ЧЕРНОВИК

Введение

Что такое файловая система? Файловую систему можно определить как способ организации, хранения и именования данных на носителях информации в компьютере или любом другом электронном оборудовании. Файловая система определяет формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов (в данном случае файл удобно определить как некоторую логически неделимую совокупность данных). Таким образом, файловая система осуществляет контроль за тем, как файлы сохраняются на физическом устройстве и извлекаются обратно для использования в работе других программ.

Без наличия файловой системы, вся информация размещалась бы на устройстве хранения данных произвольным образом, в виде одного непрерывного блока, без возможности определения, в каком месте заканчивается один блок информации и начинается другой. С помощью же логического разделения данных по отдельным частям и присвоения каждой такой части собственного имени (или идентификатора), информация может быть легко отделена и идентифицирована. Каждый такой блок информации получил название «файл», а набор структур данных и логических правил для управления и работы с файлами получил название «файловая система».

Основные функции любой файловой системы нацелены на решение следующих задач:

именование файлов;

программный интерфейс работы с файлами для приложений;

отображения логической модели файловой системы на физическую организацию хранилища данных;

организация устойчивости файловой системы к сбоям питания, ошибкам аппаратных и программных средств;

содержание параметров файла, необходимых для правильного его взаимодействия с другими объектами системы (ядро, приложения и пр.).

Файловая система не только реализует определенный набор логических правил, по которым файлы размещаются на устройстве, но также хранит дополнительную информацию о каждом файле, используемую для его однозначной идентификации (имя файла) и получения сведений о некоторых его свойствах (размер, время создания, время последней модификации и т.д.). Файловые системы управляют доступом к файлам и их метаданным.

Файловые системы могут использоваться на различных типах устройств. Каждое устройство хранения данных использует определенный тип физического носителя. Наиболее распространенными на сегодняшний день являются жесткие диски. Файловая система связывает носитель информации с одной стороны и API для доступа к файлам — с другой. Когда прикладная программа обращается к файлу, она не имеет никакого представления о том, каким образом расположена информация в конкретном файле, так же, как и на каком физическом типе носителя (CD, жёстком диске, магнитной ленте, блоке флеш-памяти или другом) он записан. Всё, что знает программа — это имя файла, его размер и атрибуты. Эти данные она получает от драйвера файловой системы. Именно файловая система устанавливает, где и как будет записан файл на физическом носителе (например, жёстком диске).

С точки зрения операционной системы, весь диск представляет собой набор кластеров (как правило, размером 512 байт и больше). Драйверы файловой системы организуют кластеры в файлы и каталоги (реально являющиеся файлами, содержащими список файлов в этом каталоге). Эти же драйверы отслеживают, какие из кластеров в настоящее время используются, какие свободны, какие помечены как неисправные.

A file system driver is the software component that interprets the structures of the file system and presents a logical hierarchical view to the applications using it. It can be a part of the operating system, or come as a third party module

Однако файловая система не обязательно напрямую связана с физическим носителем информации. Существуют виртуальные файловые системы (например, procfs), файлы и каталоги внутри которых генерируются при запросе, а также сетевые файловые системы (напрмер, NFS – Network File System, SMB – Server Message Block), которые являются лишь способом доступа к файлам, находящимся на удалённом компьютере.

Таким образом, по предназначению файловые системы можно классифицировать на нижеследующие категории.

Для носителей с произвольным доступом (например, жёсткий диск): FAT32, HPFS, ext2 и др. Поскольку доступ к дискам в несколько раз медленнее, чем доступ к оперативной памяти, для прироста производительности во многих файловых системах применяется асинхронная запись изменений на диск. Для этого применяется либо журналирование, например в ext3, ReiserFS, JFS, NTFS, XFS, либо механизм soft updates и др. Журналирование широко распространено в Linux, применяется в NTFS. Soft updates — в BSD системах.

Для носителей с последовательным доступом (например, магнитные ленты): QIC и др.

Для оптических носителей — CD и DVD: ISO9660, HFS, UDF и др.

Виртуальные файловые системы: AEFS и др.

