Рубрика 1. Информатика и информационные процессы

***УДК** 004.63*

Д. Д. Новикова[1], С. С. Чурсина[1]

[1]ФГАОУ ВО «РГУ нефти и газа (НИУ) имени И. М. Губкина», г. Москва, Российская Федерация

АНАЛИЗ СТРУКТУР HFS+ И МЕТОДИКА ВОССТАНОВЛЕНИЯ ДАННЫХ

Аннотация. В статье представлен структурный анализ файловой системы HFS+ и описана методика ручного восстановления данных на основе прямого изучения служебных областей тома. Актуальность работы обусловлена тем, что HFS+ продолжает использоваться на внешних накопителях, резервных копиях и устаревших устройствах Apple, а автоматические средства восстановления не всегда корректно обрабатывают повреждённые метаданные. Рассматриваются назначение Volume Header, параметры Catalog File, устройство B-дерева каталогов, структура листовых узлов, массивы смещений записей, ключи HFSPlusCatalogKey и основные поля записей файлов и папок. На практическом примере показано, как определить размер блока, вычислить физическое смещение каталогового файла, найти записи корневого каталога, установить идентификаторы объектов и извлечь содержимое файла по данным экстентов. Методика ориентирована на работу в hex-редакторе и не требует применения дорогостоящего специализированного программного обеспечения. Особое внимание уделено ограничениям ручного анализа: высокой трудоёмкости, необходимости точного расчёта смещений и риску ошибок при интерпретации записей. Поэтому предложенный подход рассматривается не как массовая замена автоматизированным утилитам, а как экспертный инструмент для проверки результатов и восстановления отдельных важных объектов. Полученные результаты демонстрируют, что ручной низкоуровневый анализ позволяет восстанавливать отдельные критически важные файлы, проверять корректность автоматизированных утилит и использовать HFS+ как учебную модель для освоения принципов организации файловых систем. Предложенный подход полезен специалистам по восстановлению данных, цифровой криминалистике и системному администрированию.

Ключевые слова: HFS+, восстановление данных, структурный анализ, B-дерево, каталоговый файл, экстенты, файловая система Apple.

Введение

HFS+ (Hierarchical File System Plus), также известная под названиями Mac OS Extended и HFS Extended, представляет собой основную журналируемую файловую систему, применяемую в операционных системах семейства Mac OS X с момента выхода Mac 8.1 в 1998 году. В современных версиях macOS данная файловая система стандартно идентифицируется как Mac OS Extended (Journaled) [1]. Хотя в 2017 году Apple перешла на оптимизированную для SSD систему APFS (Apple File System), заменившую HFS+ в macOS High Sierra [2], последняя сохраняет популярность как формат дисков для Mac в силу своей высокой совместимости. Несмотря на переход на APFS, файловая система HFS+ продолжает встречаться на практике — в первую очередь на внешних накопителях, используемых для резервного копирования (например, Time Machine), а также на устаревших устройствах Apple. В задачах цифровой криминалистики и аварийного восстановления данных часто требуется работа с такими legacy-томами, особенно когда повреждены метаданные или автоматические инструменты не могут корректно интерпретировать структуру раздела. В этих случаях глубокое понимание внутреннего устройства HFS+ становится критически важным. Архитектура HFS+ предполагает хранение числовых параметров в формате big endian, что необходимо учитывать при обработке служебных структур. Архитектура HFS+ подробно описана в официальной технической документации Apple (Technical Note TN1150), а её особенности в контексте цифровой криминалистики — в работах Брайана Кэрриэра и других исследователей.

Эта файловая система стала более усовершенствованной по сравнению со своей предшественницей, HFS. Ключевые улучшения включают переход на 32-битную адресацию блоков, поддержку Unicode-имён файлов, работу с томами значительно большего размера, а также внедрение эффективных механизмов хранения данных и ведения журнала. Благодаря новой архитектуре HFS+ обеспечивает работу с томами и файлами размером до 8 эксабайт (2^{63} байт при двоичном понимании единиц) и может содержать до 2,1 млрд файлов в рамках одного тома, обеспечивая необходимую масштабируемость для современных задач.

