ЧЕРНОВИК

В данной работе затрагивается проблема работы с файловыми системами, нативными для операционной системы Linux, в операционной системе Windows. Рассмотрим основные подходы, существующие на данный момент решения по

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

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

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

Введение

**Файловая система. Определение, функции, виды**

Что такое файловая система? Файловую систему можно рассматривать 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 размещением файлов 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. Среди них можно выделить носители с произвольным доступом такие как жесткий диск, носители с последовательным доступом (магнитные ленты), оптические носители, флэш-память и другие. Наиболее распространенными на сегодняшний день являются жесткие диски. Файловая система связывает носитель информации с одной стороны и программный интерфейс (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.

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.

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, а VMFS представляет собой кластерную файловую систему, предназначенную для хранения других файловых систем) и др.

На сегодняшний день IT-индустрия «перегружена» огромным количеством разнообразных файловых систем. Каждая файловая система имеет свою определенную логическую структуру, характеризуется скоростью работы, надежностью и способностью к восстановлению после возникновения критических ошибок, сбоев и отказов оборудования, предоставляемой безопасностью одновременного доступа к файлам, занимаемым размером на физическом устройстве, возможностью гибкой настройки в зависимости от используемого физического носителя данных и т.д.

Таким образом, каждая файловая система имеет свои преимущества и недостатки по сравнению с другими файловыми системами. Выбор той или иной файловой системе зависит от тех задач, в решение которых она будет использоваться. Например, в случае почтового сервера или баз данных хорошо подойдет файловая система, спроектированная специально для работы с файлами малого размера, а при использовании в маломощных компьютерных установках отличным выбором будет использование файловой системы, потребляющей малое количество процессорного времени.

**Нативные файловые системы**

Файловая система является ключевым компонентом любой операционной системы и зачастую играет решающее значение при выборе той или иной операционной системы.

Для каждой операционной системы существует набор файловых систем, поддержка которых изначально включена в ядро операционной системы. Такие файловые системы являются нативными для данной операционной системы. Многие нативные файловые системы разрабатывались специально для нужд конкретной операционной системы. Наиболее известными примерами такого рода файловых систем являются файловая система ext4 операционной системы Linux, файловая система NTFS операционной системы Windows, файловая система HFS операционной системы Mac OS и многие другие.

Количество нативно поддерживаемых файловых систем сильно варьируется в зависимости от рассматриваемой операционной системы. Так, например, основное ядро операционной системы Linux поддерживает широкий спектр разнообразных файловых систем, в то время как в операционной системе Windows включена поддержка сравнительно небольшого количества файловых систем. Рассмотрим их более подробно.

**Файловые системы операционной системы Linux**

За всё время существования и эволюции операционной системы Linux в её основное ядро была добавлена поддержка огромного количества разнообразных систем. Примерный перечень поддерживаемых файловых систем приведен в [?]. Рассмотрим наиболее популярные и известные из них.

Семейство Ext\* файловых систем

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 устаревшей и уже не поддерживается во многих дистрибутивах Linux.

Ext2

Нежурналируемая файловая система, представляет собой продолжение развития файловой системы Ext2. 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 хорошо подходит для использования в устройствах с флеш-памятью.

Ext3

Дальнейшее развитие семейства 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 ‘njuj 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. Также добавились такие возможности как 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 встроена возможность кодирования и сжатия данных. К сожалению, 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 включена в основное ядро Linux. Тем не менее, в Linux системах ZFS может использоваться посредством механизма FUSE (Filesystem in userspace – файловая система в пользовательском пространстве).

(<http://en.wikipedia.org/wiki/Category:Linux_kernel-supported_file_systems>)

**Файловые системы операционной системы Windows**

По сравнению с операционной системой Linux, операционная система Windows нативно поддерживает существенно меньшее количество файловых систем. К ним относятся FAT, NTFS, exFAT, ReFS.