Сетевые файловые системы: NFS, CIFS, SSHFS, GmailFS и др.

Для флэш-памяти: YAFFS, ExtremeFFS, exFAT.

Немного выпадают из общей классификации специализированные файловые системы: ZFS (собственно файловой системой является только часть ZFS), VMFS (т. н. кластерная файловая система, которая предназначена для хранения других файловых систем) и др.

В многопользовательских системах появляется ещё одна задача: защита файлов одного пользователя от несанкционированного доступа другого пользователя, а также обеспечение совместной работы с файлами, к примеру, при открытии файла одним из пользователей, для других этот же файл временно будет доступен в режиме «только чтение».

Некоторые файловые системы были разработаны специально для решения определенного класса задач и использовании в работе конкретных приложений. Так, например, ФС ISO 9660 была спроектирована исключительно для оптических дисков.

Таким образом, Существует множество различных видов ФС. Каждая ФС имеет определенную логическую структуру, характеризуется скоростью работы, гибкостью, надежностью, способностью ксли вн восстановлению после возникновения кртических ошибок, сбоев и отказов оборудования, предоставляемой безопасностью доступа к файлам, требуемым размером на диске, возможностью гибкой настройки в зависимости от используемого физического носителя данных являются ключевыми факторами и т.п.

Файловые системы операционной системы Linux

Основное ядро Linux имеет встроенную поддержку широкого спектра различных файловых систем. Рассмотрим наиболее популярные и известные из них:

Ext

Первая файловая система, разработанная специально для нужд операционной системы Linux. Была создана в 1992 с целью преодолеть ограничения существовавшей в то время файловой системы Minix. На сегодняшний день файловая система ext является довольно устаревшей и уже не поддерживается во многих дистрибутивах Linux.

Ext2

Нежурналируемая файловая система, представляет собой продолжение развития файловой системы ext. Включена поддержка расширенных файловых атрибутов, увеличен максимальный поддерживаемый размер физического устройства (до 2 терабайт). Поскольку журналирование не используется, в процессе своей работы данная файловая система осуществляет существенно меньше операций записи на диск, вследствие чего широко используется в USB-устройствах с флеш-памятью.

Ext3

Дальнейшее развитие ext\* файловых систем привело к появлению журналируемой файловой системы ext3. По сути является расширением файловой системы ext2, способное к журналированию. Одной из главных целей при разработке данной файловой системы была поддержка обратной совместимости. Вследствие переход с файловой системы ext2 к файловой системе ext3 не требует форматирования. На сегодняшний день ext3 является наиболее стабильной и поддерживаемой файловой системой в среде Linux.

Ext4

Представляет собой своеобразную попытку создать 64-х битную ext3, способную поддерживать больший размер файловой системы. Позже добавились возможности — непрерывные области дискового пространства, задержка выделения пространства, онлайн дефрагментация и прочие. Обеспечивается прямая совместимость с системой ext3 и ограниченная обратная совместимость при недоступной способности к непрерывным областям дискового пространства.

ReiserFS

Первая попытка создать файловую систему нового поколения для Linux. Данная файловая система, представленная в 2001 году, включает в себя журналирование, возможность динамического масштабирования и лишена многих недостатков, присутствующих в файловых системах семейства ext. Данная файловая система показывает прекрасную производительность при работе с маленькими файлами и логами, поэтому идеально подходит для использования в базах данных и почтовых серверах. Существует также файловая система Reiser4, являющаяся преемником ReiserFS, в которой реализованы такие технологии как транзакции, задержка выделения пространства, а также встроенная возможность кодирования и сжатия данных. Однако развитие Reiser4 происходит слишком медленно и она до сих пор не поддерживается основным ядром Linux.