Основу функционирования HFS+ составляют шесть структур метаданных. Заголовок тома (Volume Header), расположенный в начале раздела, содержит основополагающую

информацию о томе, такую как дата создания, размеры блоков и указатели на другие служебные файлы. Статус блоков распределения (свободен/занят) отслеживается с помощью файла распределения (Allocation File), который является простой битовой картой. Сердцем файловой системы является файл каталога (Catalog File) – большое B-дерево, содержащее записи для всех файлов и папок и образующее иерархическую структуру тома. Именно в этих записях хранятся первые восемь экстентов, описывающих расположение данных файлов. Если файл сильно фрагментирован, дополнительные экстенты сохраняются в файле переполнения экстентов (Extents Overflow File). Дополнительные метаданные могут храниться в файле атрибутов (Attribute File), а загрузочный файл (Startup File) содержит код, необходимый для загрузки системы с устаревших компьютеров. Ниже, на рис. 1 приведена схема тома HFS+ [3].

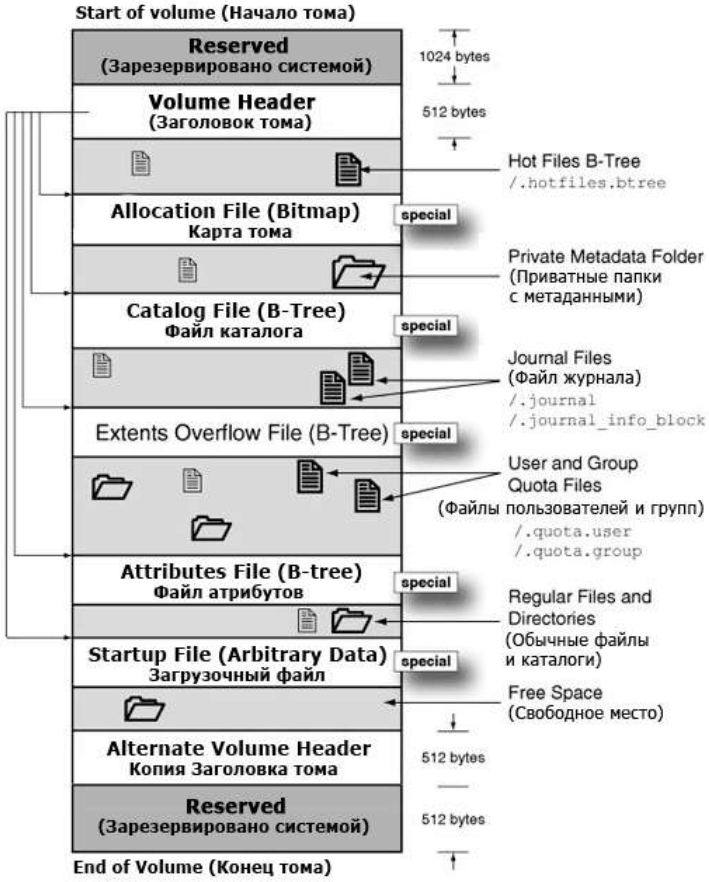

Рис. 1. Структура тома с HFS+

Помимо этих структур, в файловой системе существует множество вспомогательных, но именно перечисленные выше имеют наибольшее значение при анализе и восстановлении данных.

Заголовок Тома (Volume Header)

Volume Header размещается во втором секторе тома (offset – 0x400) и содержит все ключевые параметры файловой системы. Резервная копия заголовка располагается симметрично – во втором секторе от конца тома, обеспечивая восстановление системы при повреждениях. Ниже (табл. 1) приведены наиболее важные параметры Volume Header для анализа и восстановления данных [3].

Таблица 1 – Основные атрибуты Volume Header

Смещение	Размер (в байтах)	Название	Описание
0x000	2	signature	Подтверждение, что это HFS+ (H+ = 0x482B)
0x00C	4	journalInfoBlock	Если ≠ 0 - том журналируемый
0x010	4	createDate	Когда создан том. В секундах с 1 января 1904 года по Гринвичу (GMT). Для московского времени прибавляем 3 часа.