<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. 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 ранее использовавшуюся в MS-DOS и Windows файловую систему FAT. Впервые появилась в операционной системе Windows-NT, выпущенной 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 раздела от другого в пользовательском режиме в операционной системе Windows 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>

**Поддержка ненативных файловых систем**

Многим программам, работающим в некоторой операционной системе, не всегда достаточно поддерживаемого операционной системой набора файловых систем. Существуют целые классы программного обеспечения, которым в силу своей специфики требуется работать с широким количеством файловых систем. К ним относятся:

* Системы резервного копирования и восстановления данных
* Системы защиты данных, антивирусное программное обеспечение
* 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

Также зачастую при решении той или иной задачи может быть наиболее выгодно использовать конкретную файловую систему, например, из-за предоставляемых её определенных преимуществ, даже если выбранная файловая система не является нативной в используемой операционной системе. При этом не всегда удобно отказаться от используемой операционной системы в пользу той, в которой требуемая файловая система поддерживается в качестве нативной.

Каким же образом организовать работу с файловой системой в операционной системе, в которой она не является нативной?

Прежде всего стоит упомянуть, что для организации работы с любой файловой системы необходим так называемый «драйвер» файловой системы. Драйвер файловой системы можно охарактеризовать как некоторый программный компонент, интерпретирующий структуры файловой системы и предоставляющий использующим её приложениям логический иерархический вид её содержимого. Драйвер может являться частью операционной системы (в случае нативных файловых систем) либо поставляться сторонним производителем как отдельный программный модуль.

Для возможности работы с нативной файловой системой отдельному приложению не нужно прилагать никаких усилий. Как уже упоминалось выше, операционная система изначально имеет встроенные в её ядро драйвера, организующие полный доступ к нативным файловым системам, включающий в себя возможность чтения, записи, модификации содержащихся в файловой системе каталогов и файлов. Стоит отметить, что нативные драйвера файловых систем, как правило, пишутся, тестируются и поддерживаеются самими разработчиками операционной системы, в которой данная файловая система является нативной. Вследствие этого, такие драйвера обладают высокой надежностью, производительности, эффективно реализуют все возможности файловой системы, своевременно обновляются и активно поддерживаются разработчиками в течение всего времени существования файловой системы как нативной.

Как и в случае нативных файловых систем, для предоставления приложению возможности работать с файловой системой, не являющейся нативной, необходимо использовать драйвер этой файловой системы. Таким образом, задача поддержки ненативной файловой системы ложится на плечи разработчиков приложения (программы).

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

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

Использование портированного драйвера файловой системы, то есть драйвера, реализованного специально для работы в целевой операционной системе – в той, в которой необходимо реализовать поддержку данной файловой системы.

Исследуем оба подхода более детально на примере операционных систем Windows и Linux. При этом операционная система Windows будет выступать в качестве целевой и основное внимание будет сосредоточено на решении проблемы поддержки нативных файловых систем операционной системы Linux в операционной системе Windows.

**Использование портированного драйвера**

В рамках данного подхода существуют следующие возможные решения:

Рассмотрим каждое решение в отдельности и приведем существующие примеры каждого из решений при наличии таковых

*Реализация драйвера файловой системы «с нуля»*

Отличительной особенностью подавляющего числа современных файловых систем является их высокая сложность, непростое внутреннее устройство, использование в свой работе трудных и запутанных алгоритмов. Это объясняется тем, что к вновь появляющимся файловым системам предъявляются высокие требования по скорости доступа к данным, эффективному использованию ресурсов, предоставляемой безопасности и многие другие. Каждая новая файлов система должна быть производительнее своих предшественников, более эффективно решать поставленные перед ней задачи, чтобы быть конкурентоспособной и более привлекательной для конечного пользователя.

