АНО «Институт логики, когнитологии и развития личности»
ALT Linux

Двенадцатая конференция разработчиков свободных программ

Калуга, 16–18 октября 2015 года

Тезисы докладов

Москва,
Альт Линукс,
2015

УДК 004.91
ББК 32.97

Двенадцатая конференция разработчиков свободных программ: Тезисы докладов / Калуга, 16–18 октября 2015 года. М.: Альт Линукс, 2015. — 60 с. : ил.

В книге собраны тезисы докладов, одобренных Программным комитетом двенадцатой конференции разработчиков свободных программ.

ISBN 978-5-905167-19-5

Программа конференции

16 октября

15.00- Заезд

17 октября

9.30 – 9.45 Регистрация участников

Утреннее заседание
9.45–13.30

9.45 – 10.00 Алексей Новодворский (BaseALT). Вступительное слово

10.00–10.30 Николай Пакулин, Алексей Хорошилов (ИСП РАН)

10.30–11.15 Александр Боковой

11.15–11.45 Дмитрий Костюк

11.45–12.00 Кофе-пауза

12.00–12.30 Иван Панченко, Дмитрий Васильев (Postgres Professional)

12.30–13.00 Роман Симаков (ООО «Ред Софт»)

Дневное заседание
15.00–18.15

18 октября

Утреннее заседание
10.00–12.00

Николай Пакулин, Алексей Хорошилов
Москва, ИСП РАН

Проект: ChPOK `https://forge.ispras.ru/projects/chpok`

Свободная реализация ARINC-653-совместимой операционной системы реального времени

ARINC-653 является стандартом на операционные системы реального времени (ОСРВ), применяемые в авионике и других ответственных областях. Стандарт определяет базовые принципы работы ОСРВ, а также регламентирует прикладной программный интерфейс, предоставляемый пользовательским приложениям. Основными особенностями ARINC-653 являются статическое распределение времени и памяти между пользовательскими разделами (аналог POSIX процессов), выделение всех ресурсов на этапе инициализации системы/раздела, запрет на использование разделяемой памяти с ARINC-каналами как единственным механизмом межпроцессного взаимодействия и специфический механизм обработки ошибок (Health Monitoring).

При наличии большого количества свободных реализаций ОСРВ только одна из них [1] декларирует поддержку ARINC-653. Более того, известно, что было сделано несколько попыток реализовать ARINC-653 поверх популярных свободных ОСРВ, но об успехом эти попытки не увенчались. Единственная свободная реализация ARINC-653 (лицензия BSD) была разработана во французском университете Telecom ParisTech в рамках диссертации Джульена Деланжа и вскоре после его защиты практически перестала развиваться.

В ИСП РАН в рамках работ по верификации коммерческих ARINC-653-совместимых ОСРВ был разработан тестовый набор, предназначенный для тестирования ОСРВ на соответствие стандарту ARINC-653 часть 1 и часть 2. В результате применения этого набора к свободной реализации POK было выявлено, что POK не соответствует многим требованиям стандарта, хотя в целом система спроектирована достаточно аккуратно. В ходе инициативного проекта ИСП РАН эта ОСРВ была доработана до полного соответствия требованиям стандарта ARINC-653 часть 1 при работе на платформе x86 и PowerPC

в эмуляторе Qemu. Это позволило применять OCPB для предварительной отладки ARINC-653-совместимого бортового программного обеспечения. Все доработки ИСП РАН были опубликованы под лицензией GPLv3.

Также в ИСП РАН была выполнена интеграция POK с системой автоматизации проектирования интегрированной модульной авионики MASIW [2], в рамках которой связка Qemu+OCPB применялась в качестве основы для виртуальной интеграции комплекса бортового оборудования.

В настоящее время ведётся портирование OCPB на аппаратный модуль с процессором PowerPC QorIQ P3041 с поддержкой сетевого интерфейса, а также разработка библиотеки POSIX интерфейсов.

Литература

[1] *Julien Delange, Laurent Lec* «POK, an ARINC653-compliant runtime released under the BSD licence», Proceedings of 13th Real-Time Linux Workshop, 20-22 October, 2011.

[2] *Д. В. Буздалов, С. В. Зеленов, Е. В. Корныхин, А. К. Петренко, А. В. Страх, А. А. Угненко, А. В. Хорошилов.* «Инструментальные средства проектирования систем интегрированной модульной авионики». Труды Института Системного Программирования РАН, Том 26-1, 2014, с. 201-230.

Александр Боковой
Эспоо, Финляндия, Red Hat Ltd.

Проект: Samba Team `https://samba.org`, `https://www.fedoraproject.org/`

Планы развития Samba: upstream и downstream (Fedora)

Аннотация

В докладе рассматриваются текущие планы по разработке свободной реализации протокола SMB и решения для замены Active Directory как с точки зрения разработчиков проекта Samba, так и с точки зрения интеграции выпускаемых версий в дистрибутив GNU/Linux на примере Fedora Project.

Дмитрий Костюк, Олег Латий, Анастасия Маркина
Брест, Брестский государственный технический университет

Инструментальная оценка состояния пользователя в задаче сравнения интерфейсов офисных приложений

Аннотация

Представлен программно-аппаратный проект для автоматизированной оценки состояния пользователя, работающего с графическими приложениями. Аппаратная часть включает бытовой энцефалограф для измерения концентрации внимания, а также авторский проект на базе Arduino. С помощью данной разработки выполнен практический анализ эффективности взаимодействия оператора с инструментальными панелями современных офисных пакетов. Тестирование построено на сериях заданий, связанных с функциями форматирования, выполняемыми последовательно в сравниваемых приложениях. Сравнение выполнено для офисных интерфейсов трех категорий: классической верхней панели, секционированной боковой панели и Ribbon (Microsoft Fluent Interface). Оценка данных тестирования включает скорость работы пользователя, его умственную и физическую нагрузку, а также подверженность стрессовым ситуациям.

Введение

Появление «ленточных» инструментальных панелей (Ribbon, Microsoft Fluent Interface или MFI) можно назвать самой кардинальной GUI-инновацией Microsoft со времен панели задач — и при этом самой противоречивой. Отнюдь не новый способ переключения содержимого панели с помощью вкладок был сильно модифицирован добавлением следующих особенностей:

- отсутствие системного меню, делающее ленточную панель единственным органом управления;
- деление видимой части панели на секции, каждая с виджетами самого разного размера;
- автоматическое переключение вкладок панели в зависимости от прогноза о намерениях пользователя (в ряде случаев — миграция виджетов между видимой и скрытой частями секции).

В ряде источников отмечалось, что хотя MFI нередко обеспечивает доступность нужных элементов в один клик, совмещение в одной полосе множества элементов чередующихся размеров рождает у пользователя ощущение хаоса. Хотя в продуктах Microsoft и других популярных приложениях под Windows этот подход признаётся пользователями как минимум приемлемым (иногда — после первоначального шока), тем не менее он не набрал среди сторонних разработчиков достаточно популярности, чтобы проявиться на других платформах. В итоге, когда в 2011–2014 гг. возник ажиотаж вокруг экспериментальных интерфейсов [1], разработчики свободных офисных пакетов экспериментировали не с MFI, а с другими альтернативными элементами (например, боковой секционированной панелью).

Однако за 8 лет сформировалась категория пользователей, натренированных на применение MFI, и если окажется, что такие привычки сильно влияют на пользовательское восприятие, в ближайшее время альтернативным платформам придется существовать в мире девиантных пришельцев из Microsoft-ориентированного окружения. Поэтому нам представляет интересным выяснение того, насколько комфортны для современных пользователей традиционные инструментальные панели, и насколько различается эффективность взаимодействия с более традиционными интерфейсами и с MFI.

Выбор приложений и тестовых заданий

Для исследования этого вопроса нами были выбраны 3 варианта инструментальных панелей:

- классическая панель вверху экрана — представлена LibreOffice;
- боковая панель справа от страницы с документом, содержащая раскрываемые секции (сначала встречалась в офисных приложениях для MacOS, затем появилась в IBM Symphony Office, а после его передачи фонду Apache стала опцией в OpenOffice и LibreOffice; кроме того, используется в Calligra Suite) — представлена LibreOffice с отключенной верхней панелью;
- инструментальная панель в стиле MFI — представлена MS Office 2007.

При выборе приложений для тестирования была предпринята попытка уравнять визуальную привычность интерфейсов с учётом того, что большинство подопытных повседневно использует офис Microsoft

и реже — LibreOffice. Этим объясняется выбор Microsoft Office более старой (и потому не самой привычной) версии, а также выбор боковой панели LibreOffce вместо Calligra, незнакомой тестируемым.