0x020	4	fileCount	Количество файлов на томе
0x024	4	folderCount	Количество папок на томе
0x028	4	blockSize	Размер блока
0x02C	4	totalBlocks	Количество блоков на томе
0x110	80	catalogFile	Данные Catalog B-tree

Поле catalogFile в Volume Header описывает расположение каталогового файла на диске. Это структура HFSPlusForkData размером 80 байт:

1. 0x00 (8 байт) – logicalSize (логический размер)
2. 0x08 (4 байта) – clumpSize (размер кластера)
3. 0x0C (4 байта) – totalBlocks (всего блоков)
4. 0x10 (64 байта) – extents (массив из восьми экстентов)

Массив экстентов в структуре catalogFile состоит из восьми элементов размером 8 байт каждый. Каждый элемент содержит два параметра: номер начального блока startBlock (4 байта) и количество последовательных блоков blockCount (4 байта). Данные Catalog File физически расположены в блоках, описываемых этими экстентами. Для определения физического смещения Catalog File используется формула:

$$catalogFileOffs = startBlock \times blockSize \quad (1)$$

где значение blockSize берётся из заголовка тома (Volume Header).

На рис. 2 продемонстрирован фрагмент тома в среде шестнадцатеричного редактора HxD, на котором обозначены параметры Volume Header из таблицы 1.

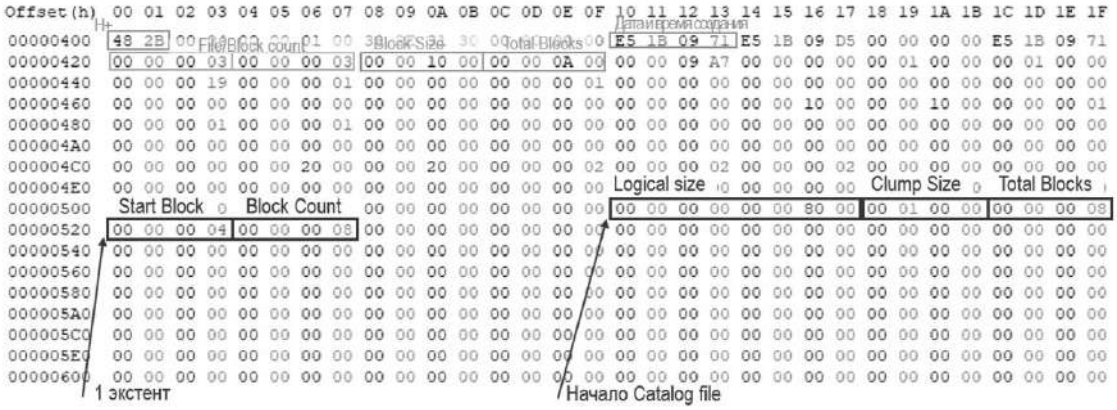

Рис. 2. Разбор основных атрибутов Volume Header

Анализ структуры catalogFile начинается со смещения 0x110 от начала Volume Header. Параметр logicalSize указывает на размер 32 768 байт (0x8000). Первый экстент содержит значения Start block = 4 и Block count = 8, что определяет расположение данных каталогового файла в виде последовательности из восьми блоков, начиная с блока №4. Физическое смещение рассчитывается по формуле:

$$catalogFileOffs = 4\ (startBlock) \times 4096\ (blockSize) = 16384\ байт = 0x4000$$

Заголовок B-дерева (B-Tree Header Node)

B-Tree Header Node – это служебная структура, расположенная в начале файла B-дерева (Catalog, Extents Overflow или Attributes File). Он содержит метаинформацию о всей структуре B-дерева, необходимую для навигации по ней. В таблице 2 выборочно представлены параметры заголовка узла B-дерева.

Таблица 2 – Атрибуты B-Tree Header Node

Смещение	Размер (в байтах)	Название	Описание
0x08	1	kind	Тип узла: –1 = leaf, 0 = index, 1 = header, 2 = map
0x0A	2	numRecords	Количество записей в узле

0x10	4	rootNode	Номер корневого узла
0x14	4	leafRecords	Количество записей в листовых узлах
0x18	4	firstLeafNode	Первый листовой узел
0x20	2	nodeSize	Размер узла

Формула для расчёта смещения листового узла:

$$leafNodeOffs = catalogFileOffs + firstLeafNode \times nodeSize \quad (2)$$

На рис. 3 визуализированы параметры заголовка B-дерева файла, найденного на предыдущем шаге. Видно, что rootNode и firstLeafNode = 1, nodeSize = 0x1000 (4096 байт).