Сложность внутренней организации современных файловых системы и используемых ими в своей работе алгоритмов оказывают прямое влияние на сложность реализации полноценного драйвера файловой системы. Разработка драйвера современной файловой системы, например, такой как BtrFS, представляет собой достаточно трудоемкий процесс, требующий больших затрат ресурсов и времени. Для реализации надежного и эффективного драйвера требуется не только глубокое изучение внутреннего устройства файловой системы и принципов её работы, но и всестороннего знание особенностей операционной системы, в которой он будет работать. Стоит также упомянуть о необходимости длительного тестирования правильной работоспособности полученного драйвера перед его использованием с целью выявления допущенных при его разработке ошибок и недоработок. Также необходимо постоянно следить за изменениями файловой системы и своевременно обновлять созданную версию драйвера.

Таким образом, одним из главных недостатков данного решения является его высокая сложность и трудоемкость, обремененная необходимостью постоянно поддерживать реализованный драйвер в актуальном состоянии при обновлениях файловой системы, а в случае необходимости одновременной поддержки нескольких файловых систем затраты на реализацию данного решения возрастают пропорционально количеству поддерживаемых файловых систем. К преимуществам же можно отнести потенциально высокую производительность работы полученного драйвера, полностью реализующего всю функциональность файловой системы, при минимально возможном потреблении системных ресурсов,

*Использование драйвера файловой системы, реализованного сторонними разработчиками*

При выборе данного способа в зависимости от «происхождения» драйвера возможны два варианта: использование свободно распространяемого драйвера либо использование драйвера, реализованного коммерческой организацией.

В большинстве случаев, качество и надежность свободно распространяемых драйверов оставляет желать лучшего. Как правило, разработка таких драйверов осуществляется непрофессиональными программистами и чаще всего носит чисто экспериментальный характер. Зачастую в таких драйверах реализуются лишь некоторые базовые функции файловой системы, многое остается недоделанным и недоработанным. Также характерной особенностью таких драйверов является отсутствие тщательного тестирования и поддержки со стороны разработчиков. 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. Таким образом, задачу доработки, тестирования и своевременного обновления драйвера необходимо решать пользователю данного драйвера. Так, например, на данный момент для некоторых файловых систем операционной системы Linux крайне сложно найти хорошую реализацию полноценного драйвера для операционной системы Windows, а иногда его просто не существует.

Коммерческие драйвера файловой системы, как правило, лишены рассмотренных выше недостатков, связанных со свободно распространяемыми драйверами. Главным же их недостатком, очевидно, является необходимость материальных затрат на покупку лицензии для использования драйвера и поддержки со стороны разработчиков. В случае необходимости полноценного доступа к большому набору файловых систем данное решение может оказаться слишком невыгодным.

Использование отдельных приложений и утилит, реализующих частичный или полный доступ к файловой системе

Существуют отдельные приложения для операционной системы, реализующие доступ к некоторым файловым системам, не являющимися нативными. Чаще всего такие приложения реализуют ограниченную функциональность и выступают в качестве файловых обозревателей, позволяя лишь просматривать содержимое файловой системы, осуществлять чтение и копирование файлов. Главным недостатком при использовании таких программ является невозможность работать с файловой системой из других программ. Для полноценной работы с содержащимися в файловой системы файлами и каталогами требуется вручную скопировать интересующие файлы на поддерживаемую операционной системой файловую систему и использовать в последующей работе только их копии. В качестве преимущества данного способа можно указать на тот факт, что поскольку доступ к файловой системе выполняется в режиме чтения, гарантируется, что содержащиеся данные внутри файловой системы не будет повреждены, и сама файловая система останется в неизменном работоспособном состоянии.

Поскольку в данной работе рассматривается возможность работы с файловыми системами операционной системы Linux в операционной системе Windows, приведем существующие на данный момент примеры рассмотренных выше решений.