BtrFS

Файловая система, изначально разрабатываемая компанией Oracle, основана на структурах B-деревьев и работает по принципу копирование-при-записи. Предоставляет такие возможности как снимки областей диска (снапшоты), контроль за целостностью данных и метаданных, прозрачное сжатие данных, журналирование, поддержка оптимизированного режима при использовании в SSD-накопителях, дефрагментация в рабочем режиме и многое другое. Немаловажным фактом является возможность перехода с файловых систем ext3 и ext4 на BtrFS без необходимости форматирования. Данная файловая система лишена многих недостатков, присущим предшествующим файловым системам операционной сисиемы Linux, и призвана стать файловой системой Linux по умолчанию.

XFS

Файловая система, разработанная компанией Silicon Graphics в 1994 году и портированная на Linux в 2001 году. Во многом похожа на файловую систему ext4, реализует журналирование, задержку выделения пространства (как метод борьбы с фрагментацией), онлайн дефрагментацию, обладает возможностью динамического расширения. Показывает хорошую производительность при работе с большими файлами, но в остальном не обладает существенными преимуществами по сравнению с ext4.

JFS

Файловая система, разработанная компанией IBM в 1990 году и позже портированная на Linux. Главными особенностями данной файловой системы является низкий уровень потребления процессорного времени и хорошая производительность при работе с файлами как большого, так и малого размера. Как и файловые системы XFS и ReiserFS, предоставляет возможность динамического масштабирования, поддерживает журналирование. Вследствие низкого уровня использования процессора, хорошо подходит для применения в работе маломощных компьютеров и серверов.

ZFS

Файловая система, разработанная компанией Sun Microsystems, первоначально для операционной системы Solaris.Среди отличительных особенностей можно выделить отсутствие фрагментации данных как таковой, возможность по управлению снапшотами, пулами хранения. ZFS показывает прекрасные результаты производительности при работе с большими дисковыми массивами. Поскольку данная файловая система является проприетарной, она не может быть в основное ядро Linux. Тем не менее, в Linux системах ZFS может использоваться посредством механизма FUSE (Filesystem in userspace – файловая система в пользовательском пространстве).

Помимо рассмотренных выше файловых систем, основное ядро Linux поддерживает большое количество других файловых систем (<http://en.wikipedia.org/wiki/Category:Linux_kernel-supported_file_systems>)

**Файловые системы операционной системы Windows**

Операционная система Windows позволяет использовать следующие файловые системы:

<http://en.wikipedia.org/wiki/File_system#Microsoft_Windows>

FAT

Классическая архитектура файловой системы, которая из-за своей простоты всё ещё широко используется для флеш-накопителей и карт памяти. Данное семейство файловых систем (8-bit FAT, FAT-12, FAT-16, FAT-32) поддерживается почти всеми операционными системами для персональных компьютеров, включая все версии операционной системы Windows и MS-DOS, и является в некотором смысле универсальной файловой системой, предназначенной для обмена данными между компьютерами и устройствами практически любого типа.

<https://ru.wikipedia.org/wiki/FAT>

NTFS

NTFS заменила использовавшуюся в MS-DOS и Windows файловую систему FAT. Используется в семействе операционных систем Windows, начиная с Windows-NT, появившейся в 1993 году. NTFS поддерживает систему метаданных и использует специализированные структуры данных для хранения информации о файлах для улучшения производительности, надёжности и эффективности использования дискового пространства. NTFS имеет встроенные возможности разграничения доступа к данным для различных пользователей и групп пользователей (списки контроля доступа — Access Control Lists, ACL), а также позволяет назначать квоты (ограничения на максимальный объём дискового пространства, занимаемый теми или иными пользователями). NTFS использует систему журналирования для повышения надёжности файловой системы. Также среди особенностей данной файловой системы можно выделить поддержку жестких ссылок, разреженных файлов, шифрования, сжатия данных, точке повторной обработки (специальный тип каталогов, используемых как точки монтирования других файловых систем, символических ссылок, ссылок удаленных файловых систем).