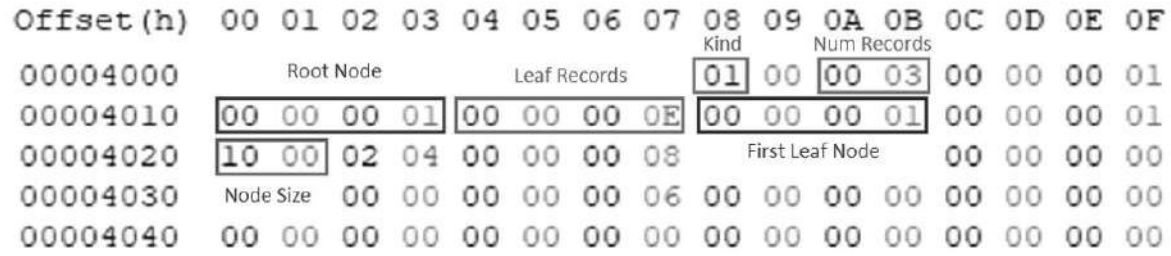

Рис. 3. Разбор основных атрибутов B-Tree Header Node

Это означает, что корневой узел (rootNode) одновременно является и первым листовым узлом (firstLeafNode). В B-деревьях это возможно, когда дерево очень маленькое (например, содержит всего несколько файлов и папок) и вся информация помещается в один-единственный узел. В данном случае структура B-дерева не требует навигации через индексные узлы – все элементы метаданных содержатся в узле №1. Смещение листового узла определяется по формуле:

$$leafNodeOffs = 0x4000 + (1 \times 0x1000) = 0x5000$$

Листовый узел (Leaf Node)

При анализе листового узла (рис. 4) проверяется значение типа узла (kind) по смещению 0x08, которое должно составлять 0xFF (Leaf Node). Затем по смещению 0x0A извлекается значение numRecords, указывающее количество записей в узле – в данном случае 0xE [4].

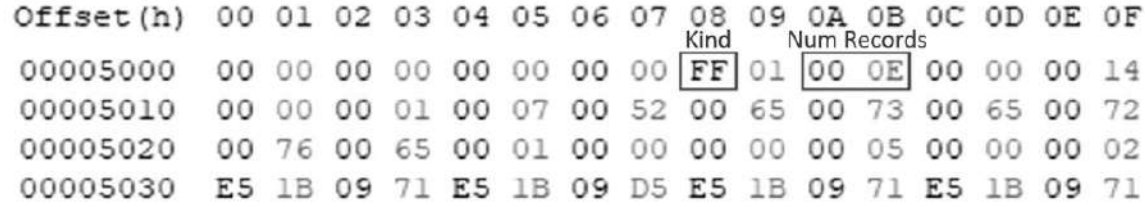

Рис. 4. Основные атрибуты в листовом узле для поиска смещений

В конце узла есть массив смещений на каждую запись. Каждое смещение – число размером 2 байта (само смещение во втором байте пары), которое показывает, насколько нужно отступить от начала узла, чтобы найти запись. Смещения идут в обратном порядке, то есть последние 2 байта узла соответствуют смещению первой записи, предпоследние 2 байта — смещению второй записи и т. д. Первая запись всегда начинается со смещения 14, поэтому последние 2 байта узла обычно содержат число 0x0E.

Для определения массива смещений записей используется следующее вычисление: при количестве записей N = 14 (0xE) и размере узла 0x1000 байт, расчёт конечного смещения массива выполняется по формуле:

$$recOffs = leafNodeOffs + nodeSize - N \times 2 \quad (3)$$

Расчёт:

$$recOffs = 0x5000 + 0x1000 - (0xE \times 2) = 0x5FE4$$

По полученному адресу (рис. 5) располагается массив двухбайтовых смещений, указывающих начала записей в узле.

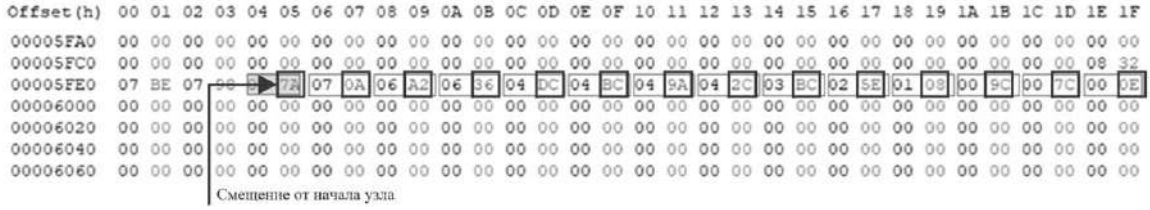

Рис. 5. Все записи в массиве смещений

Алгоритм обработки записей листового узла предполагает последовательное чтение значений из массива смещений recOffs. Каждый элемент recOffs[N] содержит относительное смещение N-ой записи от начала узла. Для доступа к N-ой записи используется формула:

$$startRecOffs = leafNodeOffs + recOffs[N] \quad (4)$$

HFSPlusCatalogKey

Ключ каталога используется для идентификации и поиска записей в B-дереве каталогов (табл. 3):

