Добрый день, тема моей работы звучит следующим образом: «Работа с файловыми системами в операционной системе Windows с использованием драйверов операционной системы Linux»

Файловая система представляет собой способ организации, хранения и именования данных на носителях информация. Она определяет, где и как данные сохраняются на физическом устройстве и по каким правилам извлекаются обратно для работы в приложениях.

Файловая система является ключевым компонентом любой операционной системы. Для каждой ОС сущ-ет определенный набор ФС, поддержка которых изначально включена в ядро ОС. Такие файловые системы мы будем называть нативными. В зависимости от ОС количество и виды таких ФС сильно варьируется. Так, в случае ОС Linux нам предоставляется возможность использовать широкий спектр разнообразных ФС. ОС Windows таким качеством похвастаться не может.

При это стоит упомянуть, что сущ-ет целые классы программного обеспечения, к-ым в сиду своей специфики нацелены на работу с широким количеством ФС. К ним можно отнести системы резервного копирования и восстановления данных, антивирусное ПО, Менеджеры жестких дисков и разделов, файловые обозреватели и т.д. любое ПО, которому требуется работать с ненативной для ОС файловой системой. В случае, если операционная система не поддерживает файловую систему, проблему организации доступа должна решать сама программа и чем с большим количеством файловых систем она способна работать, тем она более привлекательна для пользователя.

ЦЕЛЬ: Предоставить приложениям операционной системы Windows возможность работать с файловыми системами, нативными для операционной системы Linux и поддержка которых изначально отсутствует в Windows.

Для работы с файловой системой операционной системе необходимо предоставить определенный программный компонент – драйвер файловой системы. Драйвер – программный модуль, который умеет интерпретировать внутренние структуры файловой системы и предоставляет приложениям удобный иерархический вид её содержимого.

*Сложность файловых систем*

Откуда взять драйвер файловой системы Linux для Windows? Есть 2 основных подхода.

Мы можем посмотреть на исходные драйвера Linux и портировать их, написать свою реализацию для работы Windows. Стоит сказать, что данный подход очень трудоемкий, дорогой и долгий.

Тогда мы можем взять уже реализованные кем-то Windows-аналоги этих драйверов. К сожалению, на сегодняшний день таких драйверов мало, поддерживают ограниченное кол-во файловых систем ОС Linux. Зачастую такие драйвера носят чисто экспериментальный характер, поддерживают лишь ограниченный функционал ФС (обычно только чтение).

Второй же подход заключается в том, что мы берем лрайвера, написанные для Linux, и используем их. Такие драйвера реализует полный доступ к ФС, обладают высокой надежностью и эффективностью, потому как разрабатываются и тестируются огромным Linux-сообществом. НО, для их использования мы должны реализовать в Windows окружение Linux. Сделаем это с помощью виртуальной машины.

Таким образом, архитектура выбранного решения выглядит следующим образом. Реализуется библиотека, предоставляющая Windows-приложению программный интерфейс для доступа к ФС. В своей работе библиотека использует клиент-серверное взаимодействие. В отдельном процессе запускается виртуальная машина, внутри которой загружается урезанный дистрибутив Linux с необходимыми драйверами и утилитами для работы с ФС. При запуске виртуальной машины к ней присоединяется образ диска с требуемой ФС, который внутри виртуальной машины выглядит как реальное устройство хранения данных ФС, с нативной для Linux, доступ которой нужно организовать. Внутри Linux работает специальный контролирующий демоном, который монтирует файловую систему, после чего способен выполнять команды по чтению/записи ФС и возвращает результаты их исполнения. Общение с демоном осуществляется посредством сетевого взаимодействия.

Стоит сказать, то очень вовремя был обнаружен крайне полезный в рамках данной работы проект компании RedHat – проект libguestfs.

Данный проект представляет огромный набор утилит для доступа и модификации образов дисков виртуальных машин. Предназначен для работы в Linux и на первый взгляд напоминает что-то вроде этого:%

Предоставляемые инструменты позволяют делать очень много вещей связанных с редактированием образом, просмоор содержимого, создания, модификации, переноса виртуальных машин на железо и обратно и т.д.