**Примеры**

Ext2fsd

Свободно распространяемый драйвер операционной системы Windows для доступа к Ext2, Ext3 и Ext4 файловым системам. Предназначен для использования в Windows XP, Windows Vista, Windows 7. Предоставляет возможность работать с перечисленными файловыми системами нативно, обеспечивая другим приложениям доступ к файловой системе через символ диска (drive letter). Основные особенности:

* поддержка чтения и записи Ext2 и Ext3 файловых систем
* поддержка чтения Ext4 файловой системы
* отсутствие полной поддержки журналирования файловой системы Ext3
* отсутствие поддержки менеджера логических томов операционной системы Linux
* отсутствие поддержки шифрования

Данный драйвер является наиболее распространённым среди драйверов такого типа, в большинстве случаях работает стабильно, однако в очень редких случаях при записи возможны потери данных (<http://sourceforge.net/projects/ext2fsd/reviews>)

Ext2 Installable File System

Свободно распространяемый драйвер операционной системы Windows для работы с файловыми системами семейства Ext. Предназначен для использования в следующих операционных системах семейства Windows: Windows NT, Windows XP, Windows Vista. Представляет собой драйвер файловой системы Ext2, полностью поддерживающий операции чтения и записи. Работает в режима ядра, то есть на том же программном уровне, что и нативные драйвера операционной системы Windows (NTFS, FASTFAT, CDFS). Драйвер позволяет назначить разделам с файловой системой Ext2 букву диска, что предоставляет возможность всем приложениям взаимодействовать и работать с ними. Поскольку файловая система Ext3 поддерживает обратную совместимость с файловой системой Ext2, данный драйвер также может использоваться для организации работы с Ext3, но без поддержки журналирования. К основными недостатками данного драйвера можно отнести:

* отсутствие поддержки журналирования в файловой системе Ext3
* отсутствие поддержки прав доступа
* отсутствие возможности производить дефрагментацию файловой системы, как и возможности получения информации о текущей фрагментации раздела
* отсутствие поддержки менеджера логических дисков операционной системы Linux

Источник: <http://www.fs-driver.org/>

Paragon ExtFS for Windows

Программное обеспечение компании Paragon, предоставляющий полноценный доступ к файловым системам Ext2, Ext3 и Ext4. Ключевые особенности:

* обеспечивает быстрый и прозрачный доступ к файловым системам семейства Ext\*, полностью поддерживает операции чтения и записи
* предоставляет инструменты для создания и форматирования разделов с файловым системами семейства Ext\*
* поддерживает работу с LVM разделами только в режиме чтения
* поддерживает последнюю версию операционной системы Windows 8.1
* поскольку работает в режиме ядра, обеспечивает хорошую производительность и скорость работы

Источник: <https://www.paragon-software.com/home/extfs-windows-pro/>

DiskInternals Linux Reader

Приложение с графическим пользовательским интерфейсом операционной системы Windows для просмотра содержимого нативных файловых систем операционной системы Linux. Поддерживает чтение таких файловых систем как Ext2, Ext3, RaizerFS, Reizer4, а также HFS+ операционной системы Apple OS. Не предоставляет остальным программам возможность работать с данными файловыми системами через символ диска. Поскольку поддерживается только чтение, гарантируется, что при использовании данной программы файловая система останется в неизменном состоянии и не будет повреждена.

Ext2explore

Приложение операционной системы Windows, по характеристикам аналогичное предыдущему рассмотренному DiskInternals Linux Reader. Из особенностей стоит упомянуть возможность чтения Ext4 файловой системы

Ext2Read

Приложение операционной системы Windows с графическим интерфейсом похожим на интерфейс проводника Windows Explorer. Используется для просмотра и чтения содержимого Ext2, Ext3 и Ext4 файловых систем. Также поддерживает возможность чтения разделов под управлением менеджера логических томов Linux.

Explore2fs

Графическое приложения для просмотра содержимого Ext2 и Ext3 файловых систем. Поддерживает только возможность чтения и копирования файлов.

LTOOLS

Утилита командной строки, предназначенная для чтения и записи Ext2 и Ext3 файловых систем. Также поддерживает возможность работы с файловой системой ReiserFS.

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/>

ZFS-WIN

Свободно распространяемый драйвер операционной системы Windows, предназначенный для доступа к файловой системе ZFS. Позволяет монтировать раздел с файловой системой ZFS только в режиме чтения и осуществлять доступ к нему из других приложений при помощи присвоения разделу буквы диска.