Таблица 3 – Атрибуты HFSPlusCatalogKey

Смещение	Размер (в байтах)	Название	Описание
0x00	2	keyLength	Длина данных ключа (parentID + nodeName)
0x02	4	parentID	Идентификатор родительской папки (CNID)
0x06	Переменный	nodeName	Имя файла/папки в Unicode

Структура HFSUniStr255, которая поясняет поле nodeName (табл. 4):

Таблица 4 – Атрибуты HFSUniStr255

Смещение	Размер (в байтах)	Название	Описание
0x00	2	length	Количество символов в имени (N)
0x02	N × 2	unicode	Символы UTF-16BE

Файловая система HFS+ использует систему численных идентификаторов (CNID) для служебных и пользовательских объектов. Зарезервированные диапазоны включают идентификаторы с 1 по 8 для важных компонентов (например, CNID 2 для корневой папки, CNID 4 для каталогового файла. Тогда как значения от 16 и выше закрепляются за пользовательскими файлами и каталогами. Данное разделение позволяет однозначно определять тип объекта в процессе восстановления данных.

Для восстановления структуры каталогов необходимо найти все записи, содержащие в поле parentID значение 0x00000002, что соответствует корневому каталогу тома (рис. 6).

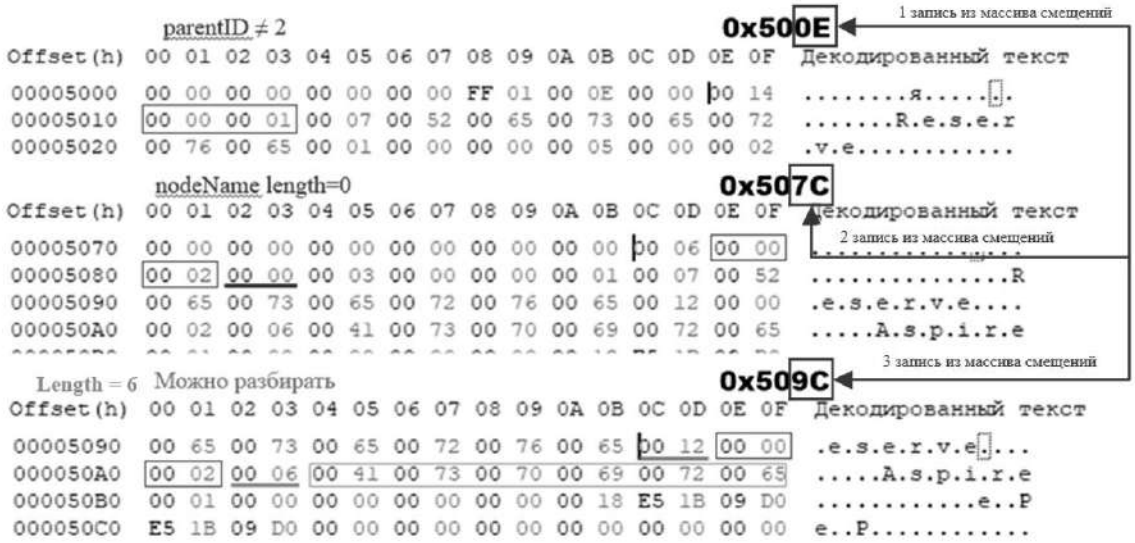

Рис. 6. Разбор записей в листовом узле

Первая запись в узле располагается по смещению 0xE, следовательно, адрес начала записи равен 0x5000 + 0xE = 0x500E.

Поле parentID имеет значение 0x00000001, что соответствует идентификатору родителя корневой папки. Таким образом, данная запись описывает корневую папку тома, имя которой является меткой тома. На данном этапе нашли метку тома (nodeName) – Reserve.

Примечание. Значения не противоречат друг другу: parentID = 0x00000001 относится к записи самого корневого каталога, тогда как parentID = 0x00000002 используется у дочерних записей, непосредственно размещённых в корневом каталоге.