<https://ru.wikipedia.org/wiki/NTFS>

exFAT

Проприетарная файловая система, является преемником файловой системы FAT-32. Разработана компанией Microsoft преимущественно для мобильных носителей таких как USB флешки, SSD-диски, смарткарты. Позволяет хранить файлы и использовать разделы значительно большего размера, чем в файловых системах семейства FAT. В системе exFAT также появилась возможность управления правами доступа на файлы/каталоги, а время доступа к данным уменьшилось. exFAT можно считать конкурентом NTFS на системах с ограниченной вычислительной мощности и памяти. Поскольку файловую систему NTFS на флэш-картах использовать очень неудобно и неэффективно (падает скорость работы и уменьшается срок службы флеш памяти), именно exFAT является идеальной файловой системой для таких устройств. Поддерживается в Windows XP, Windows Vista, Windows 7, Windows 8.

<http://house-computer.ru/soft/624-sravnenie-fat32-ntfs-exfat-na-fleshkax-i-vneshnix.html>

<http://en.wikipedia.org/wiki/ExFAT>

Буква диска

Для доступа к дисковым разделам и отличия одного от другого в ОС Windows в пользовательском режиме используется абстракция «символ диска». Каждому разделу физического устройства присваивается уникальный символ (например, «С») и для доступа к файлу, размещенному по некоторому пути «path» относительно корня файловой системы на данном разделе, используется абсолютный путь вида «C:\path». В отличие от концепции точек монтирования, используемых в системах Unix, в которой все разделы расположены в едином иерархическом пространстве имен, присвоение буквы диска позволяет использовать несколько пространств имен, где каждый раздел имеет собственное независимое дерево файлов.

<http://en.wikipedia.org/wiki/Drive_letter_assignment>

Роль файловой системы в операционной системе

Файловая система является ключевым компонентом операционной системы, и, как правило, каждая операционная система предоставляет для своей работы нативную для неё файловую систему и изначально поддерживает лишь определённый набор файловых систем.

В настоящее время IT-индустрия перегружена многообразием файловых систем. Каждая файловая система обладает своими преимуществами и недостатками и, как правило, несовместима с другими файловыми системами.

Помимо операционных систем, можно выделить следующие типы программного обеспечения, которым требуется работать с различными типами файловых систем.

Системы резервного копирования, восстановления и защиты данных

Менеджеры жестких дисков и разделов

Файловые менеджеры

Можно выделить определенные типы программного обеспечения, которым требуется работать с различными типами файловых систем

В случае если поддержка файловый системы не реализована на уровне операционной системы, это задача перекладывается на программное обеспечение

На основе вышесказанного, можно утверждать, что существует достаточно много программ, которым необходимо работать с различными вилами файловых систем. Возможность работы с определенными типами файловых систем иногда оказывается решающим фактором при выборе той или иной операционной системы. Однако не всегда возможно или удобно отказаться от использования одной ОС в пользу другой лишь для возможности использовать желаемую файловую систему. В такой ситуации возникает необходимость реализовать поддержку в данной операционной системе ненативной для неё файловой системы.

Типы программного обеспечения

В данной работе исследуется возможность поддержки файловых систем Linux в операционной системе Windows.

В настоящее время IT-индустрия переполнена многообразием существующих файловых систем.

A file system is a crucial component of any operating system and operating systems can mostly only deal with their native one. Since almost every OS has its own file system, the present day IT environment is overwhelmed with different types of file systems and the incompatibility between them. To be competitive and versatile, your device desperately needs a bridge between different file systems.

Существующие решения

Чтение/запись

Ext2fsd – драйвер ОС Windows для доступа к Ext2, Ext3 и Ext4 файловым системам. Предоставляет возможность работать с перечисленными ФС нативно, обеспечивая доступ к ФС через символ диска (drive letter), что позволяет любой программе работать с данными ФС.

Основные особенности:

Поддержка чтения и записи ext2/ext3 файловых систем

Поддержка чтения ext4 файловой системы

Отсутствие полной поддержки журналирования ext3 ФС

Отсутствие поддержки зашифрованных файловых систем семейства ext\*

Отсутствие поддержки менеджера логических томов ОС Линукс и RAID-массивов Линукс

Поддерживаемые версии ОС Windows: Windows XP, Windows Vista, Windows 7

Отзывы: чтение ext\* файловых систем работает стабильно, при записи в редких случаях возможны потери данных

<http://sourceforge.net/projects/ext2fsd/reviews>

Ext2 Installable File System – драйвер ОС Windows для доступа, чтения и записи ext2 файловой системы. Представляет собой полноценный драйвер ФС режима ядра, работающий на том же программном уровне, что и остальные нативные драйыера Windows (NTFS, FASTFAT, CDFS). Разделы с ext2 файлвоой системой получают букву диска, что позволяюет всем приложениям получать прямой доступ к ним.