WinBtrfs

Проект на github, представляющий собой драйвер операционной системы Windows, работающий в пользовательском режиме, предназначен для доступа к файловой системе BtrFS. Позволяет приложениям операционной системы Windows работать с файловой системой BtrFS, но только в режиме чтения.

**Использование нативных драйверов**

Нативные драйвера файловой системы лишены многих недостатков, присущих их портированным на другие операционные системы аналогам. Такие драйвера полностью поддерживают все функции и особенности файловой системы, показывают более высокие результаты производительности, поскольку изначально проектируются для использования в конкретной операционной системе. Они более надежны, качественны и активно поддерживаются разработчиками не только файловой, но и операционной системы, в которой эта файловая система является нативной. Используя нативные драйвера файловой системы, практически полностью пропадает необходимость дополнительного их тестирования и отслеживания изменений в реализации файловой системы.

Так, например, в случае с драйверами нативных файловых систем операционной системы Linux, прежде чем включить поддержку файловой системы в основное ядро Linux, драйвер этой файловой системы проходит тщательное тестирование огромным Linux-сообществом, в состав которого входят ведущие компании IT-индустрии.

Однако поскольку драйвер относится к типу программного обеспечения, сильно зависящего от операционной системы, его нельзя взять и просто использовать в другой, ненативной для него, операционной системе. Для реализации этой возможности необходимо в целевой операционной системе (то есть в той, в которой требуется использовать драйвер) создать окружение, эмулирующее окружение нативной для данного драйвера операционной системы.

Так, например, для использования нативных драйверов файловых систем операционной системы Linux в операционной системе Windows, требуется создать некоторое окружение опреционной системы Linux, которое ожидает увидеть драйвер, внутри операционной системы Windows.

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:

* эмулировать программный интерфейс (API) ядра нативной для драйвера операционной системы (в данном случае Linux) в пространстве пользователя целевой операционной системы (Windows)
* использовать виртуальную машину

Очевидно, что сложность реализации первого способа колоссальна, особенно в случае необходимости эмуляции такой операционной системы как Linux. Разработка современной операционной системы представляет собой одну из самых сложных задач в IT-индустри, а потому и портирование операционной системы для запуска в качестве отдельного приложения внутри другой во многом превосходит сложность портирования драйвера файловой системы.

Так, например, эмуляция окружение ядра Linux для работы нативного драйвера файловой системы операционной системы Linux в пользовательском пространстве Windows предполагает реализации как минимум:

* блочного уровня ядра Linux
* фреймворка файловых систем
* примитивов синхронизации
* фреймворка файловых систем в пользовательском пространстве (FUSE)

Также стоит отметить, что вследствие того, что API ядра Linux постоянно меняется, и эти изменения не всегда носят систематический и последовательный характер, необходим прилагать дополнительные усилия на поддержание эмулированного ядра в актуальном состоянии.

Использование виртуальной машины для эмуляции окружения одной операционной системы внутри другой позволит использовать нативную для драйвера операционную систему без изменений. В данном случае полностью исчезает проблема портирования как драйвера, так и нативной для него операционной системы, однако использование виртуальной машины предполагает дополнительные издержки ресурсов системы (главным образом, памяти и процессорного времени).

Несмотря на то, что при реализации первого подхода мы потенциально может получить менее ресурсоемкое и более производительное решение, использование виртуальной машины выглядит наиболее привлекательным с точки зрения потраченных усилий и получаемых выгод.

Именно такой подход был выбран для реализации работы с файловыми системами в операционной системе Windows при использовании нативных драйверов файловых систем операционной системы Linux.

