РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 718 233**(13) **C1**

(51) МПК
G06F 16/10 (2019.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК
G06F 16/10 (2020.02); G06F 12/00 (2020.02)

(21)(22) Заявка: **2019130830, 01.10.2019**

(24) Дата начала отсчета срока действия патента:
01.10.2019

Дата регистрации:
31.03.2020

Приоритет(ы):
(22) Дата подачи заявки: **01.10.2019**

(45) Опубликовано: **31.03.2020** Бюл. № **10**

Адрес для переписки:
121205, г. Москва, ИЦ "Сколково", ул. Нобеля, д. 5, оф. 125, Котлов Дмитрий Владимирович

(72) Автор(ы):
Башев Владимир Николаевич (RU),
Ильин Николай Олегович (RU)

(73) Патентообладатель(и):
Общество с ограниченной ответственностью «ПИРФ» (ООО «ПИРФ») (RU)

(56) Список документов, цитированных в отчете о поиске: **RU 2574824 C2, 10.02.2016. KR 100754306 B1, 03.09.2007. EP 629960 B1, 24.05.2000. RU 2408070 C2, 27.12.2010.**

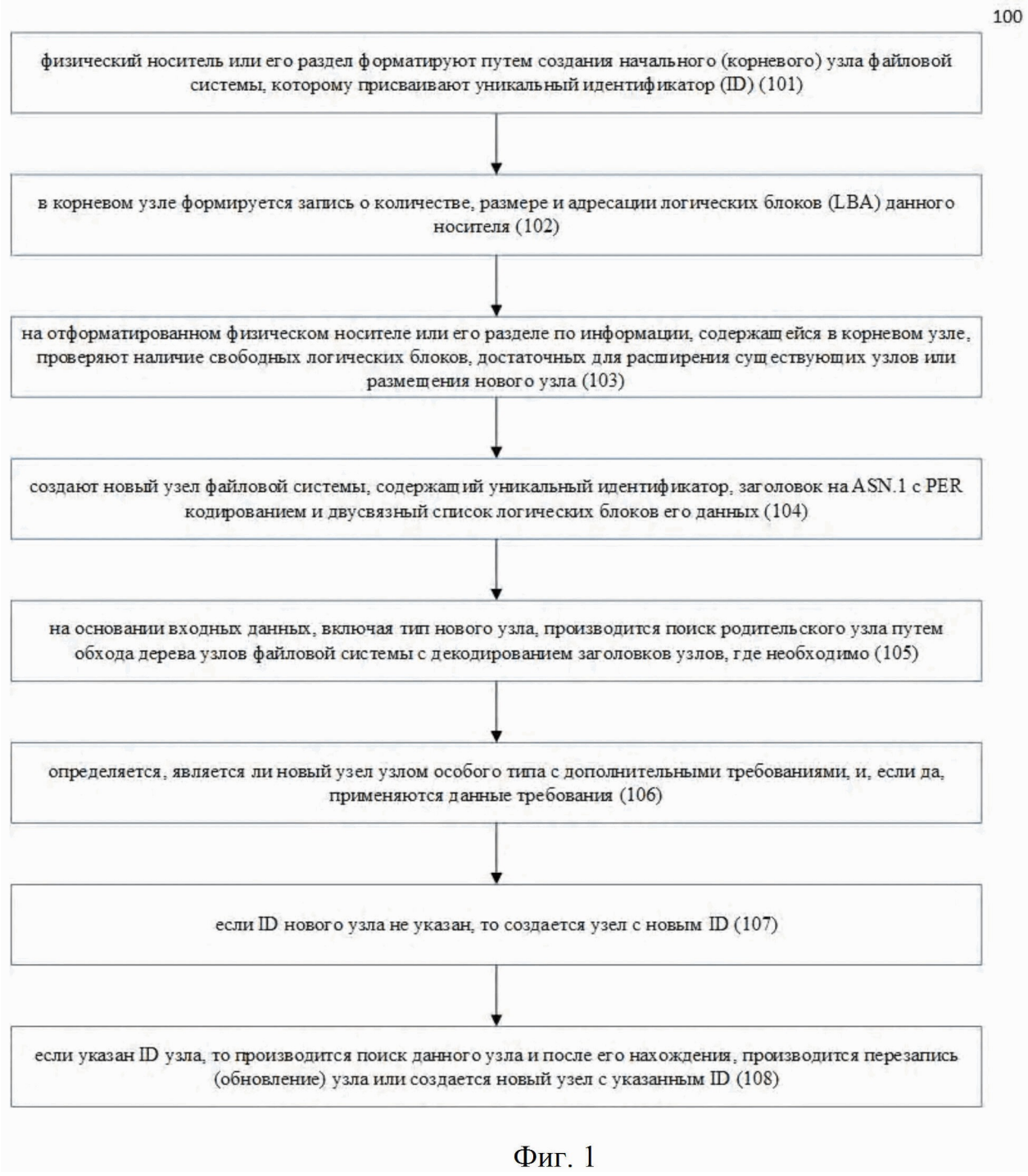
(54) СПОСОБ ПОСТРОЕНИЯ ФАЙЛОВОЙ СИСТЕМЫ НА БАЗЕ ИЕРАРХИИ УЗЛОВ

(57) Реферат:

Изобретение относится к области вычислительной техники. Технический результат заключается в расширении арсенала средств. Способ построения файловой системы на базе иерархии узлов выполняется с помощью вычислительного устройства и содержит этапы: физическому носителю присваивают уникальный идентификатор (ID), на отформатированном носителе проверяют наличие свободных логических блоков, достаточных для расширения существующих узлов или размещения нового узла, создают новый узел файловой системы, содержащий уникальный идентификатор. 9 з.п. ф-лы, 8 ил.

RU 2718233 C1

RU 2718233 C1



Фиг. 1

RU 2718233 C1

RU 2718233 C1

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 718 233**(13) **C1**

(51) Int. Cl.
G06F 16/10 (2019.01)

(12) ABSTRACT OF INVENTION

(52) CPC
G06F 16/10 (2020.02); G06F 12/00 (2020.02)

(21)(22) Application: **2019130830, 01.10.2019**

(24) Effective date for property rights:
01.10.2019

Registration date:
31.03.2020

Priority:
(22) Date of filing: **01.10.2019**

(45) Date of publication: **31.03.2020** Bull. № **10**

Mail address:
121205, g. Moskva, ITS "Skolkovo", ul. Nobelya, d. 5, of. 125, Kotlov Dmitrij Vladimirovich

(72) Inventor(s):
Bashev Vladimir Nikolaevich (RU),
Ilin Nikolai Olegovich (RU)

(73) Proprietor(s):
Obshchestvo s ogranichennoi otvetstvennostiu «PIRF» (OOO «PIRF») (RU)

(54) METHOD OF CONSTRUCTING A FILE SYSTEM BASED ON A HIERARCHY OF NODES

(57) Abstract:

FIELD: computer equipment.