Поддерживаемые ОС: Windows NT/2000/XP/2003/Vista

Источник: <http://www.fs-driver.org/>

Paragon ExtFS for Windows – ПО компании Paragon, обеспечивающее полный нативный доступ к ext2/3/4 файловым системам.

Особенности:

Поддержка чтения LVM разделов

Инструмент для создания\форматироавния Linux-разделов.

Поскольку работает в режиме ядра, обеспечивает хорошую производительность и скорость работы

DiskInternals Linux Reader – GUI-приложение ОС Windows для обзора содержимого и чтения файловых систем ОС Линукс. Поддерживает просмотр содержимого данных ФС с помощью графического интерфейса как в Windows Explorer (возможность тсползования Windows Explorer для просмотра содержимого). Поддерживает чтение таких файловых систем как ext2, ext3, RaizerFS, Reizer4, а также HFS+ операционной системы Apple OS. Не предоставляет доступ к ФС через символ диска, что делает невозможным её использование в разрабатываемых программах. Поэтому для работы с указанным файлом необходимо предварительно скопировать его в файловую систему, нативную для ОС Windows.

Поскольку поддерживется доступ только на чтение, гарантируется, что файловые системы Линукс не повредятся.

Ext2explore – GUI-приложение ОС Windows, по характеристикам аналогичное предыдущему рассмотренному DiskInternalsLinuxReader. Из особенностей стоит упомянуть возможность чтения ext4 файловой системы

Ext2Read – Windows Explorer-подобное графические приложение для просмотра и чтения содержмого ext2/3/4 файловых систем. Поддреживает LVM2 и расширение ext4. Обуспечиввет возможность рекурсивного копирования директорий с файлами.

Explore2fs – графическое приложения для просмотра содержимого ext2/3 файловых систем. Поддерживает только чтение и копирование даннх.

LTOOLS – утилита коммандной строки для чтения и записи ext2/3 файловых систем, а также ReiserFS файловой системы.

Источник: <http://www.howtogeek.com/112888/3-ways-to-access-your-linux-partitions-from-windows/>

<http://www.askvg.com/how-to-access-linux-partitions-ext2-ext3-from-windows-in-dual-boot-system/>

все официальные сайты этого ПО

Как видно из примеров, среди всех решений проблемы доступа к файловым системам ОС Linux внутри ОС Windows можно выделить два основных подхода: независимое приложение ОС Windows либо полноценный драйвер, работающий в режиме ядра, обеспечивающий нативный доступ к файловой системе.

Либо полноценные драйвера, работающие в режиме ядра операционной системы Windows, предоставляющие возможность осуществлять чтение и запись на диск с файловой системой из-под любой программы, но надежность некоторых из которых зачастую оставляет желать лучшего. Ошибки, допущенные при разработке такого драйвера, в процессе его использования могут привести к самым нежелательным последствиям: от ошибки времени выполнения операционной системы и нежелательной незапланированной перезагрузки до отказа и повреждения оборудования с частичной или полной потерей данных.

Отдельные графические приложения и утилиты командной строки как правило предоставляют доступ к файловой системе только в режиме чтения для просмотра и копирования её содержимого. При этом отсутствует возможность доступа к данным из других программ. При использовании данного подхода для работы с файлами внутри файловой системы из других программ необходимо заранее вручную скопировать необходимые файлы на раздел диска с нативной для Windows файловой системой, после чего файл становится виден внутри любой программы, использующей путь, начинающийся с символа данного раздела (drive letter).