В тестировании приняли участие 23 студента 20–22 лет. Очевидно, что размер выборки позволяет давать результатам лишь оценочную характеристику; однако он типичен для исследований подобного рода [3], вероятно, в силу организационных причин.

Тестируемые выполняли задания в два этапа: сначала в трёх приложениях для ознакомления с интерфейсом, а затем другие варианты тех же заданий в режиме мониторинга. Каждый вариант состоял из 16 операций, связанных с форматированием и разметкой документа, выбранных с учётом предположения о влиянии организации инструментальных панелей преимущественно на действия, выполняемые без клавиатуры.

Методика и результаты тестирования

В соответствии с опробованной ранее методикой [1] рассматривались быстрота выполнения заданий, физическая и умственная нагрузка. Для оценки умственной активности использованы значения концентрации внимания — параметр «Attention», связанный с β-ритмом, измеряемый бытовым энцефалографом Neurosky Mindwave. Физическая нагрузка оценивалась по частоте сердечных сокращений (ЧСС), которая измерялась совместно с электропроводностью кожи (ЭПК) аппаратным модулем собственной разработки[1], позволяющим также оценивать изменения артериального давления [2]. Парные измерения ЭПК и ЧСС позволяют регистрировать эмоциональные переживания пользователя и косвенно отслеживать уровень стресса [3].

По скорости выполнения заданий боковая панель оказалась аутсайдером, что хорошо согласуется с затруднениями большинства пользователей при прецизионных горизонтальных движениях мышью [1]. 58% тестируемых достигли наибольшей скорости работы в MFI (далее эта группа условно именуется «MFI-чемпионами»), для 32% более быстрая работа отмечена при использовании верхней панели, и только 10% проявили наибольшую эффективность в интерфейсе с боковой панелью.

[1] `http://github.com/fiowro/uxdump`

При нормировании скорости работы относительно интерфейса с верхней панелью, MFI оказался более быстрым интерфейсом для 63% пользователей (средний выигрыш времени — 30%). Соответственно только для 37% пользователей верхняя панель оптимальнее, чем MFI (причем средний выигрыш времени в их случае — всё те же 30%). Наконец, боковая панель предпочтительнее верхней для 38% пользователей (средний выигрыш времени 33%), и почти 2/3 этих пользователей — MFI-чемпионы.

Половина MFI-чемпионов показала для MFI более высокую частоту пульса, чем для верхней панели, и 2/3 из них показали меньшую концентрацию внимания при работе с MFI. Интересно, что среди «чемпионов верхней панели» её использование уменьшало концентрацию внимания только в половине случаев.

У ряда пользователей наблюдались выраженные всплески реакции ЭПК (кожно-гальванический рефлекс или КГР), соответствующие эмоциональной реакции на события. Исходя из предположения о сосредоточенности на решаемой задаче, КГР говорит о переживаниях или от тестовых заданий (различия вариантов не позволяли выполнять работу бездумно), или от поиска нужного виджета на панели. Ярко выраженный КГР зарегистрирован в 1/4 от общего числа тестов: всего 1 раз при использовании боковой панели, чаще всего при использовании MFI, в т. ч. нередко у MFI-чемпионов. Аналогичное наблюдение верно и для чемпионов верхней панели: иначе говоря, использование наиболее привычного и эффективного интерфейса не только не спасает пользователя от ярких (и едва ли положительных) эмоций — скорее, оно может косвенно им способствовать. Вопрос, насколько характерен аномально-низкий уровень переживаний при взаимодействии с боковой панелью, очевидно требует исследования на большей выборке.

Как можно заметить, несмотря на доминирование офисных MFI-интерфейсов в исследованной группе, треть пользователей демонстрирует наибольшую скорость работы в офисе с традиционным расположением инструментальной панели, даже не имея привычки его повседневного использования. Кроме того очевидна неплохая адаптируемость пользователей к интерфейсу MFI, несмотря его неоднократно озвученные недостатки.

Литература

[1] *Костюк Д., Шитиков А.* Оценка эффективности мультипрограммной работы оператора в современном графическом интерфейсе // Десятая конференция разработчиков свободных программ: Тезисы докладов. – Калуга, 20–22 сентября 2013 года. М.: Альт Линукс, 2013. – С. 51–56.

[2] *Костюк Д. А., Латий О. О.* Модуль инструментальной оценки состояния пользователя // Открытые технологии: сб-к материалов одиннадцатой международной конференции разработчиков и пользователей свободного программного обеспечения Linux Vacation / Eastern Europe 2015, Гродно, 25–28 июня 2015 г. – Брест, Альтернатива, 2015. – С. 91–95.

[3] *Drachen A., Nacke L.E., Yannakakis G., Lee Pedersen A.* Correlation between heart rate, electrodermal activity and player experience in First-Person Shooter games. // Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games, Los Angeles, CA, –2010. – PP. 49–54.

Панченко Иван Евгеньевич, Васильев Дмитрий Викторович
Москва, Postgres Professional

Проект: СУБД PostgreSQL

Диагональное масштабирование PostgreSQL

Аннотация

В докладе рассказывается о результатах тестирования производительности PostgreSQL на современных Hi-End серверах компаний HP (SuperDome X) и IBM (POWER8). Основное внимание было уделено блокировкам для доступа к разделяемым данным и связанными с этим узкими местами. Целью тестирования было проверить пределы линейного read scalability при увеличении количества ядер выделяемых для PostgreSQL. Тестирование проводилось для различных версий БД (9.4, 9.5б 9.6), чтобы проверить нововведения, призванные повысить производительность на многопроцессорных архитектурах. Дополнительно сравним версии СУБД PostgreSQL, собранной различными компиляторами: gcc, advanced toolchain ibm, xlc и clang в условиях HT/SMT.

Симаков Роман Александрович
Москва, ООО «Ред Софт»

Проект: СУБД «Ред База Данных»
`http://red-soft.biz/ru/main_products.html#rbd`

«Ред База Данных» — СУБД для органов государственной власти

Аннотация

В докладе будет дан обзор сертифицированной промышленной российской системы управления базами данных с открытым кодом «Ред База Данных», истории ее создания и планов развития. Мы рассмотрим функциональные возможности и особенности данной системы, вопросы обеспечения надежности и производительности, особенности технической поддержки. На основе опыта промышленной эксплуатации, включая федеральные службы, будут рассмотрены тонкости взаимодействия СУБД с операционной системой в части интеграции с системами безопасности и службами аутентификации.

Введение

Компания «Ред Софт», созданная в 2006 году, является российским поставщиком решений в области управления информацией, основанных на программном обеспечении с открытым исходным кодом.

Ключевым продуктом компании, на котором строятся все решения, является СУБД Ред База Данных, основанная на широко известном проекте Firebird. Работа над проектом ведется при тесном взаимодействии с сообществом пользователей и разработчиков Firebird. Компанией «Ред Софт» Firebird был доработан по нескольким направлениям, каждый их которых востребован при реализации крупных проектов как в государственных, так и в коммерческих организациях. Ряд таких доработок интегрирует СУБД Ред База Данных с ОС и в доклад раскрывает эти и другие особенности.

Функциональные доработки

Логическая синхронная и асинхронная репликация на уровне ядра

СУБД «Ред База Данных» поддерживает реплицирование всех изменений в режиме на один или несколько удаленных серверов. В

случае сбоя (в том числе аппаратного) любой из них готов к использованию в качестве основного сервера СУБД. Это позволило построить отказоустойчивые системы, функционирующие в режиме 24/7, а также разгрузить мастер, направив читающие запросы на резервные сервера. Задача организации отказоустойчивого кластера требует комплексного подхода и решается с помощью специализированных продуктов вроде pacemaker. Чтобы выполнить такую интеграцию, мы разработали агент для СУБД Ред База Данных, которые способен автоматически поддерживать функционирование на доступных серверах в прозрачном для пользователя режиме.

Аутентификация пользователей с помощью LDAP и Active Directory

Часто информационные системы крупных предприятий или ведомств государственных служб управляют пользователями с помощью таких инструментов как LDAP каталоги. Аутентификация пользователей всех ИТ-сервисов производиться в одном месте однократно за сеанс работы. СУБД при этом не исключение и Ред База Данных реализует такую интеграцию.

Консистентное состояние БД на диске в любой момент времени

Это отличительная и уникальная особенность СУБД Ред База Данных, унаследованная от ядра Firebird. Благодаря технологии Careful Write страницы пишутся на диск в строгом порядке в соответствии с графом зависимостей. Благодаря данной технологии абсолютно исключается необходимость в процедуре восстановления после сбоев, в том числе аппаратных. С точки зрения ОС и виртуализации это означает, что если СУБД Ред База Данных функционирует в виртуальной среде, то состояние файла БД на диске всегда консистентное и никак не мешает оперировать с состоянием виртуальной среды — делать снапшоты, сохранять ее состояние, мигрировать. Также можно делать снапшоты средствами файловой системы.