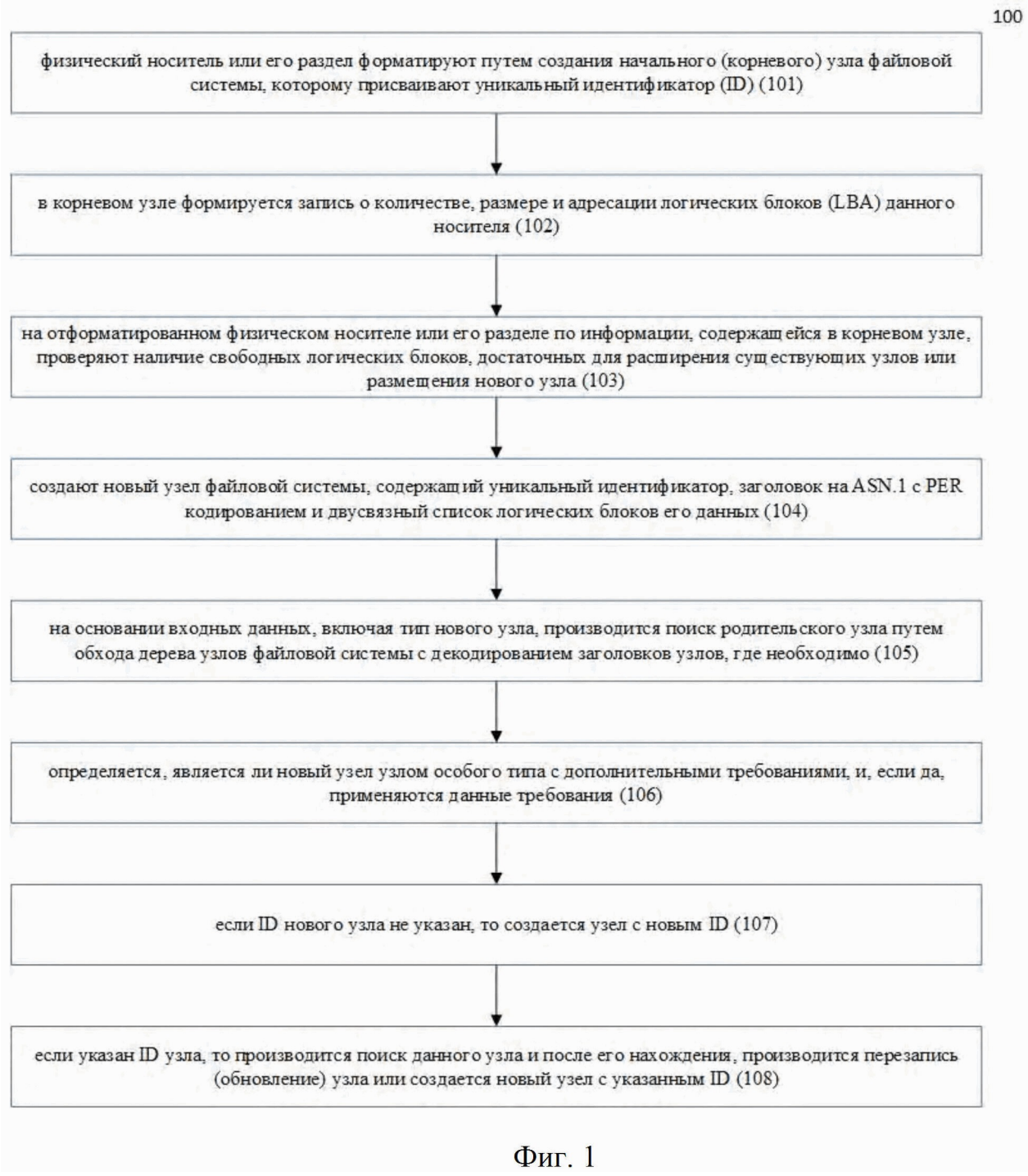
SUBSTANCE: method for constructing a file system based on a node hierarchy is carried out using a computing device and comprises steps of: assigning a unique identifier (ID) to the physical medium, on the formatted medium, presence of free logical blocks, which are sufficient for expansion of existing units or placement of a new node, is checked, a new file system node containing a unique identifier is created.

EFFECT: technical result is wider range of means.

10 cl, 8 dwg

RU 2718233 C1

RU 2718233 C1



Фиг. 1

RU 2718233 C1

ОБЛАСТЬ ТЕХНИКИ

Настоящее техническое решение относится к области вычислительной техники, в частности, к способу построения файловой системы на базе иерархии узлов.

УРОВЕНЬ ТЕХНИКИ

В настоящее время существует большое количество решений, описывающих файловые системы.

Файловые системы определяют способ хранения данных. От них зависит, с какими ограничениями столкнется пользователь, насколько быстрыми будут операции чтения и записи и как долго накопитель проработает без сбоев.

Из уровня техники известна файловая система NTFS (New Technology File System - «файловая система новой технологии») - стандартная файловая система для семейства операционных систем Microsoft Windows NT. NTFS поддерживает систему метаданных и использует специализированные структуры данных для хранения информации о файлах для улучшения производительности, надёжности и эффективности использования дискового пространства. NTFS хранит информацию о файлах в главной файловой таблице - Master File Table (MFT). NTFS имеет встроенные возможности разграничивать доступ к данным для различных пользователей и групп пользователей (списки контроля доступа - Access Control Lists (ACL)), а также назначать квоты (ограничения на максимальный объём дискового пространства, занимаемый теми или иными пользователями). NTFS использует систему журналирования USN для повышения надёжности файловой системы.

Данная файловая система имеет ограниченную межплатформенную совместимость.

Из уровня техники известна файловая система - FAT32. Однако в данной файловой системе имеются ограничения в размерах файлов и размерах разделов. FAT32 не предназначена для хранения больших объемов данных и установки тяжелых приложений.

Известно также решение, которое описывает способ реализации формата расширяемой файловой системы. Такое решение раскрыто в патенте RU2574824C2 (МАЙКРОСОФТ ТЕКНОЛОДЖИ ЛАЙСЕНСИНГ, ЭлЭлСи, опубл. 10.02.2016).

Также известна система распределенной файловой системы обслуживания, которая обеспечивает распределенную архитектуру обслуживания файлов с виртуализацией хранения метаданных (KR100754306B1 (INTERNATIONAL BUSINESS MACHINES CORPORATION, опубл. 03.09.2007).

Из патента EP0629960B1 (SUN MICROSYSTEMS INC, опубл. 24.05.2000) известна архитектура расширяемых файловых систем. Известная архитектура позволяет расширять функциональность файловой системы, устанавливая (или создавая) новые файловые системы (уровни) поверх существующих файловых систем.