Примеры

Цель

Предоставить приложениям операционной системы Windows возможность работать с файловыми системами операционной системы Linux

**Выбор решения**

Для решения задачи организации доступа к нативным файловым системам операционной системы Linux в операционной системе Windows был выбран подход, заключающийся в использовании нативных драйверов Linux с эмуляцией окружения операционной системы Linux в операционной системе Windows посредством использования виртуальной машины. Кратко перечислим основные преимущества данного подхода по сравнению с остальными (рассмотренными ранее):

* отсутствие затрат на разработку драйвера для каждой из файловой систем, доступ к которым необходимо организовать
* сложность реализации данного подхода никак не зависит от количества файловых систем, возможность работы с которыми нужно предоставить в операционной системе Windows
* нативные драйвера операционной системы Linux реализуют полный доступ к файловой системе, поддерживают все её функции. Обладают высокой производительностью и надёжностью, активно развиваются и поддерживаются разработчиками, лишены ошибок и недоработок, зачастую встречающихся в их портированных аналогах, реализованных сторонними компаниями и отдельными программистами
* драйвера операционной системы Linux, как и сама операционная система Linux, относятся к свободно распространяемому программному обеспечению, следовательно, отсутствуют материальные затраты, связанные с их использованием
* полное переиспользование исходного кода операционной системы Linux, отсутствие затрат на портирование

Одним из главных недостатков данного подхода является существенные потери в производительности, а именно скорости выполнения операций чтения и записи. Также происходит высокое потребление системных ресурсов, связанных с тем, что осуществляется запуск целой виртуальной машины, внутри которой работает полноценная операционная система Linux. Существует несколько способов устранения данных недостатков, о них будет рассказано далее более подробно

Архитектура разрабатываемого решения

Разработать библиотеку, предоставляющую API для монтирования файловой системы и осуществления базовых операций: создание, чтение, запись, удаление, модификация, перемещение файлов и каталогов.

Разрабатываемая библиотека имеет клиент-серверную архитектуру.

Работа разрабатываемой библиотеки заключается в запуске дистрибутива Linux внутри qemu и взаимодейтвию с ним посредством отправки команд на чтение и запись файлов и получение результатов их исполнения.

Библиотека в отдельном процессе запускает виртуальную машину, внутри которой загружается дистрибутив Linux, содержащий нативные драйвера файловых систем и определенный набор вспомогательных утилит для работы с ними. После загрузки операционной системы Linux в ней запускается специальный контролирующий демон, способный выполнять команды по взаимодействию с требуемыми файловыми системами.

При запуске виртуальной машины к ней присоединяется физическое устройство либо образ диска с файловой системой, доступ к которой необходимо организовать. Внутри виртуальной машины присоединенное устройство (либо образ диска) выглядит для операционной системы Linux как некоторое устройство хранения данных (например, как жесткий диск) с нативной для операционной системы Linux файловой системой. Благодаря наличию в операционной системе Linux необходимых драйверов, работающий внутри виртуальной машины демон монтирует это устройство, после чего способен взаимодействовать с монтированной файловой системой в соответствии с командами, полученными от библиотеки, работающей на стороне приложения.

В случае, если вместо реального физического устройства к виртуальной машине присоединяется образ диска, то виртуальная машины транслирует доступ ядра Linux к «физическому» устройство в доступ к образу диску.

Библиотека, выступающая в роли клиента, общается с демоном с помощью механизма удаленного вызова процедур (RPC – Remote Procedure Call) - отправляя ему определенные команды по выполнению операций чтения/записи файлов и получает обратно от демона результаты их выполнения.

Данный механизм позволяет осуществить доступ к любой нативной файловой системе операционной системы Linux, поддержка которой включена в дистрибутив Linux, используемый в виртуальной машине.

Проект libguestfs

Libguestfs – проект с открытым исходным кодом компании RedHat, активно развивается с 2009 года. Предназначен для использования в операционной системе Linux. Представляет собой набор утилит для доступа и модификации образов дисков для виртуальных машин. Предоставляемые инструменты позволяют просматривать и редактировать файлы внутри гостевых систем, выполнять

позволяет создавать образы для виртуальных машин, модифицировать их, ужимать, модифицировать таблицу разделов, управлять конфигурационными файлами, переносить «железные» машины в виртуальную среду, переносить виртуальные машины с одного образа на другой, переносить виртуальные машины из образа на железо. Причем для выполнения этих операций не требуются права суперпользователя (администратора), что является несомненным преимуществом данного подхода.