Доработки безопасности

Одним из слабых сторон многих проектов с открытым исходным кодом являются вопросы защиты данных. Мы провели анализ требований руководящих документов ФСТЭК и реализовали ряд доработок, которые позволили получить сертификат безопасности ФСТЭК.

К наиболее значимым доработкам относятся:

- Использование аутентификации пользователей, с поддержкой использования сертификатов PKCS7 или X.509, в том числе используя механизмы многофакторной аутентификации.
- Использование сертифицированных отечественных криптобиблиотек включая КриптоПро.
- Аудит событий базы данных в соответствии с нормативными требованиями ФСТЭК России на основе модуля FBTrace, а также мониторинг производительности работы сервера для целей оптимизации взаимодействия прикладных систем с СУБД и СУБД с ОС.
- Контроль доступа к системному каталогу.
- Обезличивание освобождаемой памяти, что исключает получение доступа к участкам памяти, которые остались после завершения процесса.

В новой версии СУБД Ред База Данных 2.6 реализована интеграция мандатной системы контроля доступа ко всей иерархии объектов БД с использованием меток SELinux и его библиотек. Другими словами, метка пользователя в SELinux, распространяется и на объекты внутри БД: таблицы, записи, столбцы и другие объекты. Соответствие меток и прав доступа управляется с помощью политик SELinux и компания Ред Софт выступила контрибьютором соответствующих доработок сделав несколько соответствующих патчей в проекты SELinux и SELinux Reference Policy для поддержки объектов специфичных для СУБД Ред База Данных.

Реализация мандатной системы контроля доступа выполнена в виде подключаемых плагинов, что позволяет добавлять способы таких проверок как независимых, так и основанных на реализации ОС, отличной от SELinux.

Опыт эксплуатации

Данные доработки позволили использовать СУБД «Ред База Данных» в ряде крупных проектов.

Наиболее ярким примером служит «Автоматизированная информационная система федеральной службы судебных приставов», положительно отмеченная председателем правительства РФ в 2014 (`http://government.ru/news/10513`).

Структура ФССП России включает три уровня:

1. Центральный аппарат ФССП России
2. 85 территориальных органов ФССП России
3. 2561 структурных подразделений ФССП России

Ежегодный документооборот службы превышает 1,2 млрд. документов. Непрерывный сбор информации со всей службы выполняется федеральным центром обработки данных, через который также осуществляется основной объем межведомственного взаимодействия в электронном виде.

Максимальный объем БД достигает 10ТБ. Работа и техническая осуществляется в режиме 24/7 во всех часовых поясах страны.

Награды

Корпорация «Ред Софт» победила в конкурсе «Лучший свободный проект в госсекторе — 2011» (`http://www.raspo.ru/content/28.html`) в двух номинациях — «Внедрение свободного ПО на рабочих станциях» и «Внедрение свободного ПО на серверах» с проектом «Доработка, разработка, внедрение и сопровождение подсистем автоматизированной информационной системы Федеральной службы судебных приставов в 2011 году».

Заключение

Эффективность эксплуатации СУБД в значительной степени зависит от степени ее интеграции с серверной ОС, на которой строится инфраструктура информационной системы предприятия или ведомства. СУБД Ред База Данных обладает всеми необходимыми инструментами для этого.

Леонид Юрьев
Москва, Петер-Сервис РнД, Сколково

Проект: ReOpenLDAP `https://github.com/ReOpen/ReOpenLDAP`

Кластерный LDAP для «Больших телекомов»

Аннотация

Производственная необходимость и обстоятельства подтолкнули Петер-Сервис использовать OpenLDAP в своих решениях, а затем заставили добиться «от этого кошмара» надёжной работы в нагруженном кластере.

Увы и ах, но общеизвестный проект с открытым исходным кодом и 25-летней историей, лежащий в основе LDAP как технологии, оказался хорошим примером «так делать НЕ надо». Но отступать нам было не с руки...

В результате мы сделали клон исходного проекта и за год получили LDAP-сервер относительно пригодный для промышленной эксплуатации в сфере телекоммуникаций: десятки и сотни миллионов записей, высокие нагрузки, высокая доступность, 24×7.

Вход в кротовую нору: Как мы умудрились этим заняться?

Компания Петер-Сервис является разработчиком и поставщиком решений для для крупных операторов связи более 20 лет.

В свою очередь, LDAP «прописан» в стандартах и рекомендациях как интерфейс взаимодействия многих Telco-подсистем, а сервер LDAP является одним из необходимых «кубиков». Поэтому мы решили получить свой «кубик».

В нашем случае к LDAP-серверу предъявляется несколько далеко не рядовых требований:

- Производительность порядка 10–100K запросов в секунду почтению и порядка 5–50K по-записи, с перспективой роста.
- Возможность построения географически разнесенного кластера, как минимум из четырёх узлов (2+2), с сохранением производительности.
- Высокие требования к стабильности...

Но прежде чем делать свой «велосипед» решили посмотреть на готовые решения, особенно с открытым кодом.

Критерии выбора были следующе: производительность, поддержка multi-master репликации, достаточно широкое использование и приемлемая история проекта. В список кандидатов попали: 389DS, OpenLDAP, ApacheDS и OpenDJ.

При тестировании производительности ни один из кандидатов не справился с нагрузкой как нам хотелось. Но в фавориты вышел OpenLDAP, так как показал[1] самую высокую производительность по-чтению. И в отличие от всех других кандидатов, не деградировала при подаче нагрузки по-записи.

Такое отличие хорошо объяснялось теоретически, так как в OpenLDAP доступен собственный движок LMDB[2], работающий в модели MVCC[3] и показывающий в тестах рекордную[4] производительность по-чтению.

Дополнительно в пользу OpenLDAP сыграли ещё два немаловажных момента:

1. OpenLDAP позволяет разделить данные между несколькими разнородными хранилищами, в том числе реализовать кэширование и т.п.

2. LMDB имеет режим работы, в котором не фиксирует данные на диске непосредственно после каждой транзакции, а перепоручает это механизму виртуальной памяти ОС (mmap/msync с флагом MS_ASYNC). Это дает сверхбыстрое хранилище, которое будет сохранено ядром ОС при крахе приложения, OOM или kill -9.

Так мы залезли в кротовую нору...

Исходная диспозиция: Сказочная стабильность, facepalm от исходного кода

Задействовать OpenLDAP начали в одном из проектов. В то время сам OpenLDAP мы не дорабатывали, просто взяли последний релиз.

[1] `http://www.slideshare.net/ldapcon/benchmarks-on-ldap-directories`
[2] `https://en.wikipedia.org/wiki/Lightning_Memory-Mapped_Database`
[3] `https://en.wikipedia.org/wiki/Multiversion_concurrency_control`
[4] `http://symas.com/mdb/inmem/`

В ходе разработки были замечены первые проблемы, а при тестировании ситуация стала критической. Постепенно был выявлен букет всевозможных проблем:

- Падения: в коде сервера LDAP, в коде движка LMDB, при активации MALLOC_CHECK, при проблемах в опорной сети или перезапуске отдельных узлов кластера...
- Утечки памяти, много.
- Пробуксовка репликации, иногда кластер часами не мог синхронизироваться.
- Оперативное переполнение БД, когда в стационарном состоянии свободно более половины.

Взаимодействие с авторами не давало никакого ощутимого результата. А так как исходный код был доступен, мы решили попробовать самостоятельно решить хотя-бы часть проблем.

Нет смысла перечислять все замеченные «особенности» исходного кода OpenLDAP, мы просто озвучим наши выводы:

- OpenLDAP — это один из образцов как НЕ следует разрабатывать программное обеспечение.
- Имея открытую лицензию, OpenLDAP не ориентирован на развитие силами сообщества.
- Объём технического долга и непрозрачность стиля кодирования, не только затрудняют какое-либо участие (очень плохая кривая обучения), но и отбивают желание стоять рядом.

К сожалению, в ответ на умеренную критику и замечания авторы выдавали перлы: «*coverity смешон*», «*отключите предупреждения, они не нужны*», «*Вы не понимаете деталей дизайна, вникайте*». При этом вступая в дискуссии и делая ошибки, которые ставят под сомнение глубокое владение темой. Так например, до демонстрации проблем (ITS#7969[5], ITS#7970[6]) утверждалось, что в движке LMDB, который работает по принципам lockfree в разделяемой памяти, не нужны барьеры памяти (memory barriers) и спецификаторы volatile.