Однако, известные из уровня техники решения, описывающие файловые системы, имеют ограниченную функциональность, в частности:

- Ограничения по максимальному размеру файловой системы, числу элементов;
- Ограничения по максимальному размеру файла;
- Ограничения по типу данных, которые могут быть представлены в виде файла, включая использование файловой системы как реестра системных компонентов ОС.

СУЩНОСТЬ ИЗОБРЕТЕНИЯ

Данное техническое решение направлено на устранение недостатков, присущих существующим решениям из известного уровня техники.

Технической проблемой, на решение которой направлено заявленное техническое решение, является создание нового компьютерно-реализуемого способа построения

файловой системы на базе иерархии узлов, который охарактеризован в независимом пункте формулы. Дополнительные варианты реализации настоящего изобретения представлены в зависимых пунктах изобретения.

Техническим результатом, на решение которого направлено заявленное техническое решение, является использование уникального идентификатора узла файловой системы, который в свою очередь для компонентов операционной системы совпадает с уникальным идентификатором компонента. Таким образом, файловая система одновременно представляет своего рода реестр для компонентов, в котором поиск и доступ к компонентам осуществляется напрямую без дополнительного уровня представления. Данный подход особенно эффективен для устройств с малым объемом оперативной памяти, для которых критично использовать небольшие объемы реестров, загружаемых в память.

В заявленном решении поиск компонент происходит по UGUID в самой файловой системе, так как UGUID совпадает с ID узла системного типа.

Дополнительной гибкостью заявленного решения является, то что в корне или в каталоге файловой системы может быть несколько файлов или каталогов с одинаковыми именами, а также файлы, не содержащие имен и это не будет конфликтом на уровне файловой системы.

Еще одним положительным эффектом данного решения является сниженный размер узла файловой системы, в котором заголовок, описывающий тип элемента файловой системы может быть пустым, и такой узел состоит только из уникального идентификатора узла, пустого заголовка и данных, что также актуально на устройствах с ограниченными ресурсами. Таким образом, допустима наиболее минимальная иерархия файловой системы, состоящей только из корневого узла и нескольких узлов системных компонентов, у которых заголовок пустой, т.е. не требующий кодирования и декодирования, а доступ к компонентам осуществляется по их системному идентификатору.

В предпочтительном варианте реализации заявлен компьютерно-реализуемый способ построения файловой системы на базе иерархии узлов, выполняемый с помощью вычислительного устройства и содержащий этапы, на которых:

физический носитель или его раздел форматируют путем создания начального (корневого) узла файловой системы, которому присваивают уникальный идентификатор (ID), при этом в корневом узле формируется запись о количестве, размере и адресации логических блоков (LBA) данного носителя;

на отформатированном физическом носителе или его разделе по информации, содержащейся в корневом узле, проверяют наличие свободных логических блоков, достаточных для расширения существующих узлов или размещения нового узла;

создают новый узел файловой системы, содержащий уникальный идентификатор, заголовок на ASN.1 с PER кодированием и двусвязный список логических блоков его данных, с использованием следующих действий:

- на основании входных данных, включая тип нового узла, производится поиск родительского узла путем обхода дерева узлов файловой системы с декодированием заголовков узлов, где необходимо;

- определяется, является ли новый узел узлом особого типа с дополнительными требованиями (например, ID узла системного типа является системным идентификатором UGUID), и если да, применяются данные требования;

- если ID нового узла не указан, то создается узел с новым ID, если указан ID узла, то производится поиск данного узла и после его нахождения, производится перезапись

(обновление) узла или создается новый узел с указанным ID.

В частном варианте формат уникального идентификатора узла файловой системы (ID) состоит из двух частей: преамбулы и целого числа и является неограниченным целым.

В другом частном варианте файловая система является иерархической и имеет древовидную структуру.

В другом частном варианте особым типом узла являются системные узлы, у которых уникальный идентификатор ID соответствует уникальному идентификатору UGUID системного компонента, и которые формируют связный список на файловой системе.

В другом частном варианте заголовок узла не имеет фиксированного размера и может расширяться.

В другом частном варианте заголовок может включать новый, определенный пользователем тип узла, а узел содержать данные соответствующего типа.

В другом частном варианте узел с типом каталог, может содержать другие узлы типа каталог и узлы типа файл с одинаковыми именами.

В другом частном варианте каждый используемый узлом логический блок имеет запись номера предыдущего логического блока и номера следующего логического блока для организации двусвязного списка, что в свою очередь позволят делать операции создания/вставки/перемещения/удаления логических блоков в двусвязном списке.

В другом частном варианте логический блок, у которого запись номера предыдущего логического блока равна номеру самого логического блока, т.е. указывает сам на себя, является начальным логическим блоком узла.

В другом частном варианте логический блок, у которого запись номера следующего логического блока равна номеру самого логического блока, т.е. указывает сам на себя, является завершающим логическим блоком узла.

ОПИСАНИЕ ЧЕРТЕЖЕЙ

Реализация изобретения будет описана в дальнейшем в соответствии с прилагаемыми чертежами, которые представлены для пояснения сути изобретения и никоим образом не ограничивают область изобретения. К заявке прилагаются следующие чертежи:

Фиг. 1 иллюстрирует компьютерно-реализуемый способ;

Фиг. 2 иллюстрирует тип данных Неограниченное Целое Число (Unlimited Integer);

Фиг. 3 иллюстрирует пример расширения Неограниченного Целого Числа, на примерах представлено число имеющее одинаковое значение равное единице;

Фиг. 4 иллюстрирует дерево файловой системы;

Фиг. 5 иллюстрирует элемент файловой системы;

Фиг. 6 иллюстрирует развернутое представление элемента файловой системы;

Фиг. 7 иллюстрирует связный список родственных узлов (системных компонентов);

Фиг. 8 иллюстрирует блок схема создания нового узла.

ДЕТАЛЬНОЕ ОПИСАНИЕ ИЗОБРЕТЕНИЯ

В приведенном ниже подробном описании реализации изобретения приведены многочисленные детали реализации, призванные обеспечить отчетливое понимание настоящего изобретения. В виду того, что специалистам в области проектирования операционных и компьютерных систем знакомы общеупотребительные термины, компоненты и процедуры, то данные методы, названия и компоненты не были описаны подробно, чтобы не затруднять понимание особенностей настоящего изобретения и отличительные возможности его использования.

Кроме того, из приведенного изложения будет ясно, что изобретение не ограничивается приведенной реализацией. Многочисленные возможные модификации,

изменения, вариации и замены, сохраняющие суть и форму настоящего изобретения, будут очевидными для квалифицированных в предметной области специалистов.

Настоящее изобретение направлено на создание компьютерно-реализуемого способа построения файловой системы на базе иерархии узлов.

В заявленном решении архитектура файловой системы построена на использовании уникального идентификатора узла файловой системы.

Формат уникального идентификатора узла файловой системы, состоит из двух частей: преамбулы и непосредственно самого целого числа.