Use BTRFS:

coLinux

zfs-win : read-only <https://code.google.com/p/zfs-win/>

win-btrfs

virtual machine + network server

Файловые системы: значение, функции, виды

Windows ФС. Устройство, виды

Linux ФС. Многообразие, сравнение с Windows, необходимость поддержки. Наличие возможности Linux -> Windows, но отсутствие Windows -> Linux.

НЕОБХОДИМОСТЬ использования ФС Linux из-под Windows. Типы ПО, требующие (либо реализующие для своих нужд) данную функциональность.

Существующие решения. Основные подходы. Self-contained приложения, драйвера, read-only, виртуальные машины coLinux. Пример с хабра, возгласы пользователей.

Выбор подхода с виртуальной машиной. Обоснование. Libguestfs.

Необходимость поддержки ненативных файловых систем

В IT-индустрии на сегодняшний день можно выделить целые классы программного обеспечения, которым в силу своей специфики необходимо поддерживать работу с широким спектром разнообразных файловых систем. К ним относятся:

Системы резервного копирования и восстановления данных

Системы защиты данных, антивирусное программное обепечение

Менеджеры жестких дисков и разделов

Файловые менеджеры – программы для просмотра и управления содержимым файловой системы, реализующие возможность копирования, перемещения, удаления файлов и каталогов внутри файловой системы

Чем с большим количеством файловых систем способна работать программа – тем более она привлекательна в использовании

Для организации работы с файловой системы необходим так называемый «драйвер» файловой системы. Драйвер файловой системы можно охарактеризовать как программный компонент, интерпретирующий структуры файловой системы и предоставляющий логический иерархический вид использующим его приложениям. Он может являться частью операционной системы либо поставляться сторонним производителем как отдельный модуль.

Для возможности работы с нативными для данной операционной файловыми системами отдельному приложению не нужно прилагать никаких усилий. Операционная система изначально имеет встроенные в ядро драйвера, организующие доступ к нативным файловым системам. Как правило, каждый драйвер для нативной файловой системы пишется и тестируется разработчиками этой операционной и файловой системы, обладают высокой производительностью, являются достаточно надежными, эффективными, активно поддерживаются в течение всего времени существования файловой системы как нативной для данной операционной системы.

В случае, если программе требуется работать с ненативной для данной операционной системы файловой системы, задача реализации драйвера файловой системы ложится на плечи разработчиков данной программы.

Можно выделить 2 принципиально отличающихся друг от друга подхода к решению данной проблемы:

Использование нативного драйвера файловой системы, то есть драйвера, реализованного специально для работы в целевой операционной системы.

Использование драйвера файловой системы, написанного для той операционной системы, в которой данная файловая система является нативной

В рамках 1-ого подхода можно выделить следующие возможные решения:

Реализация нативного драйвера файловой системы «с нуля»

Использование драйвера файловой системы, реализованного сторонними разработчиками

Использование приложений, реализующие базовые операции работы с файловой системой: просмотр содержимого, чтение и запись файлов

Реализация п

Отличительной особенностью современных файловых систем является их сложность. Новые файловые системы должны обеспечивать высокую производительность и эффективность работы при минимальных затратах, отвечать высоким требованиям надежности и безопасности данных, предоставлять возможность эффективного управления и восстановления данных. Каждая вновь появившаяся файловая система должна обладать лучшими характеристиками по сравнению с уже существующими, чтобы быть конкурентоспособной и привлекательной для конечного пользователя.

Все это прямым образом влияет на техническую сложность реализации эффективных драйверов файловой системе. Разработка драйвера представляет собой достаточно трудоемкий процесс, требующий больших затрат ресурсов и времени. Реализация надежного и эффективного драйвера предполагает не только глубокое изучение внутреннего устройства и анализ принципов работы файловой системы, но значительных затрат на тестирование с целью выявления недоработок, багов и прочих ошибок в реализации драйвера.

При выборе данного решения также необходимо постоянно следить за изменения файловой системы и своевременно обновлять реализацию драйвера с целью его поддержания в актуальном состояние.

