp <<= 8 * i;
    OffsetFromPrev += temp;
    pointer++;
    i++;
  }
  OffsetFromPrev <<= (8 * (8 - ByteCountPerFirstCluser));
  OffsetFromPrev >>= (8 * (8 - ByteCountPerFirstCluser));
  INT64 firstCluster = firstCluserPrev + OffsetFromPrev;
  firstcluserPrev = firstCluster;
  //сохранение cluserCount, firstCluster
 }
```

Рисунок 5. – Фрагмент кода выполняющий расчет номеров кластеров файла

Первые 16 файлов NTFS (метафайлы) носят служебный характер. Каждый из них отвечает за какой-либо аспект работы системы. Метафайлы находятся в корневом каталоге NTFS диска – их имена начинаются с символа «$».

$MFT – основная таблица MFT

$MFTmirr – копия первых шестнадцати записей MFT

$LogFile – журнал файловой системы

$Volume – служебная информация (метка и ID тома, версия файловой системы)

$AttrDef – список стандартных атрибутов файлов на томе

. – корневой каталог

$Bitmap – карта свободного места тома

$Boot – загрузчик (только на первичном томе)

$BadClus – повреждённые кластера

$Quota – записи с правами пользователей на использование дискового пространства (квотами)

$Secure – дескрипторы безопасности файловых объектов (права доступа)

$UpCase – файл соответствия строчных и прописных букв

Литература

1. Операционные системы и системное программирование: методические указания к лабораторным работам для студентов специальностей 1-40 01 01 «Программное обеспечение информационных технологий» и 1-40 01 02 «Информационные системы и технологий» / сост.: Н.А. Разоренов. – Минск: БНТУ, 2011. – 94 с.