Минимальный размер формата равен 2-м байтам информации (байт - единица измерения согласно ГОСТ 8.417-2002). Максимальный размер формата стремиться к бесконечности, ограничение составляет аппаратные свойства вычислительной техники.

Минимальный размер преамбулы один байт. Преамбула не может принимать нулевое значение. Преамбула состоит из двух частей: бита расширения и длины целого числа в байтах. При переполнении битов преамбулы - длины целого числа, бит расширения сдвигается вправо, увеличивая размер преамбулы на один байт. Таким образом, размер преамбулы стремиться к бесконечности, тем самым увеличивая битовые поля для задания длины целого числа в байтах. В следствии чего за счет увеличения задания длины размера целого числа, целое число также стремиться к бесконечности.

На Фиг. 1 представлен компьютерно-реализуемый способ (100) построения файловой системы на базе иерархии узлов, выполняемый с помощью вычислительного устройства.

На этапе (101) физический носитель или его раздел форматируют путем создания начального (корневого) узла файловой системы, которому присваивают уникальный идентификатор (ID). При этом на этапе (102) в корневом узле формируется запись о количестве, размере и адресации логических блоков (LBA) данного носителя.

Далее на этапе (103) на отформатированном физическом носителе или его разделе по информации, содержащейся в корневом узле, проверяют наличие свободных логических блоков, достаточных для расширения существующих узлов или размещения нового узла.

На этапе (104) создают новый узел файловой системы, содержащий уникальный идентификатор, заголовок на ASN.1 с PER кодированием и двусвязный список логических блоков его данных, с использованием следующих действий:

На этапе (105) на основании входных данных, включая тип нового узла, производится поиск родительского узла путем обхода дерева узлов файловой системы с декодированием заголовков узлов, где необходимо. Например, для системных узлов не требуется декодирования, и они определяются по своим уникальным UGUID).

На этапе 106 определяется, является ли новый узел узлом особого типа с дополнительными требованиями, и, если да, применяются данные требования.

Узлами особого типа являются, например, системные узлы или узлы с новыми типами, которые определил разработчик операционной системы или компонента менеджера файловой системы, с которыми используется данная файловая система. Компонентная архитектура позволяет делегировать обработку. К примеру, при чтении и декодировании заголовка узла нестандартного типа (т.е. типа, определенного разработчиком), файловый менеджер или утилиты, работающие с файловой системой, делегируют обработку узла пользовательскому компоненту, который знает, что это за тип узла и как его обработать. В случае, если в системе для файлового менеджера не установлен соответствующий компонент, способный обработать данный узел, система может отобразить его как неизвестный или запросить установку необходимого компонента для обработки неизвестных типов узлов.

На этапе 107, если ID нового узла не указан, создается узел с новым ID. На этапе 108, если указан ID узла, то производится поиск данного узла и, после его нахождения, производится перезапись (обновление) узла или создается новый узел с указанным ID.

Архитектура файловой системы, построена на использовании уникального идентификатора узла файловой системы, имеющий тип данных Неограниченное Целое Число (Unlimited Integer), формат которого представлен на фиг.2.

Формат уникального идентификатора узла файловой системы, состоит из двух частей: преамбулы и непосредственно самого целого числа. Минимальный размер формата равен двум Байтам. Максимальный размер формата стремиться к бесконечности, ограничение составляет аппаратные свойства вычислительной техники. Минимальный размер преамбулы один Байт.

Преамбула не может принимать нулевое значение. Преамбула состоит из двух частей бита расширения и длины целого числа в Байтах. При переполнении битов преамбулы - длины целого числа, бит расширения сдвигается вправо, увеличивая размер преамбулы на один Байт.

Таким образом, размер преамбулы стремиться к бесконечности, тем самым увеличивая битовые поля для задания длины целого числа в Байтах. В следствии чего за счет увеличения задания длины размера целого числа, целое число также стремиться к бесконечности. Примеры расширения формата приведены на фиг.3.

Иерархия файловой системы представлена на фиг.4, она состоит из элементов файловой системы. Каждый элемент файловой системы, представляет собой узел дерева иерархической файловой системы и состоит из трех частей фиг.5:

ID - Уникальный идентификатор узла файловой системы используется в формате Unlimited Integer.

Header (заголовок) - описывает тип узла файловой системы на ASN.1 с использованием PER кодирования. Полное описание заголовка описывается спецификацией, при этом заголовок может быть пустым.

Data - представляет собой двусвязный список блоков данных данного узла.

На фиг.6 изображено развернутое представление элемента файловой системы, ниже приведены пояснения к данной фигуре:

Архитектура файловой системы использует современную адресацию логических блоков (LBA), для старой системы адресации (CHS), необходимо воспользоваться правилами преобразования для получения адреса блока в режиме LBA установленными техническим комитетом X3T10. В связи с тем, что комитет стандартизации постоянно увеличивают разрядность LBA, что в свою очередь обусловлено ростом размеров диска, то номер блока LBA задается с помощью типа данных.

Благодаря использованию Неограниченное Целое Число (Unlimited Integer) фиг.2, в качестве адресации LBA, файловая система не ограничена по размеру (числу логических блоков). Для первого блока LBA=0, в формате Неограниченного Целого Числа, первый блок будет записан как 0x010100h.

Данные узла на физическом носителе, представляют из себя двусвязный список логических блоков LBA. При этом сам узел состоит из идентификатора, заголовка и данных, которые размещаются в данном двухсвязном списке из логических блоков (сами логические блоки могут иметь разный размер 512 байт, 1024 байт …).

Каждый, используемый узлом, логический блок LBA имеет начальную запись номера предыдущего блока и номера следующего блока для организации двусвязного списка, что в свою очередь позволят делать операции создания/вставки/перемещения/удаления логических блоков в двусвязном списке.

Логический блок, у которого запись номера предыдущего блока (находится в начале блока) равна номеру первого логического блока, т.е. указывает на загрузочный сектор, является корневым блоком.

На физическом носителе может находится несколько таких блоков, что указывает на резервные копии корневых блоков, для повышения надежности и возможности восстановления целостности файловой системы при сбое.

Логический блок, у которого запись номера предыдущего блока равна номеру самого логического блока, т.е. указывает сам на себя, является начальным логическим блоком самого узла.

Логический блок, у которого запись номера следующего блока равна номеру самого логического блока, т.е. указывает сам на себя, является завершающим блоком узла.

Логический блок, у которого отсутствует запись на предыдущий блок, считается свободным (не занятым).

Для логических блоков, которые были повреждены, в заголовке корневого узла (не путать с корневым логическим блоком), формируется запись согласно спецификации списка поврежденных блоков, как одиночных, так и диапазона логических блоков.

Для логических блоков, которые были освобождены в результате перемещения/удаления логических блоков в двусвязном списке, в заголовке корневого узла (не путать с корневым логическим блоком), формируется запись согласно спецификации списка освобожденных блоков, как одиночных, так и диапазона логических блоков.