[5] `http://www.openldap.org/its/index.cgi/Software%20Bugs?id=7969` Globally shared fields of meta-data are not 'volatile'.

[6] `http://www.openldap.org/its/index.cgi/Software%20Bugs?id=7970` Critical Heisenbug — Inconsistent reading & SIGSEGV due to the race condition.

Но отступать мы не собирались и решили продолжить битву. Однако, достаточно быстро получили отказ вливать наши изменения, и это заставило нас задуматься о создании собственного клона проекта.

Почему форк, зачем ещё один клон?

Дело в том, что при сборке OpenLDAP генерируется около 5000[7] предупреждений[8]. Один из наших первых патчей устранял из них порядка 90%, за счёт использования variadic macros из набора C99. В результате *становились заметны реальные ошибки*, которые ранее просто терялись в море предупреждений.

В качестве причины отказа было названо использование именно *variadic macros из C99*, что конфликтовало с целями целях проекта, где зафиксирована поддержка самого широкого спектра платформ и C89.

В ответ мы предоставили краткий обзор компиляторов C, из которого следовало:

- В настоящее время нет поддерживаемых платформ или компиляторов, в которых не было бы поддержки *variadic macros*.
- В большинстве компиляторов возможность *variadic macros* поддерживаются более 10 лет.

Но это не дало результата и мы пришли к решению сделать форк проекта. При этом было решено отбросить максимум «старья», не распыляться и сосредоточиться только на поддержке наиболее актуальной платформы: компиляторов gcc и clang (либо совместимых) и современных[9] версий Linux[10].

В продолжение (или в преддверие) дискуссии о целесообразности и оптимальности создания клона проекта, хочется отметить:

- Упомянутый патч не был использован даже для какой-либо тестовой сборки. Все предупреждения, в том числе о явных ошибках, не были проанализированы и остались в OpenLDAP на своих местах.

[7] Пять тысяч, Карл! Пять тысяч!

[8] При использовании компиляторов gcc версий 4.4.7-5.2.1 и clang 3.3-3.6, с установками генерации предупреждений по-умолчанию.

[9] Имеются в виду активно эксплуатируемые версии, грубо говоря не старше RHEL6.

[10] Поддержка FreeBSD также рассматривалась, но меинтейнера не нашлось.

- Выяснилось что OpenLDAP никак не тестируется на устаревших платформах, с которыми декларируется совместимость. Нет никакой информации о последних успешных сборках, о попытках таких сборок.
- Весьма вероятно, что для всех платформ, на которых может работать OpenLDAP доступен[11] компилятор gcc или другой альтернативный с поддержкой *variadic macros*. Поэтому, весьма вероятно, что вливание патча всё равно позволило бы собрать OpenLDAP для устаревших систем.

Таким образом, вместо наведения элементарного порядка в проекте и создания условий для выявления ошибок, в OpenLDAP в приоритете «абстрактная» совместимость с устаревшими, реально не эксплуатируемыми платформами. Причем эта мнимая «совместимость» никак не тестируется, никак не отслеживается её востребованность и вообще какая-либо необходимость.

Поэтому мы считаем и настаиваем, что в OpenLDAP произошло замещение исходных полезных целей проекта мнимыми, и обеспечение совместимости с «неуловимым Джо» яркий тому пример. Just non reasonable.

Год сурка: Тестирование, корки, медитация, правки

На устранение проблем и доработки было затрачено около года. За это время устранено порядка полусотни различных проблем. Некоторые[12] из ошибок ждали нас в багтрекере OpenLDAP по многу лет.

Оказалось, что наш сценарий использования отстоит очень далеко от обычных режимов работы OpenLDAP. Соответственно, мы наступили на большинство «грабель», которые проявляются при высокой нагрузке (race conditions), больших объёмах данных (повреждение памяти), стрессовых ситуациях (разрывы соединений), мульти-мастер репликации (логика).

Часть из исправленных проблем задокументированы как issues на github, но многие мы исправили молча. Особенно это относится к проблемам инфраструктуры и тестам.

[11] Это утверждение не удалось ни опровергнуть, ни подтвердить, по причине отсутствия у нас необходимого парка оборудования и ПО

[12] `http://www.openldap.org/its/index.cgi/Software%20Bugs?id=8012`/`http://www.openldap.org/its/index.cgi/Software%20Bugs?id=5452` SIGSEGV while disconnect/abandon.

Отдельно можно упомянуть про механизм репликации. Кроме технических ошибок, в нём оказалось несколько высокоуровневых изъянов[13] в логике алгоритмов.

После исправления ошибок текущая версия ReOpenLDAP не падает в условиях стрессового нагрузочного тестирования. Это банальный, но очень труднодостижимый результат. Как уже отмечалось, стиль кодирования не располагает к анализу исходного кода — очень часто это ребус в виде спагетти написанный на птичьем языке.

Не так давно на «Хабре» удачно опубликовали руководство[14] по стилю кодирования для OpenLDAP. Но я убежден, после завершения проекта ещё останется достаточно материала для написания книги *«Как НЕ надо делать софт»*.

Результаты доработок

Сделанные доработки можно разделить на шесть групп:

- Наведение минимального порядка. Проект успешно собирается с опциями -Wall -Werror. При этом было выявлено более 20 ошибок различной тяжести, о которых компилятор предупреждал всегда, но ранее эти предупреждения терялись в море менее опасных.

- Исправление ошибок. Перечислять их скучно и малопонятно. Их много, и многие потрясают «а как вообще могло работать». Механизм репликации частично переписан.

- Доработка инфраструктуры, как например стабилизация тестов или встроенный контроль динамической памяти. Пришлось потрудиться чтобы тесты можно было зациклить и они «не падали» через 5–10 итераций из-за своих собственных проблем.

- Специфические доработки заказанные нашей службой эксплуатации. Так например, в конфигурации сервера можно задать предел использования оперативной памяти и включить запись backtrace вместо создания coredump.

[13] ReOpenLDAP `https://github.com/ReOpen/ReOpenLDAP/issues/43`

[14] Как писать код, который никто не сможет сопровождать `http://habrahabr.ru/company/friifond/blog/268063/`

- Доработка внутренностей движка LMDB. Собственно это отдельная тема, которой будет посвящен доклад[15] на конференции Highload. Здесь можно кратко перечислить: полностью переписан путь сброса данных на диск с устранением двух принципиальных дефектов, steady и weak точки фиксации, OOM-handler при исчерпании места в БД, LIFO-режим для Garbage Collector, утилита mdb_chk для проверки образа БД.
- Доработки slapd, в том числе механизма репликации и механизма манипуляции данными основанного на LMDB.

В последнем пункте есть несколько ключевых доработок, на которых хочется остановиться чуть подробнее:

1. *Механизмы oom-handler и dreamcatcher* — без пенальти полностью решают так называемую «проблему отставших читателей».

   Суть проблемы в том, что в LMDB любая долгая (открытая, приостановленная) транзакция чтения на фоне изменения данных может приводить к быстрому исчерпанию места в БД и отказу сервиса. Проблема усугубляется тем, что такими «медленными читателями» является процедура горячего резервного копирования и процессы репликации.

   Поэтому без этих механизмов OpenLDAP не пригоден для промышленной эксплуатации.

2. *Механизм biglock*. Название отражает суть, это «большая блокировка по записи».

   К сожалению в OpenLDAP есть неустранимый архитектурный дефект, который может приводить к неверной репликации. Biglock позволяет защитить фазу трансляции UUID в DN-ключи и последующие операции, гарантируя этим целостность.

   Включение Biglock даёт небольшой пенальти (рекурсивный мьютекс) для операций записи/модификации, но это крайне незначительная величина, так как сам движок LMDB поддерживает только одного «писателя»и всё равно сериализует все транзакции записи.

3. *Механизм «Кворума» репликации*. Кавычки здесь крайне важны поскольку никакого протокола или алгоритма согласования

[15] Highload++2015, Движок LMDB — особенный чемпион `http://www.highload.ru/2015/abstracts/1831.html`

(Paxos) не используется. Однако «кворум» позволяет перевести узел кластера в readonly-режим при нарушении связанности, во избежания дальнейшего расхождения данных на узлах кластера.

4. *Ограничение конкуренции сеансов синхронизации.* Это отдельная «вишенка на торте» в возможностях «Кворума», которая позволяет в разы уменьшить пиковое использование оперативной памяти и CPU, сократить время синхронизации кластера при восстановлении связанности.

5. *Автоматическое формирование контрольных точек* (сброс данных на диск), как по объёму накопившихся изменений, так и периодически с секундной точностью.