Предоставляемый набор инструмент позволяет получить практически к любой файловой систем: поддерживаются все нативные файловые системы операционных систем Linux, Windows, Mac OS X, BSD, в том числе менеджер логических разделов Linux, дисковые разделы с таблицей разделов типа MBR и GPT, «сырые» образов дисков, CD и DVD диски, образы дисков ISO, SD-карты и многое другое. Поддерживаются диски используемые виртуальными машинами: qcow2, VirtualBox VDI, VMWare VMDK, Hyper-V VHD/VHDX. Доступ к указанным файловым системам может осуществляться как локально, так и удаленно.

Libguestfs также предоставляет приложениям программный интерфейс для доступа к файловыми системам физических устройств и образов дисков, реализуется в виде библиотеки на языке C, имеет привязки более чем к 10 языкам, таким как Ocaml, Python, Ruby, Java, Haskell и т.д.

Архитектура библиотеки libguestfs

Предоставляемая проектом libguestfs библиотека в своей работе использует архитектуру, описанную в разделе [?].

Работа библиотека начинается с запуска в отдельном процессе виртуальной машины. В качестве виртуальной машины используется qemu.

Внутри виртуальной машины работает ядро операционной системы Linux с полным набором утилит пространства пользователя и небольшой контролирующий демон «guestfsd». Общение библиотеки с демоном осуществляется посредством сетевого взаимодействия через сокеты с помощью удаленного вызова процедур, используя протокол XDR. Обычно вызову каждой функции, предоставляемой библиотекой приложениям, соответствует отправка демону определенной команды на выполнению. В конечном итоге, образ диска присоединяется к виртуальной машине, которая транслирует обращения Linux к физическому устройству в доступ к этому образу диска.

Принцип Модель работы libguestfs

Для н

Для возможности внутри пользовательского приложения операционной системы Windows осуществлять настройку параметров запуска виртуальной машины и впоследствии взаимодействовать с запущенным внутри нее демоном библиотека libguestfs предоставляет специальный обработчик – объект типа “guestfs\_h”, который создается при помощи вызова функции “guestfs\_create”. Данный обработчик характеризует текущее состояние создаваемого библиотекой «устройства», которое представляет собой виртуальную машину с запущенным внутри неё демоном. Данный обработчик используется в качестве параметре при вызове всех функций, предоставляемый библиотекой libguestfs в качестве программного интерфейса (API).

«Устройство» может находиться в одном из трех состояний:

* Конфигурация
* Запуск
* готовность к работе

Конечный автомат состояний «устройства» представлен на рисунке [?].

Нормальная работа предполагает следующую смену состояний:

Программный интерфейс библиотеки предлагает один способ перейти из состояния «конфигурация» через состояние «запуск» в состояние «готов к работе». Вызов функции “guestfs\_launch” блокирует выполнение программы до тех пор, пока дочерний процесс/устройство не перейдет в состояние «готовность к работе», в котором устройство готово принимать команды на исполнение. «guestfs\_launch» в процессе своего вызова изменяет состояние из «конфигурация» в «запуск» пока происходит запуск вирутальной машины и инициализация операционной системы и демона.

Вызов многих функций возможен только после того, как устройство находиться в состоянии «готовность к работе»

Процесс загрузки устройства

При правильной работе библиотеки

Для моделирования текущего состояния устройства используется конечный автомат

Для настройки параметров виртуальной машины, её запуска и последующей работы с демоном используется специальный обработчик (handle) - объект типа “guestfs\_h”, который создается с помощью функции “guesfs\_create”

Для моделирования состояния дочернего процесса, внутри которого запущена виртуальная машина с операционной системой Linux, Libguestfs использует следующий конечный автомат состояний:

« схема конечного автомата libguestfs »

Библиотека может находиться в одном из трех возможных состояний:

Решение портировать библиотеку

Поскольку архитектура и принцип работы предоставляемой libguestfs библиотеки идеально подходит для выбранного решения проблемы доступа к нативный файловым системам операционной системы Linux внутри операционной системы Windows было решено использовать исходный код этой библиотеки в разрабатываемом решении. Поскольку сам а библиотека написана для работы в операционной системе Linux, было принято решение портировать исходный библиотеки для работы в операционной системе Windows.