В связи с тем, что корневой узел - элемент файловой системы имеет заголовок описывающий тип элемента файловой системы на ASN.1 с PER кодированием, что в сою очередь делает файловую систему гибкой и функциональной, а с другой стороны говорит о недостатке - сложном процессе кодирования и декодирования самого заголовка, для быстрого поиска системных узлов используется механизм использования системного UGUID для поиска системных компонентов (системных узлов). Для этого в завершающем логическом блоке корневого узла отводится место под адрес логического блока первого системного компонента данного носителя информации. А именно, после начальной записи (номера предыдущего блока и номера следующего блока), следует адрес - номер логического блока первого системного компонента, размещенного на физическом носителе.

В завершающем логическом блоке первого системного компонента фиг.7, следует адрес- номер логического блока второго системного компонента и т.д. При этом ID узла соответствует UGUID системного компонента. Заголовок данного узла может быть пустым или закодирован согласно спецификации и в то же время может игнорироваться, т.е. не декодироваться, так как двусвязный список логических блоков представляет из себя системный компонент, содержащий в завершающем логическом блоке узла адрес- номер логического блока следующего системного компонента (системного узла), что в свою очередь облегчает реализацию начального загрузчика операционной системы для поиска и загрузки системных компонентов.

Вся информация о типе узла, его свойствах кодируется в заголовке. Заголовок представляет собой бинарные данные, закодированные с помощью PER согласно спецификации, описанной на ASN.1.

В результате использования ASN.1 и PER кодирования - заголовок становиться универсальным инструментом, может включать в себя десятки - сотни страниц описания заголовка (спецификации), постоянно расширяться, кастомизироваться, сохраняя обратную совместимость и при этом кодироваться только в тот размер данных, на которые есть информация, что выгодно отличает заявленное решение от других

известных из уровня техники форматов файловых систем, где поля данных заголовка представляют собой жесткие структуры и обычно еще включают в себя зарезервированные (Reserved) поля.

Заголовок определяет тип узла, который описан в спецификации - это может быть директория, файл, ссылка, устройство, база данных, пользовательский узел, что в свою очередь, дает гибкие возможности по использованию файловой системы и ее развитию. Помимо этого, в зависимости от типа узла, заголовок может включать в себя другие спецификации, к примеру стандарт ITU-T X.509.

Заголовок не имеет фиксированного размера и может расширятся за счет вставки дополнительных логических блоков в двусвязный список логических блоков узла.

В результате работы с файловой системой - удалением, перемещением, вставкой данных на физическом носителе, образуются пропущенные, или поврежденные логические блоки (сектора), данные об этих логических блоках, как единичных, так и диапазоне, хранятся в заголовке корневого узла.

Информация о неполных логических блоках, т.е. логических блоках, которые не полностью заполнены данными, которые могут образоваться в результате операции вставки, также хранится в заголовке узла, к которому данные логические блоки относиться.

На файловой системе может находится несколько корневых узлов, являющимися резервными копиями основного корневого узла.

Информация об основном корневом узле, его начальном логическом блоке преобразованным в формат CHS, может быть записана в таблицу разделов главной загрузочной записи MBR, код типа раздела должен соответствовать 0xE0h (ОС архитектуры АОМК), в остальные свободные разделы может быть записана информация о резервных копиях корневого узла.

В случае повреждения корневого узла или отдельных узлов файловой системы и отсутствии резервных копий корневого узла, простое последовательное сканирование логических блоков на основании начальной записи в логическом блоке, позволяет найти узлы - цепочки двухсвязных списков, а при декодировании заголовка узла определить тип узла и его взаимосвязи: родительские, дочерние узлы, тем самым восстановить иерархию файловой системы.

В случае, если заголовок узла невозможно декодировать, то узел считается поврежденным.

В случае, если при восстановлении были найдены узлы, у которых родительский узел был поврежден и целостность иерархии была нарушена, то решения по присоединению к корневому узлу или созданию нового родительского узла для найденных узлов возлагается на пользователя, через интерфейс утилиты по восстановлению файловой системы.

На фиг.8 представлена блок-схема создания узла файловой системы. Данный алгоритм может использоваться файловым менеджером или утилитами для работы с представленной файловой системой.

При создании узла файловой системы, входными данными, являются тип узла, путь к новому узлу, который представляет собой массив из ID узлов или имена каталогов для классического представления, и вспомогательные параметры в зависимости от типа создаваемого узла. Для примера, если типом узла является директория, то вспомогательными (не обязательными) параметрами являются имя директории, атрибуты политики, возможно пароль, если предполагается доступ к каталогу по паролю, используемый алгоритм шифрования и т.п.

Если предполагается создания узла под системный компонент, обязательным входным параметром, является идентификатор компонента UGUID, для проверки уникальности идентификатора создаваемого узла.

Так как идентификатор узла в данном случае будет совпадать с идентификатором компонента, то не допускается в рамках одного дерева иерархии файловой системы наличия одинаковых идентификаторов узлов, все идентификаторы узлов должны быть уникальными.

После определения входных данных производится чтение корневого узла с последующим декодированием его заголовка.

Из заголовка узла, определяется информация о родственных и дочерних узлах, на основании информации по узлам, производится поиск родительского узла для нового узла. Поиск родительского узла сопровождается чтением и декодирование узлов согласно пути к созданию нового узла.

Поиск может производиться как по ID узла, так и по классическому символьному пути, при этом стоит учитывать, что родительские узлы могут иметь дочерние узлы с одинаковыми именами, путь к таким узлам должен включать дополнительное представление, как вариант индекс (пример: ..\FooName[1] , ..\ FooName[2] , ..\ FooName [3]).

В случае отсутствия имён представление будет иметь вид, например: ..\NoName[1] , ..\ NoName[2] , ..\NoName[3].

После успешного поиска родительского узла, определяется требуется ли создание системного узла. Если ID узла не указано, то создается новый узел с новым ID.

ID узла формируется простым увеличением индекса из диапазона допустимых значений или берется из кэша свободных ID узлов, образовавшихся в результате удаления.

Если входными данными является создание системного узла, указано ID узла и адрес последнего логического блока связного списка системных компонентов, то производится поиск существующего ID узла. В случае нахождения узла, производится перезапись (обновление) системного компонента, в противном случае создается новый системный узел с указанным ID, в данном случае ID узла равен UGUID компонента.

Процесс создания нового узла сопровождается проверкой наличия свободных логических блоков (размера свободного пространства на носителе информации) достаточных для размещения узла.

В настоящих материалах заявки было представлено предпочтительное раскрытие осуществление заявленного технического решения, которое не должно использоваться как ограничивающее иные, частные воплощения его реализации, которые не выходят за рамки испрашиваемого объема правовой охраны и являются очевидными для специалистов в соответствующей области техники.