Комбинация всех доработок, включая допеределку LMDB, позволяет получить производительность в *5–50 раз* больше оригинального OpenLDAP. Дабы не ввести в заблуждение, специально отмечу, что такой разгон возможен:

- За счёт пакетной фиксации транзакций на диске (управляемый компромисс);
- При наличии у дисковой подсистемы кэша обратной записи, который начинает «работать» благодаря доработкам LMDB (LIFO Reclaiming, Checkpoints).
- В стадии тестирования находится доработка устраняющая лишние обновления набора отметок contextCSN в ходе репликации. В зависимости от конфигурации кластера это может в 2-3 раза снижать количество фиксируемых транзакций. Соответственно, пропорционально снижается write amplification, нагрузка на диски и повышается интегральная производительность.

Стоит отметить, что все добавленные и протестированные возможности документированы в русскоязычных и англоязычных man-страницах.

Отдельно хочется поблагодарить *Егора Левинца*[16] за предоставленный перевод man-страниц.

[16] `http://www.pro-ldap.ru/`

Что дальше?

Мы вложили массу сил и времени в реанимацию ReOpenLDAP и вывод его на орбиту. В наших сценариях использования этот проект гораздо более стабильный и работоспособный в сравнении с исходным OpenLDAP, особенно переработанный механизм репликации.

Но к сожалению, субъективная оценка качества исходного кода и объёма технического долга заставляет сделать вывод о том, что OpenLDAP следует не чинить и развивать, а похоронить или переписать с чистого листа.

Стратегически Петер-Сервис уже решил в перспективе мигрировать на более стабильные по дизайну решения, и с более качественным исходным кодом. Так в качестве LDAP-платформы рассматривается OpenDJ, а в качестве оперативного хранилища Tarantool. Возможно, будет принято решение о подключении к ним движка WiterTiger или собственного, основанного на идеях LMDB.

Какое-то время ReOpenLDAP будет поддерживаться силами Петер-Сервис. Мы планируем исправлять замеченные ошибки, а также портировать исправления из OpenLDAP. Скорее всего, эта работа будет продолжаться до момента вывода из эксплуатации сервисов работающих на основе ReOpenLDAP.

После мы постараемся передать проект сообществу, если конечно будет соответствующая заинтересованность.

Андрей Михеев
Москва, Процессные технологии
http://www.runawfe.org

Свободная система управления бизнес-процессами и административными регламентами RunaWFE. Новые возможности версии 4.2

Аннотация

RunaWFE — свободная, ориентированная на конечного пользователя система управления бизнес-процессами и административными регламентами. В настоящем докладе описаны изменения, вошедшие в выпущенную летом 2015 г. версию 4.2

Система RunaWFE

RunaWFE — относится к классу BPM-систем. Основная задача таких систем — раздавать задания исполнителям и контролировать их исполнение. Вместе с заданием исполнителю передается требующаяся для его выполнения информация. Последовательность заданий определяется схемой бизнес-процесса, которую менеджер или бизнес-аналитик может быстро изменять при помощи среды разработки. Эта схема похожа на блок-схему алгоритма. По схеме перемещаются точки управления. В определенных узлах схемы генерируются задания исполнителям. Использование BPM-систем дает следующие преимущества: Существенно повышается производительность труда за счет исключения рутинных операций, связанных с получением и передачей информации между работниками, заметно упрощается деятельность по контролю выполняемых работ, повышается качество продукции предприятия, т.к. за счёт автоматической регламентации и средств мониторинга обеспечивается лучшее соблюдение всех предусмотренных правил, появляется возможность оперативно изменять бизнес-процессы в ответ на изменение условий деятельности предприятия, уменьшается стоимость работ по автоматизации (повышается скорость разработки и модификации программного обеспечения).

Изменения в версии 4.2 RunaWFE

1. Добавлена возможность настройки системы через веб-интерфейс

В меню системы добавлен пункт (см. Рис. 1). Новый пункт меню дает возможность настройки системы не только через конфигурационные файлы, но и через веб-интерфейс. Настройки разделены на несколько групп:

- Настройки веб-интерфейса
- Основные настройки
- Настройки работы с приложениями Office
- Настройки графа процесса
- Настройки полей представления
- Настройки календаря
- Настройки бот станции

Рис. 1: Меню настроек системы

2. Добавлена возможность делегирования заданий пользователя другим пользователям

Если пользователь по каким-то причинам не имеет возможности выполнить задание, то он может делегировать задание другим пользователям. Возможность делегирования заданий задается в настройках системы. В версии 4.2 кнопка делегирования находится в форме задания рядом с кнопкой «взять на выполнение» (см. Рис. 2). Если делегирование запрещено, то кнопка делегирования не показывается. В случае делегирования можно в задании открыть список исполнителей (как пользователей, так и групп пользователей), на которого у данного пользователя есть права, и поставить галочки тем, кому делегируется задание (см. Рис. 3). Также в задании есть признак, оставлять ли задание текущим исполнителям. Если в делегировании выбран один исполнитель и значение признака равно «false», то выбранный исполнитель назначается на роль. Если в делегировании выбрано более одного исполнителя или значение признака равно true, то создается временная группа, которой инициализируется роль.

3. Добавлена возможность обновления исполняемого экземпляра бизнес-процесса на следующую версию определения

Иногда для бизнес-процессов, выполняющихся длительное время, требуется внести изменения в уже выполняющиеся экземпляры. Для таких случаев в версии 4.2 была добавлена возможность обновления экземпляра бизнес-процесса на следующую версию. При обновлении все точки управления переносятся в новую схему в узлы с такими же идентификаторами узлов, в которых точки управления находились в

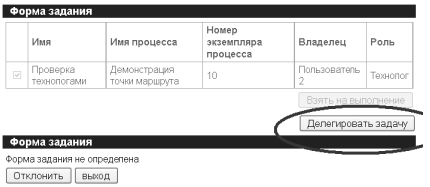

Рис. 2: Кнопка делегирования задания

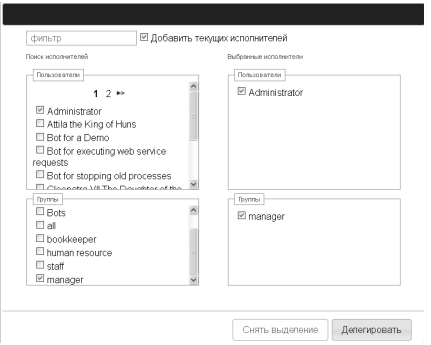

Рис. 3: Форма выбора исполнителей для делегирования задания

старой схеме. Обновление версии происходит в форме свойств экземпляра бизнес-процесса. При включении в настройках режима обновления на новую версию в свойствах появляется ссылка «Upgrade to next definition version». См. Рис 4.

5. Добавлена возможность выбора переходов для которых будут применяться правила проверки значений, введенных в поля формы задания

На Рис. 5 приведен пример бизнес-процесса, в котором из одного узла-действия выходит два перехода. На Рис. 6 для этого бизнес-процесса задано правило проверки значения, введенного в поле «значение_i» (значение принадлежит интервалу от 10 до 20), которое будет применяться только в случае выбора перехода «Способ 1».

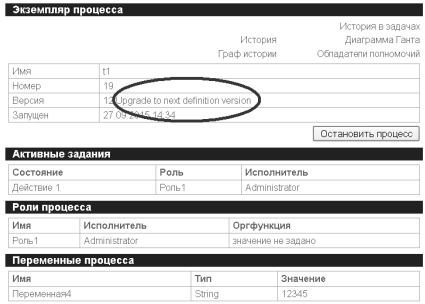

Рис. 4: Ссылка для обновления исполняющегося экземпляра бизнес-процесса на новую версию

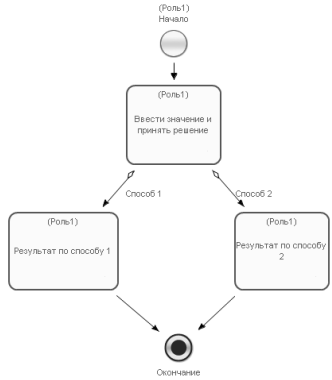

Рис. 5: Пример бизнес-процесса, на схеме которого два перехода выходят из одного узла-действия

6. Расширены возможности работы с Excel — документами. Добавлен режим последовательной обработки заданий бота

В версии 4.2 для целей обучения добавлена возможность использовать листы документов Microsoft Excel в качестве хранилища данных. Для того, чтобы при этом избежать конфликтов при одновременном изменении данных несколькими экземплярами бизнес-процессов,

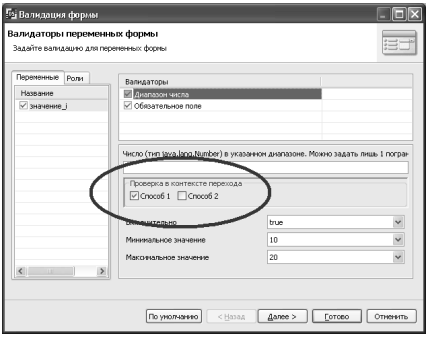

Рис. 6: Выбор переходов (путем проставления «галочек»), при выборе которых будет производиться проверка введенных значений в поля формы задания

в версии 4.2 был добавлен режим последовательной обработки заданий бота.