Но нас больше интересует библиотека, реализуемая в рамках данного проекта.

Она инкапсулирует в себе базовую функциональность, используемую во всех генерируемых в рамках данного проекта утилитах и инструментах. предоставляет приложениям программный интерфейс для доступа и работы с практически с любыми файловыми системами и образами дисков. Библиотека реализована на языке C.

Причем в своей работе библиотека использует ту архитектуру, о которой я только что говорил. В качестве виртуальной машины предоставляется возможность использовать qemu, qemu-kvm, uml. Для взаимодействия используется удаленным вызов процедур с помощью протокола XDR

Итак, задачи данной работы были сформулированы следующим образом:

* Выбрать виртуальную машину для запуска Linux в Windows
* Портировать исходный код библиотеки libguestfs на Windows
* Реализовать передачу файлов через разделяемую память
* Сравнить производительность с исходной (с той, которая на Linux)

Портирование библиотеки я начал с реализации возможности её запуска в Cygwin. Cygwin – проект, предоставляющий Unix-среду в Windows. Ключевой частью Cygwin является его динамически подключаемая библиотека (DLL) реализует значительную часть стандарта POSIX на основе системных вызовов Win32.

В данном случае портирование потребовало минимального количества изменений, сделанные в способе запуска виртуальной машины и организации сетевого взаимодействия. Больше сил ушло изучение системы сборки, на разрешение сторонних зависимостей и собственной сборки библиотеки.

Результаты тестирования представлены на слайде. Очевидно, что в результате данного портирования библиотека показывает менее привлекательные результаты, но главной целью данного портирования была необходимость оценить сложность и в принципе возможность портирования данной бибилотеки без её полного переписывания.

После этого я приступил к нативному портированию библиотеки. Здесь мне уже потребовалось реализовать Windows-аналоги некоторых Linux-функций. Платформозависимый код был релизован внутри кроссплатформенных интерфейсов, которые я реализовал в виде небольшой библиотеки. Был изменен режим запуска виртуальной машины. Переписаны отдельные модули бибилотеки libguestfs.

В ходе портирования была произведена попытка заменить XDR на Protobuf. Это было сделано, но, к сожалению, оказалось не очень полезным.

Результаты тестирования данного портирования представлены на слайде.

Поскольку скорость чтения (и записи) файлов оставляет желать лучшего, было принято решение организовать передачу файлов через разделяемую память. Для этого требовалось предоставить демону доступ к этой памяти. Это было сделано с помощью представления разделяемой памяти внутри виртуальной машины в виде PCI-устройства. В дистрибутив Linux был включен специальный драйвер для работы с этим устройством.

Полученные результаты очень вдохновляют. Операции чтения и записи выполняются в разы быстрее, чем раньше, что не может не радовать.

Таким образом, в рамках данной работы была реализованы библиотека, позволяющая Windows-приложениям работать с нативными для Linux файловыми системами. Для доступа к ФС используются нативные Linux-драйвера, эмуляция окружение Linux осуществляется с помощью виртуальной машины. Реализована возможность передачи файлов посредством механизма разделяемой памяти для повышения скорости выполнения операций чтения и записи.

Дальнейшая работа

Портирование и использование UML в качестве виртуальной машины

Портирование всего проекта libguestfs.

*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*. *Файловые система использует определенный набор правил и структур данных для контроля за тем, где и как файлы сохраняются на физическом устройстве и извлекаются обратно для использования в работе приложений.*

Каждая файловая система изначально поддерживает работу с определенным набором файловых систем. Такие файловые системы называются нативными для данной операционной системы. в зависимости от рассматриваемой операционной системы Количество нативно поддерживаемых файловых систем сильно варьируется. 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.

Существуют целые классы программного обеспечения, которым в силу своей специфики требуется работать с широким количеством файловых систем. К ним относятся:

* Системы защиты данных, антивирусное программное обеспечение
* 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. -> драйвер до цели

..