(57) Формула изобретения

1. Компьютерно-реализуемый способ построения файловой системы на базе иерархии узлов, выполняемый с помощью вычислительного устройства и содержащий этапы, на которых:

физический носитель или его раздел форматируют путем создания начального корневого узла файловой системы, которому присваивают уникальный идентификатор (ID), при этом в корневом узле формируется запись о количестве, размере и адресации логических блоков (LBA) данного носителя;

на отформатированном физическом носителе или его разделе по информации,

содержащейся в корневом узле, проверяют наличие свободных логических блоков, достаточных для расширения существующих узлов или размещения нового узла;

создают новый узел файловой системы, содержащий уникальный идентификатор, заголовок на ASN.1 с PER кодированием и двусвязный список логических блоков его данных, с использованием следующих действий:

- на основании входных данных, включая тип нового узла, производится поиск родительского узла путем обхода дерева узлов файловой системы с декодированием заголовков узлов, где необходимо;

- определяется, является ли новый узел узлом особого типа с дополнительными требованиями, и если да, применяются данные требования;

- если ID нового узла не указан, то создается узел с новым ID, если указан ID узла, то производится поиск данного узла и после его нахождения, производится перезапись, обновление узла или создается новый узел с указанным ID.

2. Способ по п.1, характеризующийся тем, что формат уникального идентификатора узла файловой системы состоит из двух частей: преамбулы и целого числа и является неограниченным целым.

3. Способ по п.1, характеризующийся тем, что файловая система является иерархической и имеет древовидную структуру.

4. Способ по п.1, характеризующийся тем, что особым типом узла являются системные узлы, у которых уникальный идентификатор ID соответствует уникальному идентификатору UGUID системного компонента и которые формируют связный список на файловой системе.

5. Способ по п.1, характеризующийся тем, что заголовок узла не имеет фиксированного размера и может расширяться.

6. Способ по п.1, характеризующийся тем, что заголовок может включать новый, определенный пользователем тип узла, а узел содержать данные соответствующего типа.

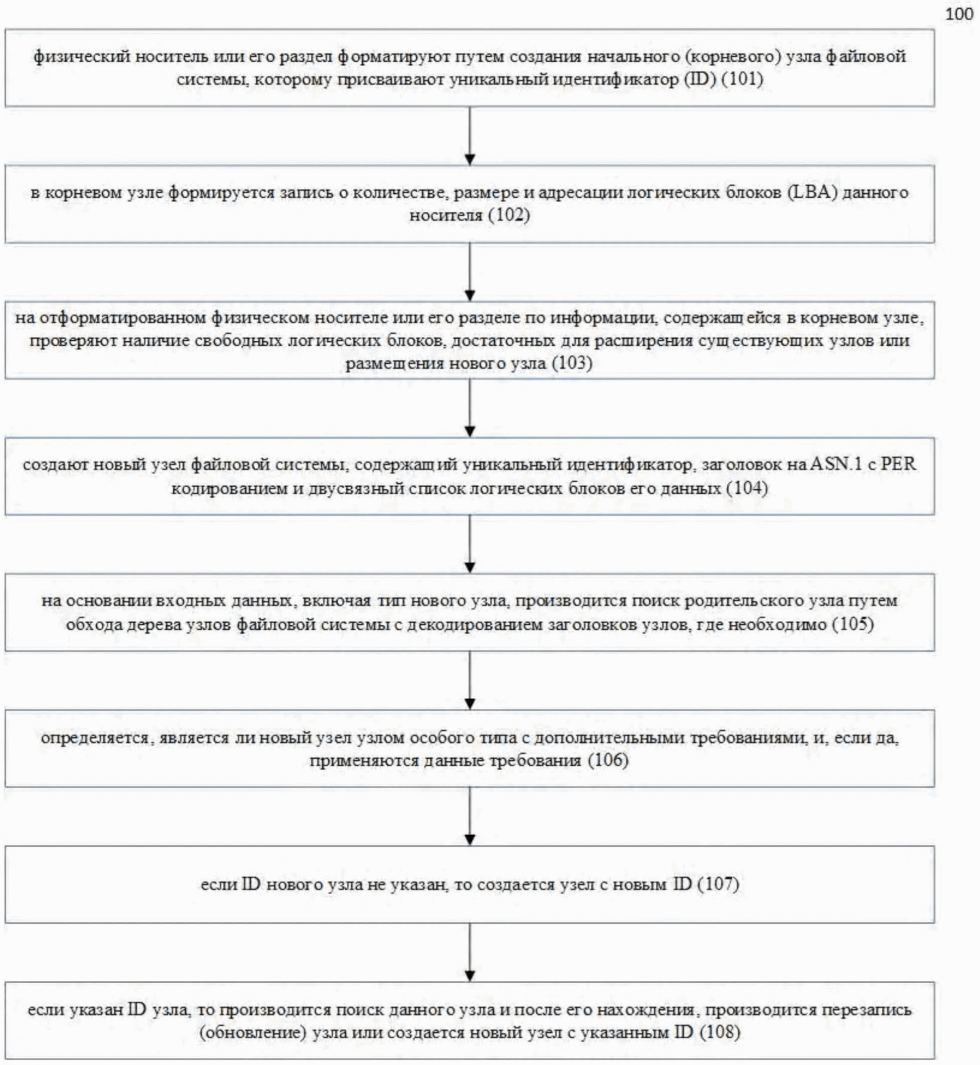
7. Способ по п.1, характеризующийся тем, что узел с типом каталог может содержать другие узлы типа каталог и узлы типа файл с одинаковыми именами.

8. Способ по п.1, характеризующийся тем, что каждый используемый узлом логический блок имеет запись номера предыдущего логического блока и номера следующего логического блока для организации двусвязного списка, что в свою очередь позволят делать операции создания/вставки/перемещения/удаления логических блоков в двусвязном списке.

9. Способ по п.1, характеризующийся тем, что логический блок, у которого запись номера предыдущего логического блока равна номеру самого логического блока, т.е. указывает сам на себя, является начальным логическим блоком узла.

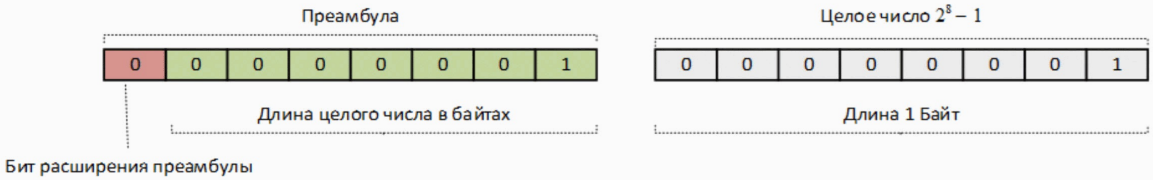
10. Способ по п.1, характеризующийся тем, что логический блок, у которого запись номера следующего логического блока равна номеру самого логического блока, указывает сам на себя, является завершающим логическим блоком узла.