Главные задачи портирования были определены как:

Возможность монтирования файловой системы

Возможность выполнения операций чтения

Возможность выполнения операций записи

К главным задачам портирования были отнесены возможность монтирования файловой системы, выполнения операций чтения и записи.

Cygwin

В самом проекте Libguestfs рассматривалась возможность запуска библиотеки в операционной системе Windows.

В TODO листе проекта libguestfs в качестве одной из задач указано необходимость портирования библиотеки на Windows.

На внутреннем форуме компании обсуждалась возможность портирования, в которой советовалось вначале портировать библиотеку для запуска в Cygwin-е, чтобы оценить сложность и возможность портирования этой библиотеки без необходимости полного её переписывания.

Cygwin представляет собой большрй набор GNU инструментов, позволяющий портировать многие Unix-приложения на Windows с минимальными изменениями в их реализации. Ключевой частью Cygwin-а является его библиотека, которая реализует существенную часть POSIX API (системные вызовы и окружение), которую и ожидают увидеть программы, написанные для Unix-систем.

Портирование на Cygwin

Портирование на Cygwin не потребовало внесения существенных изменений в исходный код библиотеки. Практически все Linux-зависимые функции, используемые в реализации libguestfs, поддерживаются в Cygwin.

Большая часть из всех изменений была внесена в процесс сборки проекта. В частности, была отключена генерация кода, не использующаяся в билиотеке, а также

Поскольку во время сборки проекта происходит помимо самой C-библиотеки происходит сборка многих

Внесены изменения в процесс сборки проекта: отключено сборка многих вспомогательных утилит.

Внесены изменения в процесс запуск виртуальной машины:

Создана кросс-платформенная обертка над сетевым соединением.

Qemu использовали

Результат работы

/\* графики, на которых Cygwin намного хуже по скорости чтения файлов: мелких и больших

По скорости записи файлов: мелких и больших

По потреблению оперативной памяти: потребление главным образом за счет запуска qemu \*/

Использование Cygwin-прослойки предполагает дополнительные издержки

Тестирование работы библиотеки libguestfs, собранной для запуска в Cygwin,

Производительность работы полученной библиотеки оценивалась следующими показателями:

время запуска виртуальной машины,

Как видно из графиков, представленных на рисунке [?],

Как видно из графиков, представленных на рисунке [?], производительность работы библиотеки снизилась. Во-первых, увеличилось время запуска виртуальной машины, время чтения, время записи файлов. По сути, возрасло время выполнеиия каждой из команд. Это можно объяснить следующим. Во-превых, использование Cygwin предполагает дополнительные издержки. Во-вторых, сюда можно отнести особенности работы операционных систем.

Основное потребление оперативной памяти связано с запуском виртуальной машины. В данном случае, виртуальная машина

В Linux используется KVM, в Windows этого нет.

В зависимости от количества предоставляемой qemu оперативной памяти он использует для своих нужд прядка 40 -50 МБ при запуске виртуальной машины с 128 МБ

Чтение файлов осуществляется примерно за то же время, связно с тем

Полученные показатели потребления оперативной памяти системы играют вторичную роль, поскольку зависят от параметров запуска виртуальной машины: а именно, сколько физической памяти будет предоставлено гостевой операционной системе внутри виртуальной машины.

Полученный результат носит экспериментальный характер, служит доказательством возможности портировать часть библиотеки libguestfs, отвечающей за доступ к файловым системам, для работы в Windows.

М 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.

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. 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