Поле parentID (CNID) этой записи равно 0x00000002, что указывает на корневой каталог (kHFSRootFolderID), структура которого далее будет восстанавливаться [3].

Переходим ко второй записи 0x5000 + 7C = 0x507C. ParentID = 2, но длина имени nodeName = 0. Пустая запись пропускается. Адрес 3 записи: 0x5000 + 9C = 0x509C. Длина имени ненулевая, parentID = 0x2 (корень) – разбираем.

keyLength – 00 12 (18_{10})

parentID – 00 00 00 02

nodeName:

length – 00 06

Само имя 6 * 2 = 12 байт – 00 41 00 73 00 70 00 69 00 72 00 65 (Aspire)

Catalog File/Folder Records

После ключа (HFSPlusCatalogKey) в листовом узле располагается тело записи – область, содержащая информацию о файле или папке [5].

Тип записи определяется полем recordType (2 байта) и описывается одной из структур: HFSPlusCatalogFile – для файлов; HFSPlusCatalogFolder – для папок. Эти структуры содержат основные атрибуты объектов файловой системы: уникальный идентификатор (CNID), даты создания и изменения, права доступа, а также указатели на данные (экстенты для файлов).

Типы записей (recordType):

1. 0x0001 – HFSPlusFolderRecord (запись о папке)
2. 0x0002 – HFSPlusFileRecord (запись о файле)
3. 0x0003 – HFSPlusFolderThread («поток» папки, то есть служебная ссылка на родителя)
4. 0x0004 – HFSPlusFileThread («поток» файла, то есть служебная ссылка на родителя)

В таблице 5 смещения представлены относительно начала тела записи.

Таблица 5 – Структура записи папки (HFSPlusCatalogFolder)

Смещение	Размер (в байтах)	Поле	Описание
0x0000	2	recordType	Тип записи (0x0001)
0x0004	4	valence	Количество элементов в папке
0x0008	4	folderID	CNID папки
0x000C	4	createDate	Дата создания

Поле folderID используется для построения иерархии – на него ссылаются записи, где parentID = folderID. Поле valence позволяет сверить, сколько элементов должно содержаться в папке (рис.7).

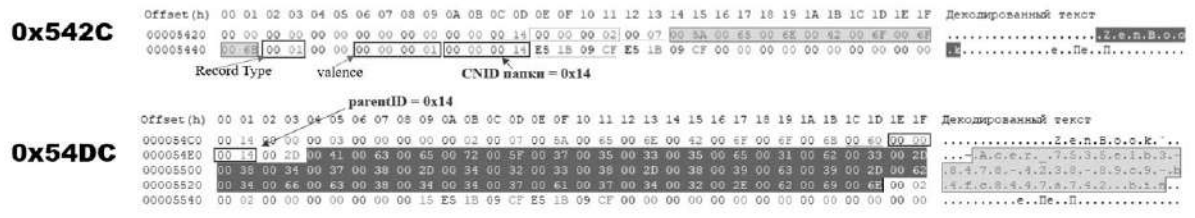

Рис. 7. Разбор записи папки

По адресу 0x5000 + 0x042C = 0x542C видим, что у нас папка (recordType = 1), смотрим ее CNID = 0x14. Теперь мы перебираем оставшиеся смещения и записи. Наша задача – найти значение parentID = 0x14. Нашли одну запись по смещению 0x54DC, которая является файлом. Аналогичным образом ищется содержимое остальных каталогов.

Восстановление файлов

Для извлечения содержимого файлов требуется анализ соответствующих записей в каталоговом B-дереве. Структура HFSPlusCatalogFile содержит всю необходимую метаинформацию (табл. 6), включая указатель на данные файла.

Таблица 6 – Структура записи файла (HFSPlusCatalogFile)

Смещение	Размер (в байтах)	Поле	Описание
0x0000	2	recordType	Тип записи (0x0002)
0x0008	4	fileID	CNID файла
0x000C	4	createDate	Дата создания
0x0058	80	dataFork	Данные data fork (содержимое файла) logicalSize (8 байт) содержит фактический размер файла в байтах

Поле fileID используется для идентификации записи, а dataFork – для поиска реальных данных файла на диске. Экстенты (HFSPlusExtentDescriptor) в dataFork указывают начальный блок и длину данных (рис.8).