1

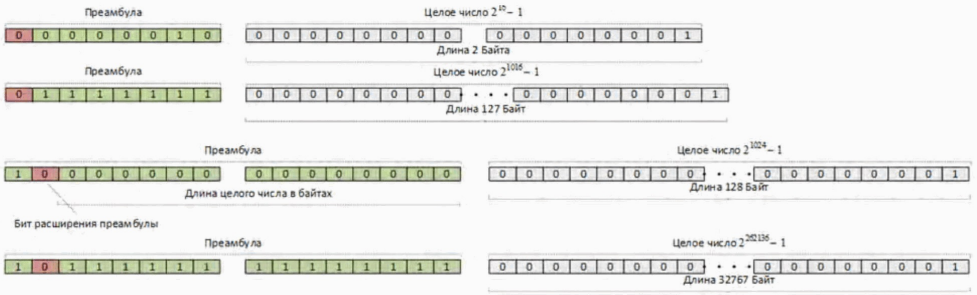

100
физический носитель или его раздел форматируют путем создания начального (корневого) узла файловой системы, которому присваивают уникальный идентификатор (ID) (101)
в корневом узле формируется запись о количестве, размере и адресации логических блоков (LBA) данного носителя (102)
на отформатированном физическом носителе или его разделе по информации, содержащейся в корневом узле, проверяют наличие свободных логических блоков, достаточных для расширения существующих узлов или размещения нового узла (103)
создают новый узел файловой системы, содержащий уникальный идентификатор, заголовок на ASN.1 с PER кодированием и двусвязный список логических блоков его данных (104)
на основании входных данных, включая тип нового узла, производится поиск родительского узла путем обхода дерева узлов файловой системы с декодированием заголовков узлов, где необходимо (105)
определяется, является ли новый узел узлом особого типа с дополнительными требованиями, и, если да, применяются данные требования (106)
если ID нового узла не указан, то создается узел с новым ID (107)
если указан ID узла, то производится поиск данного узла и после его нахождения, производится перезапись (обновление) узла или создается новый узел с указанным ID (108)

Фиг. 1

2

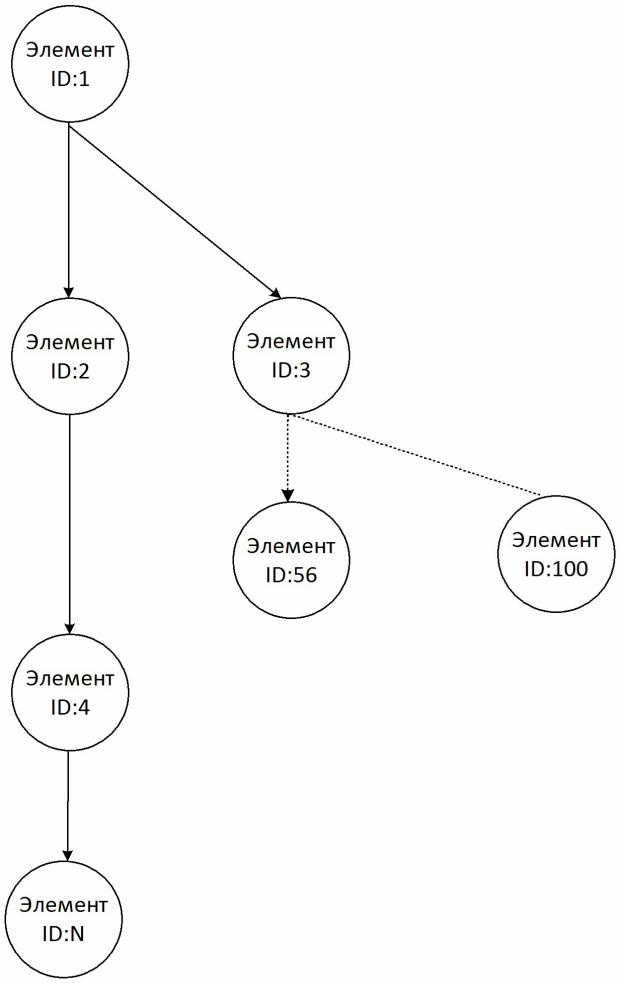

Преамбула
0 0 0 0 0 0 0 1
Длина целого числа в байтах
Бит расширения преамбулы
Целое число $2^8 - 1$
0 0 0 0 0 0 0 1
Длина 1 Байт

Фиг. 2


Преамбула
Целое число $2^{15} - 1$
Длина 2 Байта
Преамбула
Целое число $2^{1016} - 1$
Длина 127 Байт
Преамбула
Целое число $2^{1024} - 1$
Длина 128 Байт
Длина целого числа в байтах
Бит расширения преамбулы
Преамбула
Целое число $2^{262136} - 1$
Длина 32767 Байт

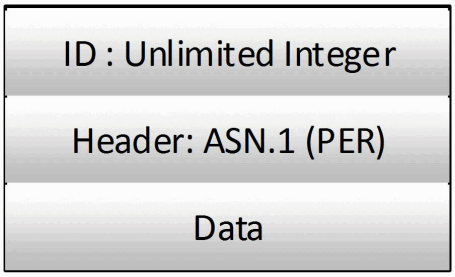
Фиг.3

3

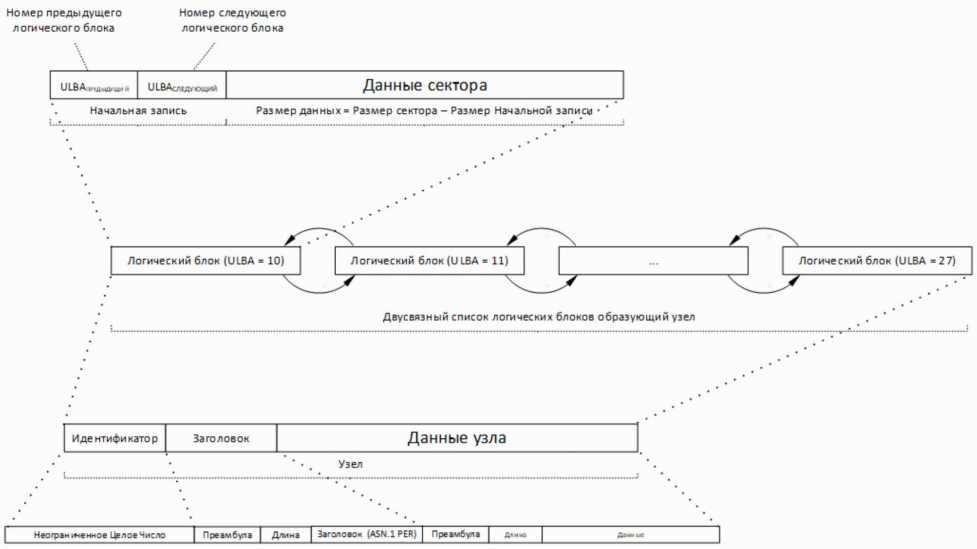

Элемент
ID:1
Элемент
ID:2
Элемент
ID:3
Элемент
ID:56
Элемент
ID:100
Элемент
ID:4
Элемент
ID:N