При использовании драйвера файловой системы, реализованного сторонними разработчиками, решается множество проблем, рассмотренных выше. Здесь возможны два варианта: использование либо свободно распространяемой драйвера (с открытым исходным кодом) либо драйвера, реализованного коммерческой организацией.

В большинстве случаев, качество драйверов с открытым исходным кодом оставляет желать лучшего. Как правило, разработка таких драйверов осуществляется непрофессиональными программиста и чаще всего носит чисто экспериментальный характер. В таких драйверах реализуются лишь некоторые базовые особенности фацловой системы, многое остается нереализоанным. Крайне сложно найти хорошую реализацию драйвера файловой системы, а иногда её просто не существует. В таком случае, проблему доработки, тестирования и поддержки данного драйвера приходить решать самим пользователям данного драйвера.

Коммерческие драйвера файловой системы лишены рассмотренных выше недостатков, но, очевидно, требует материальных затрат на покупку лицензии и поддержки со стороны разработчиков. В случае необходимости поддерживать большой набор файловых систем данный подход может оказаться невыгодным.

Существует также отдельные приложения, реализующие доступ к некоторым файловым системам. Такие приложения чаще всего выступают в качестве файловых обозревателей, позволяя лишь просматривать содержимое файловой системы, осуществлять чтение файлов. Главным недостатком при использовании таких программ является невозможность работать с файловой системой из других программ. Для полноценной работы с содержимым файловой системы из других программ требуется вручную скопировать интересующие файлы на нативную файловую систему и уже использовать их копии в программе с помощью доступа к нативной файловой системе, реализованной самой операционной системой.

Использование нативных Linux-драйверов

Нативные драйвера файловой системы лишены многих недостатков, присущим их портированным аналогам. Большинство современных файловых систем изначально разрабатываются для использования в какой-то конкретной операционной системе. Соответственно, сами авторы файловой системы реализуют для неё драйвер. Такие драйвера, написанные командой профессиональных разработчиков, обладает высоким качеством и надежностью. Проблема тестирования, своевременного обновления снимается с пользователя. Такие драйвера полностью поддерживают все особенности и свойства файловой системы, показывают более высокие результаты производительности по сравнению с портированными версиями сторонних разработчиков.

Однако такие драйвера невозможно взять и просто использовать в другой операционной системе – для этого необходимо в целевой операционной системе создать окружение, эмулирующее окружение нативной для данного драйвера операционной системы.

Таким образом, с одной стороны, проблема поиска надежного и эффективного драйвера оказывается полностью решенной, но, с другой стороны, возникает не менее тривиальная задача – эмуляции окружения одной операционной системы внутри другой.

Данную проблему можно решить следующими способами:

Портировать исходный код ядра операционной системы для запуска внутри другой как отдельного приложения

Использовать виртуальную машину

Очевидно, что сложность реализации первого способа колосальна и намного выше, чем разработка портированного драйвера файловой системы. Разработка современной операционной системы (такой как Linux) представляет собой одну из самых сложных и трудоемких задач, существующих на данный момент в IT-индустрии, а потому и портирование операционной системы представляет собой не менее сложную задачу.

Использование виртуальной машины позволит использовать нативную для драйвера операционную систему без изменений. Несмотря на то, что при реализации первого подхода мы потенциально может получить менее ресурсоемкую, а как следствие, более производительное решение, использвоание виртуальной машины выглядит наиболее привлекательным с точки зрения потраченных усилий и получаемых выгод.

Именно такой подход был выбран в данной работе для реализации работы с файловыми системами операционной системы Linux в операционной системе Windows.

Архитектура разрабатываемой библиотеки

К вновь появляющимся файловым системам предъявляются

Использовать нативный драйвер файловой системы для той операционной системы, в которой эта файловая система является нативной.

Возможность работы с ненативными файловыми системами в операционной системе

В нередких случаях невозможно отказаться от использования той или иной операционной системы, однако некоторым программам необходимо работать с файловыми системами, не являющимися нативными для данной операционной системы. В этом случае используемое программное обеспечение должно само организовать доступ к данной файловой системе. Для этого необходимо использовать драйвер файловой системы, способный работать в используемой операционной системе.