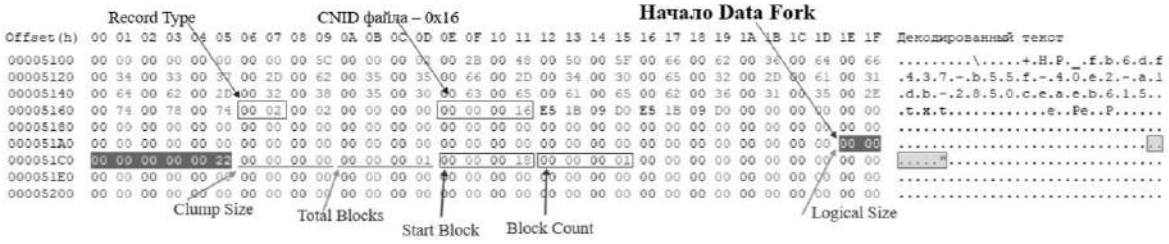

Рис. 8. Разбор записи файла

В данном случае по смещению 0x5000 + 0x0108 = 0x5108 находится HP_fb6df437-b55f-40e2-a1db-2850ceaeb615.txt. Так как recordType = 2, мы понимаем, что это – файл. Продолжаем разбор атрибутов и находим его startBlock = 0x18 и blockCount = 1. Содержимое файла находится по адресу 0x1000 (blockSize) * 0x18 = 0x18000. Значение logicalSize = 0x22 (34 байта) указывает, что реальная длина содержимого файла – 34 байта несмотря на то, что ему выделен один блок размером 4096 байт (blockCount = 1).

Файловая система HFS+ не обнуляет содержимое блоков при записи, особенно в случаях, когда файл был удалён или перезаписан. Поэтому только первые 0x22 байта блока содержат актуальные данные, соответствующие значению logicalSize (34 байта), а оставшееся пространство может включать остаточные данные от ранее записанных файлов, не имеющие отношения к текущему содержимому.

Переходим к рассчитанному смещению 0x18000, копируем 0x22 байта информации и сохраняем в новом файле с соответствующим расширением .txt (рис.9).

Рис. 9. Восстановление содержимого файла

Проведённое исследование показало, что структурный анализ файловой системы HFS+ может быть эффективно применён для решения практических задач по восстановлению данных. Предложенная методика основана на пошаговом изучении ключевых метаданных – от Volume Header до листовых узлов B-дерева каталогов – и позволяет извлекать пользовательские файлы даже без использования специализированного программного обеспечения. Важно подчеркнуть, что предложенная методика не предназначена для массового восстановления больших объёмов данных. Ручной анализ в hex-редакторе — трудоёмкий и подверженный ошибкам процесс. Однако он оказывается незаменимым в ряде узких сценариев: при восстановлении отдельных критически важных файлов, при обучении принципам низкоуровневого анализа файловых систем, а также для верификации результатов, полученных автоматизированными утилитами. Таким образом, методика представляет собой скорее инструмент эксперта, чем повседневное решение для системного администратора.

Существуют открытые инструменты, такие как TestDisk, The Sleuth Kit и утилита hfsinspect, которые автоматизируют анализ и восстановление HFS+-томов [6, 7]. Однако при серьёзных повреждениях — например, полной потере заголовка тома, некорректной таблицы разделов или сбое журналирования — автоматика часто «сдаётся», не находя даже базовых метаданных. В таких ситуациях ручной анализ, основанный на прямом чтении структур в hex-редакторе, позволяет восстановить данные там, где программные решения оказываются бессильны.

Несмотря на переход Apple на файловую систему APFS, HFS+ остаётся актуальной благодаря широкому распространению носителей, использующих данную структуру. Разработанный алгоритм представляет интерес для специалистов в области компьютерной криминалистики и восстановления данных, поскольку помогает глубже понять принципы организации HFS+ и может послужить основой для разработки автоматизированных инструментов анализа. В дальнейшем планируется расширить методику для работы со сложными случаями восстановления, включая обработку фрагментированных файлов и нестандартных структур каталогов.

Библиографический список

1. HFS+: Structures and Features, exFAT vs. HFS+ vs. NTFS [Электронный ресурс] // iBoysoft. – 2025. – URL: https://iboysoft.com/wiki/hfs-plus.html (дата обращения: 05.11.2025).