Фиг. 4

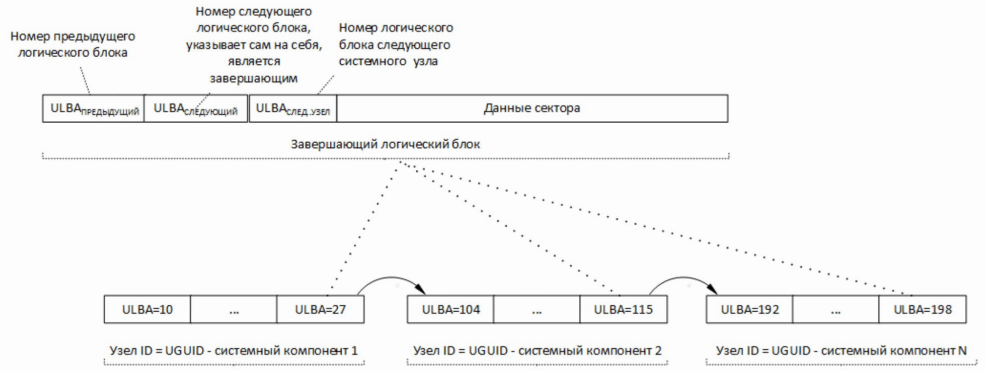
4


ID : Unlimited Integer
Header: ASN.1 (PER)
Data

Фиг. 5

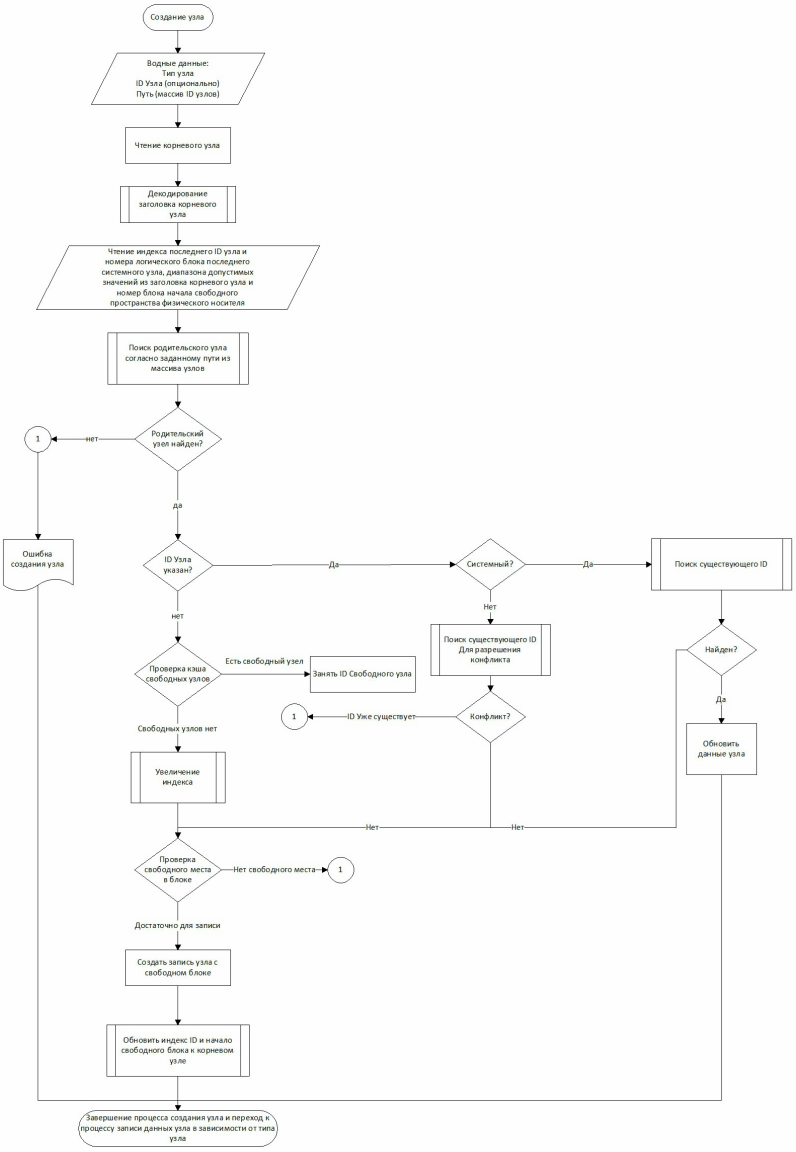

Номер предыдущего логического блока
Номер следующего логического блока
ULBAпредыдущий
ULBAследующий
Данные сектора
Начальная запись
Размер данных = Размер сектора – Размер Начальной записи
Логический блок (ULBA = 10)
Логический блок (ULBA = 11)
...
Логический блок (ULBA = 27)
Двусвязный список логических блоков образующий узел
Идентификатор
Заголовок
Данные узла
Узел
Неограниченное Целое Число
Преамбула
Длина
Заголовок (ASN.1 PER)
Преамбула
Длина
Данные

Фиг. 6


Номер предыдущего логического блока
Номер следующего логического блока, указывает сам на себя, является завершающим
Номер логического блока следующего системного узла
ULBA предыдущий
ULBA следующий
ULBA след.узел
Данные сектора
Завершающий логический блок
ULBA=10
...
ULBA=27
ULBA=104
...
ULBA=115
ULBA=192
...
ULBA=198
Узел ID = UGUID - системный компонент 1
Узел ID = UGUID - системный компонент 2
Узел ID = UGUID - системный компонент N

Фиг. 7

6


Создание узла
Водные данные:
Тип узла
ID Узла (опционально)
Путь (массив ID узлов)
Чтение корневого узла
Декодирование заголовка корневого узла
Чтение индекса последнего ID узла и номера логического блока последнего системного узла, диапазона допустимых значений из заголовка корневого узла и номер блока начала свободного пространства физического носителя
Поиск родительского узла согласно заданному пути из массива узлов
Родительский узел найден?
нет
1
Да
Ошибка создания узла
ID Узла указан?
Да
Системный?
Да
Поиск существующего ID
Нет
нет
Поиск существующего ID Для разрешения конфликта
Найден?
Проверка кэша свободных узлов
Есть свободный узел
Занять ID Свободного узла
Да
1
ID Уже существует
Конфликт?
Обновить данные узла
Свободных узлов нет
Увеличение индекса
Нет
Нет
Проверка свободного места в блоке
Нет свободного места
1
Достаточно для записи
Создать запись узла с свободном блоке
Обновить индекс ID и начало свободного блока к корневом узле
Завершение процесса создания узла и переход к процессу записи данных узла в зависимости от типа узла

Фиг. 8