Ссылки

1. Ссылка на сайт проекта RunaWFE: `http://runawfe.org/rus`

Бронников Сергей Викторович
Москва

Проект: OpenVZ project `https://www.openvz.org`

Когда уже OpenVZ будет в основном Linux ядре?

Аннотация

Доклад о том, что делать форк открытого проекта вредно для здоровья, потому что поддержка форка требует больших усилий. Докладчик расскажет об опыте работы с разработчиками основной ветки Linux ядра в плане интеграции патчей для контейнерной виртуализации, наших успехах в этом направлении и предстоящих планах.

Проект по созданию открытой реализации Linux контейнеров — OpenVZ — был начат компанией SWsoft начат в 2015 году. Вся разработка была сосредоточена на двух основных компонентах: утилите

для управления контейнерами (vzctl) и Linux ядре (vzkernel) с набором патчей, реализующих функциональность контейнеров. Если утилита для управления контейнерами изначально была проектом созданным с нуля, то ядро было форком ванильного ядра. По мере развития функциональности контейнеров увеличивался и набор патчей и при выходе новой версии ядра разработчики переносили весь набор патчей на новые версии ядер. Но в какой-то момент количество патчей стало таким большим, что усилия по поддержке этих патчей стали отнимать много времени и ресурсов и встал вопрос по интеграции этих патчей в основную ветку Linux ядра.

Работа с разработчиками основной ветки всегда требует времени, потому что редко когда патчи принимают без переделки. Но несомненный плюс такой работы в том, что усилия по переносу набора изменений на новую версию будут требовать меньше сил.

В 2009 году разработчики начали процесс адаптации патчей vzkernel для основной ветки ядра. В первую очередь были добавлены такие компоненты как пространства имён PID и NET. Позднее к ним добавились memory cgroups, device cgroups, патчи для реализации «живой» миграции процессов, виртуализация NFS.

В том же году компания Parallels Inc. появилась в десятке самых крупных контрибьюторов ядра Linux согласно отчёту Linux Foundation. За всё время было принято более 2000 патчей от разработчиков OpenVZ. Эта работа по адаптации наших изменений продолжается до сих пор и на данный момент мы снизили размер патчсета в четыре (!) раза.

В нашем понимании, идеальное светлое будущее — это когда OpenVZ патч к ядру будет нулевого размера, то есть мы хотим, чтобы вся функциональность, которая есть в OpenVZ, появилась в ванильном ядре. Когда это наступит? Я боюсь, что никогда, ибо мир неидеален. Но если, скажем, в ванильном ядре будет 90% нашей функциональности — мы будем счастливы (сейчас там примерно 60%, точнее сложно сказать).

Емельянов Павел Владимирович
Москва, Один
Проект: CRIU, OpenVZ http://criu.org

Живая миграция контейнеров: плюсы, минусы, подводные камни

Аннотация

Доклад о том, как делать живую миграцию виртуальной среды вообще, и контейнеров в частности. Я расскажу о том, почему живая миграция является более сложной задачей, чем простое копирование содержимого памяти процессов, об её особенности в случае с контейнерами и о существующих готовых решениях.

Живая миграция — это процедура переноса виртуальной среды с одного компьютера на другой. В широком смысле данная тема не является новой и достаточно хорошо освещена в научной фантастике, где она называется телепортацией — объект помещается в гипотетическое устройство-телепортатор, сканируется, информация передаётся на пункт-приёмник, после чего объект оказывается в другом месте пространства. Применительно к виртуальным средам этот процесс выглядит точно так же — среда анализируется, её состояние сохраняется в специальном формате, передаётся на принимающий компьютер, после чего среда восстанавливается. Для стороннего наблюдателя это выглядит как «перемещение» среды с одного компьютера на другой.

Применение таких манипуляций на практике может быть оправдано. Во-первых, живой перенос рабочих программ с компьютера на компьютер производит сильное впечатление на неподготовленного пользователя. Во-вторых, возможность передвигать виртуальные среды между компьютерами позволяет перераспределять нагрузку между последними. В-третьих, освободив компьютер от выполняющихся на нём программ, можно произвести обновление системного ПО или аппаратуры.

Тем не менее, как и в случае с телепортацией, ошибки при миграции могут привести к непредсказуемым или нежелательным последствиям, по-этому для решения этих задач могут применяться другие средства. Например, в ряде случаев балансировку нагрузки можно

производить перебрасывая сетевые потоки с сервера на сервер. Обновление ПО или железа можно делать «по расписанию», заранее предупредив всех заинтересованных лиц о временном выключении, или по мере внеплановых остановок серверов, т.е., по просту, падений или зависаний. Кроме того перенос среды можно заменить её перезапуском в случае, если вся система выполнена в архитектуре микросервисов — небольших программ, не имеющих важного внутреннего состояния, допускающих частые перезапуски и сосуществование нескольких запущенных экземпляров («клонов»).

Тем не менее, технология живой миграции существует и реализована в рамках проекта CRIU [2]. Для понимания особенностей живой миграции контейнеров проведём её сравнение с живой миграцией более простого объекта — виртуальной машины. Для корректности сравнения, отметим, что полная процедура миграции включает в себя много стадий, а именно — заморозка контейнера, сбор и сохранение состояния, копирование состояния на принимающий сервер, восстановление состояния и разморозка.

Все эти этапы в случае миграции виртуально машины оперируют небольшим количеством объектов и достсточно просто раскладываются на ряд несложных действий. А именно: заморозка машины — это процедура погружения её в сон, сбор состояния — это последовательный обход конечного набора виртуальных устройств и сохранение информации о них, восстановление — создание устройств и запись в их виртуальные регистры или области памяти, разморозка — простое продолжение работы виртуальных процессоров.

Для контейнеров задача усложняется. Заморозка контейнера — это сбор и остановка всех процессов, причём в течение этого действия неостановленные ещё процессы могут порождать новые. Сбор состояния контейнера — это сбор информации о графе объектов: процессов, файлов, сетевых соединений, памяти (которой в Линукс может быть 4 типа), и т.д. Количество объектов в графе на несколько порядков может превышать количество виртуальных устройств в машине, а процедура сбора их состояния не универсальна и зависит от вида и особенностей предоставляемого ядром интерфейса. Восстановление состояния контейнера — это восстановление состояния графа, причём возможности создания новых объектов в нём строго ограничены (задача по сложности сходна с задачей разбора контекстно-зависимой аттрибутной грамматики).

Кроме того, важно отметить, что при таком простом подходе размер собранного «состояния» может оказаться огромным из-за присутствия в нём содержимого памяти процессов. Это, в свою очередь, может привести к тому, что время передачи состояния займёт слишком много времени, в течение которого контейнер будет остановлен (заморожен). Для преодоления этого недостатка применяются две технологии — предварительного или последующего копирования памяти. Обе технологии возможно реализовать как для виртуальных машин, так и для контейнеров, но, в последнем случае, сложность будет выше. Связано это, в первую очередь, с тем, что память машины — это память одного процесса-гипервизора, а память контейнера — это память «разбросанная» по дереву процессов.

Несмотря на все сложности, технология миграции контейнеров была реализована в рамках проектов CRIU и P.Haul.

Дмитрий Левин
Москва, BaseALT

Проект: The GNU C library `https://www.gnu.org/software/libc/`

glibc: жизнь после Дреппера

Аннотация

2012 год оказался поворотным в истории the GNU C library — ключевой для GNU/Linux и GNU/Hurd библиотеки, ведущей свою историю с 1987 года. С уходом главного разработчика Ульриха Дреппера, в течение многих лет развивавшего проект в авторитарном стиле, glibc разрабатывается и поддерживается сообществом активных (и не очень) мантейнеров, а решения по спорным вопросам принимаются (или откладываются) через достижение консенсуса.