Текуще положение дел -> драйвер до цели

Цель:

Для работы с файловой системой операционной системе необходимо предоставить определенный программный компонент – драйвер файловой системы. Драйвер – программный модуль, который умеет интерпретировать внутренние структуры файловой системы и предоставляет приложениям удобный иерархический вид её содержимого.

Откуда взять драйвер файловой системы Linux для Windows?

Портировать драйвер файловой системы Linux: взять и написать свой

Использовать исходные нативные драйвера файловой системы, создав в Windows специальное окружение операционной системы Linux

|  |  |
| --- | --- |
| Исходнык драйвера Linux  Качетсвенные , надежные  Разрабатываются ведущими IT-компаниями вместе с файловыми системами  Тестриуемые (прежду чем попасть в основное ядро Linux – долгий этап тестирования и исправления багов)  Поддержка со стороны разработчиков  Открытый доступ | Экспериментальные  Отсутсвие поддержки  Неполная поддержка всех возможной файловой системы, недоработанные  Ненадежные, используем на свой страх и риск  Тяжелая разработка  Для каждой файловой системы – отдельная реализация драйвера |

Портироватьк каждый драйвер – не подходит. Создадим окружение Linux и будем использовать нативные драйвера.

Архитектура решения

Разраьа

Модель работы разрабатываемой библиотеки представляет собой клиент-серверное взаимодействие, в котором библиотека, предоставляющая приложению определенный программный интерфейс, выступает в роли клиента, а запускаемая ею виртуальная машина

Разрабатываемая библиотека имеет клиент-серверную архитектуру.

Библиотека, работающая на стороне приложения, в отдельном (дочернем) процессе запускает виртуальную машину, внутри которой загружается дистрибутив Linux. В состав загружаемого дистрибутива входят 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, полученные от библиотеки, по взаимодействию с требуемыми файловыми системами. Таким образом, библиотека, предоставляющая приложению определенный программный интерфейс по работе с файловой системой, выступает в роли клиента, а запущенная ею виртуальная машина с загруженной внутри операционной системой Linux и демоном выступает в роли сервера.

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 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 и осуществлять требуемые операции чтения и записи в соответствии с командами, полученными от библиотеки, работающей на стороне приложения.

Стоит заметить, что в случае, если вместо реального физического устройства к виртуальной машине присоединяется образ диска, то виртуальная машины сама транслирует доступ ядра Linux к «физическому» устройству в доступ к образу диску.

Библиотека, выступающая в роли клиента, «общается» с демоном с помощью механизма удаленного вызова процедур (RPC – Remote Procedure Call). Библиотека отправляет демону определенные команды для осуществления операций чтения и записи файловой системы и получает обратно от демона результаты их выполнения.

Рассмотренный механизм работы позволяет осуществить доступ к любой нативной файловой системе операционной системы Linux, расположенной на реальном физическом устройстве либо образе диска, поддержка которой включена в используемый в виртуальной машине дистрибутив Linux.

**Проект libguestfs**

Библиотека libguestfs.

Задачи.

Выбор виртуальной машины.

Портирование на Cygwin. Устранение зависимостей. Кроссплатформенное сетевое взаимодействие. Результаты – proof of concept, конечно медленее.

Портирование на Windows. Protobuf

Портирование на MinGW. Реализация кроссплатформенных интерфейсов.

Реализация разделяемой памяти. Qemu, ivshmem. Proto.c , shared\_memory

Собственная система сборки с Autotools.

Дальнейшая работа: UML

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

Реализация выбранного подхода будет выполнена в виде динамической библиотеки операционной системы Windows, написанной на языке программирования C. Выбор языка C в качестве языка разработки является наиболее целесообразным, поскольку:

Архитектура решения

Реализуем библиотеку, которая предоставляет приложениям определенный программный интерфейс по доступу к файловым системам, расположенным на физическом устройстве или образе диска.

Работа библиотеки представляет основывается на клиент-серверном взаимодействии. Библиотека выступает в роли клиента, виртуальная машина – в роли сервера.

Библиотека в отдельном процессе запускает виртуальную машину. Внутри виртуальный машины запускается урезанный дистрибутив Linux, с необходимы набором драйверов и утилит по работе с файловыми системами.