Рубрика 1. Информатика и информационные процессы

2. Apple File System Reference [Электронный ресурс] // Apple Developer Documentation. – Apple Inc., 2017. – URL: https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf (дата обращения: 15.10.2025).
3. Apple Inc. Technical Note TN1150: HFS Plus Volume Format [Электронный ресурс] // Apple Developer Documentation. – 2004. – URL: https://developer.apple.com/library/archive/technotes/tn/tn1150.html (дата обращения: 15.10.2025).
4. Порядок восстановления данных в файловой системе HFS+ [Электронный ресурс] // Хабр. – 2021. – URL: https://habr.com/ru/companies/hetmansoftware/articles/547716/ (дата обращения: 06.11.2025).
5. Carrier, B. File System Forensic Analysis / B. Carrier. – Boston : Addison-Wesley Professional, 2005. – 640 p.
6. The Sleuth Kit [Электронный ресурс] // The Sleuth Kit. – URL: https://www.sleuthkit.org/sleuthkit/ (дата обращения: 06.11.2025).
7. Grenier, C. TestDisk Documentation [Электронный ресурс] / C. Grenier // CGSecurity. – URL: https://www.cgsecurity.org/wiki/TestDisk (дата обращения: 06.11.2025).

References

1. iBoysoft. (2025). HFS+: Structures and features, exFAT vs. HFS+ vs. NTFS. iBoysoft. https://iboysoft.com/wiki/hfs-plus.html
2. Apple Inc. (2017). Apple File System Reference. Apple Developer Documentation. https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf
3. Apple Inc. (2004). Technical Note TN1150: HFS Plus Volume Format. Apple Developer Documentation. https://developer.apple.com/library/archive/technotes/tn/tn1150.html
4. Hetman Software. (2021). Procedure of data recovery in HFS+ file system. Habr. https://habr.com/ru/companies/hetmansoftware/articles/547716/
5. Carrier, B. (2005). File system forensic analysis. Addison-Wesley Professional.
6. The Sleuth Kit. (n.d.). The Sleuth Kit. https://www.sleuthkit.org/sleuthkit/
7. Grenier, C. (n.d.). TestDisk documentation. CGSecurity. https://www.cgsecurity.org/wiki/TestDisk

Информация об авторах

Чурсина Серафима Сергеевна – студент кафедры «Математических методов обеспечения безопасности систем», ФГАОУ ВО «РГУ нефти и газа (НИУ) имени И.М. Губкина», г. Москва, e-mail: chursinass@mail.ru.

Новикова Дарья Дмитриевна – студент кафедры «Математических методов обеспечения безопасности систем», ФГАОУ ВО «РГУ нефти и газа (НИУ) имени И.М. Губкина», г. Москва, e-mail: novikova.d@mail.ru.

ANALYSIS OF HFS+ STRUCTURES AND DATA RECOVERY TECHNIQUE

D. D. Novikova1, S. S. Chursina1
1National University of Oil and Gas «Gubkin University»

Abstract. This article presents a structural analysis of the HFS+ file system and describes a manual data recovery technique based on direct examination of service areas of a volume. The relevance of the study is determined by the fact that HFS+ is still encountered on external drives, backup media and legacy Apple devices, while automated recovery tools do not always correctly interpret damaged metadata. The paper examines the role of the Volume Header, Catalog File parameters, the organization of catalog B-trees, leaf node layout, record offset arrays, HFSPlusCatalogKey fields and the main structures of file and folder records. A practical example demonstrates how to determine the allocation block size, calculate the physical offset of the catalog file, identify root directory records, establish object identifiers and extract file contents using extent descriptors. The proposed method is intended for work in a hex editor and does not require expensive specialized software. The results show that manual low-level analysis can be used to recover individual critical files, verify automated tool output and study the internal principles of file system organization. The technique is useful for data recovery specialists, digital forensics practitioners and system administrators working with Apple storage media.

Keywords: HFS+, data recovery, structural analysis, B-tree, catalog file, extents, Apple file system.

Information about the authors

Chursina Serafima Sergeevna – student of the Department of Mathematical Methods of System Safety Assurance, Gubkin Russian State University of Oil and Gas (National University), Moscow, e-mail: chursinass@mail.ru.

Novikova Daria Dmitrievna – student of the Department of Mathematical Methods of System Safety Assurance, Gubkin Russian State University of Oil and Gas (National University), Moscow, e-mail: novikova.d@mail.ru.