Предыстория

Летом 1987 года в проекте GNU, целью которого с самого начала являлось создание свободной unix-подобной операционной системы [1], и в котором на данный момент насчитывается 382 пакета [2], ещё не было [3] стандартной библиотеки C. В том же году Роланд Макграт приступил [4] к ее созданию, в следующем, 1988 году, FSF

анонсировали [5] почти полный набор функций ANSI C, а в начале 1992 года — совместимость [6] с ANSI C-1989 и POSIX 1003.1-1990.

Вплоть до 1995 года всю разработку библиотеки продолжал вести Роланд Макграт, хотя эпизодически уже появлялись патчи от других разработчиков.

Тем временем, предположительно [7], в начале 1994 года, разработчики ядра Linux создали форк the GNU C library под названием «libc». Развитие Linux libc продолжалось несколько лет параллельно с разработкой the GNU C library примерно до выпуска glibc версии 2.0 в начале 1997 года, после чего развитие Linux libc постепенно прекратилось [8], и дистрибутивы GNU/Linux мигрировали на glibc.

Ульрих Дреппер: от рассвета до заката

Первый патч для glibc от Ульриха Дреппера появляется в начале 1995 года, а в начале 1996 FSF уже отмечает [9] его существенный вклад в разработку glibc.

С 1996 года Ульрих Дреппер работает в Cygnus Solutions, его доля в разработке glibc продолжает расти, становится более заметным вклад других разработчиков. В том же году Роланд Макграт заканчивает свою многолетнюю работу над проектом GNU, и его активность в разработке glibc сильно снижается. В середине 1996 года автор выпускает свой последний тестовый релиз 1.93, после которого самым активным разработчиком и мантейнером the GNU C library становится [10] и долгие годы остается Ульрих Дреппер.

Ульрих Дреппер быстро превращается в единственного принимающего решения в glibc, и созданный несколько лет спустя, в 2001 году [11], руководящий комитет glibc уже не оказывает никакого видимого влияния на проект.

Под руководством Ульриха Дреппера the GNU C library от релиза к релизу существенно прибавляет в функциональности, но в то же время проект glibc становится, пожалуй, самым враждебным по отношению к сторонним контрибьютерам среди всех свободных проектов GNU/Linux, а Ульрих Дреппер приобретает репутацию самого грубого и заносчивого персонажа мира GNU/Linux СПО.

Осенью 2010 года Ульрих Дреппер уходит из Red Hat — компании, поглотившей Cygnus Solutions в начале 2000 года, и его активность в разработке glibc постепенно снижается, а остальных разработчиков, наоборот, увеличивается. Немногим ранее, после перехода проек-

та glibc на git весной 2009 года, разработчики получают возможность сопровождать ветки, в которых Ульрих Дреппер не принимает участия, ограничиваясь веткой master. В конце 2011 года многолетний мантейнер the GNU C library поставил свой последний тэг для версии 2.15, которую сам так и не выпустил, а весной следующего, 2012 года, участие Ульриха Дреппера в разработке glibc сошло на нет.

Сообщество разработчиков

Начиная с версии 2.10, выпущенной весной 2009 года, стабильные релизы glibc сопровождаются сообществом разработчиков в форме релизных веток.

С уходом авторитарного лидера надобность в руководящем комитете glibc пропадает, и весной 2012 года он самораспускается [4].

Перед проектом GNU за glibc числятся ответственными 10 человек, круг разработчиков ещё шире — на данный момент это около 50 комиттеров [12], решения теперь принимаются через достижение консенсуса [13].

На практике это не всегда работает гладко. Иногда приходится долго ждать, пока найдется тот, кто сможет отрецензировать нетривиальное изменение [14]. Система patchwork, внедренная специально для ускорения и упрощения community review, пока не очень помогает [15]. По некоторым изменениям, например, по tunables API, консенсуса не удается достичь уже более двух лет.

Однако все эти издержки кооперативной разработки, по-видимому, не очень отпугивают разработчиков. Отвратительная репутация glibc как проекта, где не место контрибьютерам, ушла в прошлое.

Таблица 1: Число коммитов, их авторов, число багфиксов по версиям

дата выпуска	номер версии	число коммитов	число авторов	число авторов 4/5 коммиттов	число исправленных ошибок
30.10.2009	2.11	408	44	3	55
03.05.2010	2.12	389	49	7	78
17.01.2011	2.13	261	32	5	43
31.05.2011	2.14	256	36	3	95

Таблица 1 — продолжение

дата выпуска	номер версии	число коммитов	число авторов	число авторов 4/5 коммиттов	число исправленных ошибок
25.12.2011	2.15	582	32	3	67
30.06.2012	2.16	1189	61	9	234
24.12.2012	2.17	880	56	13	148
10.08.2013	2.18	935	66	11	132
07.02.2014	2.19	902	78	15	179
07.09.2014	2.20	911	69	12	151
06.02.2015	2.21	583	61	13	101
05.08.2015	2.22	670	64	15	195

Конечно, the GNU C Library можно сделать ещё лучше. Есть несколько направлений, где, как утверждает автор glibc Роланд Макграт [16], помощь будет совсем не лишней:

- приоритизация, воспроизведение, анализ, и исправление ошибок;
- дополнение микробенчмаркинга;
- общесистемный интегральный бенчмаркинг;
- инфраструктура тестирования: чруты, контейнеры, ABI;
- инфраструктура проекта: wiki, buildbot, bugzilla, patchwork.

Много интересных задач разной степени сложности перечислено в главном TODO проекта [17].

Литература

[1] *Richard Stallman, 27 Sep 1983.* new Unix implementation. `https://groups.google.com/forum/#!msg/net.unix-wizards/8twfRPM79u0/1xlglzrWrU0J`

[2] All GNU packages. `https://www.gnu.org/software/software.html#allgnupkgs`

[3] Status of the GNU Project. June 1987. `https://www.gnu.org/bulletins/bull3.html#SEC11`

[4] *Roland McGrath, 26 Mar 2012.* glibc steering committee dissolving.
https://sourceware.org/ml/libc-alpha/2012-03/msg01038.html

[5] GNUs Flashes. February 1988.
https://www.gnu.org/bulletins/bull4.html#SEC8

[6] Project GNU Status Report. January 1992.
https://www.gnu.org/bulletins/bull12.html#SEC15

[7] libc4 forked from GNU libc version 1.07.4.
http://git.uclibc.org/uClibc/plain/docs/uclibc.org/FAQ.html?id=0a8f8bd708e743fe5c692e4acd32911151674ec3

[8] The End of libc5: A Mini-Interview with H.J Lu.
http://www.tldp.org/LDP/LGNET/032/tag_libc5.html

[9] Forthcoming GNUs. January, 1996.
https://www.gnu.org/bulletins/bull20.html#SEC24

[10] Forthcoming GNUs. January, 1997.
https://www.gnu.org/bulletins/bull22.html#SEC21

[11] *Mark Brown, 22 May 2001.* About the glibc steering committee announcement.
https://sourceware.org/ml/libc-alpha/2001-05/msg00274.html

[12] Maintainers (developers) for libc.
https://sourceware.org/glibc/wiki/MAINTAINERS

[13] Community Consensus.
https://sourceware.org/glibc/wiki/Consensus

[14] [PING9̂][PATCH 3/2] Use strspn/strcspn/strpbrk ifunc in internal calls.
https://sourceware.org/ml/libc-alpha/2015-02/msg00227.html

[15] Patchwork for the GNU C Library. Incoming patches.
http://patchwork.sourceware.org/project/glibc/list/

[16] *Roland McGrath, Carlos O'Donell, 19 Feb 2015.* Where do we need help? «The GNU C Library Update!»,
Linux Foundation Collaboration Summit 2015
https://sourceware.org/glibc/wiki/ProjectTalks?action=AttachFile&do=get&target=LFCS2015-glibc.odp

[17] Master TODO List
https://sourceware.org/glibc/wiki/Development_Todo/Master

Эльвира Хабирова, Дмитрий Левин
Москва, ВМК МГУ, BaseALT

Проект: strace `https://sourceforge.net/projects/strace/`

Поддержка multiple personalities в strace или как обеспечить корректную трассировку 32-битных программ на 64-битных архитектурах

Аннотация

Исторически поддержка multipers в strace была ненадежной. Некоторые системные вызовы совсем не поддерживали различия между моделями данных, например, между LP64 и ILP32; в таких случаях аргументы системных вызовов типов long, pointer, а также всех составных типов, содержащих long и pointer, печатались неправильно. Поскольку это в большинстве случаев единственное, чем различается трассировка системных вызовов для разных personality, верным решением было бы использовать один и тот же исходный код для разных personality.