Таким образом, задачами данного проекта являются

Выбор архитектуры

Выбор виртуальной машины

Портирование Libguestfs на Windows

Портирование системы сборки

Кросс-платформенный протокол взаимодействия клиента и сервера

Локализация кода, зависимого от типа операционной системы внутри реализаций кросс-платформенных интерфейсов

Выбрать подходящую виртуальную машину для запуска Linux

Сравнить производительность с Linux

Выбор QEmu

В качестве виртуальной машины было решено использовать QEMU, поскольку:

Виртуальна машина с открытым исходным кодом

Эмулирует большое количество платформ

Работает в пространстве пользователя

Имеется возможность использовать в Windows. Есть дистрибутивы для запуска в Windows

Портирование на Cygwin

Много сторонних зависимостей. Минимальное количество изменений, proof of concept, изменения в запуске виртуальной машины, изменения в сетевом взаимодействии.

Результаты

Тестировалась производительность библиотеки. Сравнивалось врем выполнения операций чтения и записи. Cygwin очевидно работает медленней, зато работает.

Нативное портирование

Реализация кроссплатформенных интерфейсов: conn-socket, pipe, create process – fork

Реализация Windows-аналогов некоторых функций на основе предоставленной в сети интернет документации

Замена XDR на Protobuf.

Реализации конвертеров из Protobuf в XDR

Результаты

Linux XDR Protobuf. Быстрее, но все равно медленнее Linux. Быстрее, чем Cygwin

Портирование с MinGW  
Интеграция в систему сборки Autotools.

Реализация общей памяти

Как уже упоминалось ранее, всё взаимодействие с демоном осуществляется посредством сетевого взаимодействия. Все данные, передаваемые демону и получаемые обратно от него, передаются через сокеты. В частности, команды и результаты их выполнения также передаются через сокеты.

Например, в случае выполнения операции чтения содержимое файлов пересылается частями, каждая часть размеров максимум в 8 КБ. Учитывая, что пересылка каждой из частей сопровождается дополнительными издержками, связанными с организацией сетевого взаимодействия, такими как копирование данных для пересылки, упаковка данных в пакет, передача пакета, извлечение данных и т.д., время чтения файла размеров несколько мегабайт и выше существенно возрастает, скорость сильно замедляется и далеко от нативной. То же самое касается и операции записи.

Собственно, для выполнения операции операций чтения и записи было принято решение организовать передачу файлов с помощью разделяемой памяти.

Использование разделяемой памяти позволяет:

Избавиться от лишнего копирования данных, происходящий при их передаче через сокеты.

Собственно, передавать файл частями гораздо большего размера, порядка несколько мегабайт и даже гигабайт, в зависимости от доступной оперативной памяти.

Доступ к разделяемые файлы можно получить через его имя.

Реализация доступа к разделяемой памяти через ivshmem

Внутри qemu разделяемая память выглядит как память специального pci-устройства. Реализуем драйвер для этого устройства. Используем драйвер для работы с устройством. Существует реализация для Linux (nahanni project). Модуль ivshmem. Портирован для использования в Windows как ivshmem-windows.c

Тестирование

Очевидно, что быстрее теперь. Но скорость далека от нативной

Результаты

Библиотека, реализованная на языке C, позволяющая осуществлять доступ к нативным файловым системам.

Возможность передачи файлов посредством разделяемой памяти

Доступ к любой нативной файловой системе операционной системы Linux

Низкая скорость чтения файлов

Высокие издержки памяти связанные с запуском виртуальной машины

Дальнейшая работы

Использование UML в качестве виртуальной машины

Что такое Cygwin? Cygwin – проект компании RedHat, представляет собой Unix-подобную среду.

Кроме того, Cygwin включает в себя инструменты разработки GNU для выполнения основных задач программирования, а также и некоторые прикладные программы, эквивалентные базовым программам UNIX

Таким образом, Cygwin представляет собой удобный инструмент для портирования программного обеспечения Unix в Windows.

Прежде чем приступить к реализации данного решения был обнаружен очень интресный проект

Хочется сказать, что оченб вовремя был обраружен