Most 64-bit platforms use 'long', while most 32-bit platforms use '__u32'. Yes, they differ in signedness as well as size. Special cases can override it for themselves – except for S390x, which is just a little too special for us. And MIPS, which I'm not touching with a 10' pole. [1]

Что такое strace

strace — утилита для отладки программ под Linux. Отображает системные вызовы, вызванные трассируемым процессом (tracee), а также полученные tracee сигналы и изменения его состояния.

strace активно использует системный вызов `ptrace()`. Он позволяет производить чтение и запись любого места адресного пространства и регистров tracee, останавливать выполнение по сигналу, системному вызову или пошагово, и т. д. Для более подробной информации см. `man 2 ptrace` и [2] (последний содержит тайное знание и рекомендуется к прочтению).

С помощью `ptrace()` при входе в обработчик системного вызова strace останавливает tracee и извекает *из адресного пространства tracee* же аргументы, а при выходе — вернувшееся значение, тоже из адресного пространства tracee.

Проблема

Ядро Linux умеет поддерживать сразу несколько ABI. Например, при выполнении на процессоре с архитектурой x86_64 поддерживаются три ABI: x86, x32[1] и x86_64.

Обычно поддерживаемые ABI различаются используемыми моделями данных, например, в случае с x86_64: ILP32 (данные типов int, long и pointer в 32 бита длиной) для x86 и x32, LP64 (int длиной в 32 бита, long и pointer — в 64) для x86_64.

Очевидно, что ядро принимает системные вызовы от всех процессов, использующих совместимые ABI. Ядро содержит много магии по преобразованию аргументов системных вызовов в типы данных модели родного ABI [3], а также по обработке передаваемых из пользовательского пространства данных составных типов (см., например, [1]), содержащих multipers[2]-типы[3].

strace обычно запускается с personality ядра, а не tracee, и поэтому он исторически имел подобную же магию, но далеко не для всех системных вызовов. Особенно проблемно было обрабатывать multipers-зависимые составные типы данных.

В подавляющем большинстве случаев трассировка программ с разными personality различается только используемыми определениями multipers-типов.

DWARF

Как просто и максимально автоматизированно получить информацию о том, как выглядят составные типы данных в программах,

[1] x32 — ABI, использующее 4-байтные указатели, но позволяющее использовать некоторые преимущества x86_64, например, дополнительные регистры и более быстрые операции с плавающей точкой.

[2] Возможность запускать в одной и той же системе программы, соответствующие различным ABI, имеет разные наименования. Поддержка этой функциональности со стороны дистрибутивов часто называется multilib (из-за необходимости предоставлять библиотеки чужих ABI параллельно с собственными библиотеками) или multiarch. В strace используется термин personality для обозначения одного из поддерживаемых ABI, и термин multiple personalities / multipers / mpers для понятия поддержки одновременно нескольких personality. Этим же термином обозначается зависимость (определения типа, работы кода) от текущей personality.

[3] Новые системные вызовы в ядре теперь делают multipers-независимыми.

имеющих personality, отличные от родной ядру? Можно обратиться к DWARF.

DWARF — стандартизированный формат отладочной информации. Генерация DWARF-информации поддерживается в том числе в gcc. DWARF состоит из единиц информации — DIEs, Debugging Information Entries, и обладает древовидной структурой. DWARF содержит в себе описание функций и процедур, макросов, таблицу номеров строк кода, и многое другое, в том числе переменных и типов данных [4]. Самое важное — описание всех типов данных сводится по дереву к базовым типам или к их композиции.

Решение

Что это всё дает? Можно автоматически составить простейший `.c`-файл вида:

```
typedef struct unknown_struct unknown_t;
unknown_t mpers_target_var;
```

Скомпилировать его для целевой personality (`gcc -m32` / `gcc -mx32`, например) с отладочной информацией (например, `gcc -gdwarf-4`), и прочитать секцию `ELF .debug_info` (`readelf -debug-dump=info`).

В итоге, основываясь на полученной отладочной информации, можно рекурсивно построить определение целевого типа. Для strace был написан awk-скрипт, выполняющий эту задачу.

Во время сборки strace в случае возможности генерации кода для неродных personality строятся описания каждого multipers-типа для каждой дополнительной personality. Отдельные multipers-зависимые функции автоматически создаются в нескольких вариантах, по одному на personality (включая родную). То, какие функции использовать, определяется в зависимости от personality tracee во время трассировки.

Литература

[1] `Linux/include/uapi/asm-generic/statfs.h`

[2] `https://sourceforge.net/p/strace/code/ci/master/tree/README-linux-ptrace`

[3] `https://www.kernel.org/doc/Documentation/x86/entry_64.txt`

[4] `http://www.dwarfstd.org/`

Михаил Быков
Москва, diglossa.ru

О Полярной звезде

Аннотация

Тексты вообще — это обеспечение процесса чтения для нас. Текст должен начинаться с авторитетного текста в системе контроля версий — это документ. Все его последующие представления диктуются волей современного исследователя. А его воля определяется состоянием научного знания в данный момент. Поэтому число интерпретаций текста — бесконечно, причем они логически не совместимы друг с другом, поскольку несовместимы сменяющие друг друга научные теории. Среди всех интерпретаций есть единственная, которая не будет ни изменяться со временем, ни устаревать — интерпретация согласно состоянию знания, современному автору текста. Хорошим примером может служить вопрос о том, что такое слог для древнего автора. Пример аутентичной интерпретации слога можно видеть в простом синтезаторе речи для Санскрита, доступном по ссылке dict.diglossa.org.

Тексты вообще — это обеспечение процесса чтения для нас. Раньше были тексты на бумаге, теперь — на экранах компьютеров. Что такое электронный текст (просто текст для чтения) вообще? Сегодня вы его в одной форме видите и читаете, завтра в иной — это процесс. Где начало процесса («откуда начало движения»), и какова его цель — куда он развивается? И почему такой бардак с е-текстами? И люди до сих пор копируют какие-то неизвестные куски html в неизвестных источниках, и не имеют ни малейшей уверенности, что читают то самое. И думают, главное, что так и должно быть. Впрочем, аккуратные бумажные издания тоже были редкостью — мы читаем на бумаге, сторого говоря, неизвестно что.

Можно-ли привести текст — любой текст, но в первую очередь, конечно, классические тексты — к единому источнику? Конечно, можно, и для этого не требуется ни денег, и почти никаких знаний и технологий, а лишь капля ума и (коллективного) желания.

Текст — источник всего процесса, голый текст. Но текст — уже результат работы с документом. До текста у нас в руках есть манускрипт, его приводят в виде изображения (одну страничку, или фрагмент) — читать его нельзя. Этот манускрипт некоторые специалисты

превращают в е-текст. Пусть, например, — некий Вася отсканировал и распознал книжку, отличный пример. (Далее, однако, этот Вася неизбежно добавляет хидеры, футеры, иксемели-шмиксемели, звездочки, указывающие на примечания, ссылки на конкордансы — и далее по пословице — горе от ума. Сравните библиотеку Персея в университете Тафта и lib.ru. В последнем тексты хоть читать можно. В Персее — только изучать. Как избежать этой неизбежной порчи?)

Из манускрипта (документа) мы получили текст — источник всего дальнейшего процесса, голый текст. В системе контроля версий, ибо это процесс. В различных ветках могут лежать разные списки с разных манискриптов, для специалистов. А в ветке main, которую все видят — рекомендованный канонический вариант.

А примечания и аппарат, и addendum et corrigendum, и прочая, и прочая? Есть обязательные примечания, являющиеся частью самого текста. Например есть текст перевода, и переводчик публиковал свой текст вместе с примечаниями. И мы не имеем права их разорвать — это одно целое, это воля автора. Но тогда это уже не текст, а гипертекст? А текст не годится, слишком прост?

Нет, текст годится. Гиертекст — путь в никуда, это путь того Васи из нашего примера. Аппарат к тексту — сам по себе бесконечность. Типов примечаний может быть бесконечно много — вплоть до описания повреждения краев бумаги, пятен, и даже погоды и состояния самочувствия публикатора в момент работы с текстом (см. русские летописи). И все это — в одном XML, — и будет источник всего дальнейшего процесса работы с текстом? Абсурд. А иначе — что есть единый источник?

Нужно так: примечания лежат в текстовом же файле рядом, и, очевидно, привязываются к тексту якорями (уникальными указателями) в самом тексте. Якорь — уникальная подстрока, заканчивающаяся там, где в обычном тексте стоит звездочка с номером примечания. Она однозначно указывает место в тексте. И здесь же рядом должен быть отдельный скрипт, устанавливающий их соответствие. И так же может работать любой аппарат, который нужен публикатору. Можно описать пятна на бумаге/папирусе, варианты написания букв, отметить пропуски, etc, etc. Если к тексту есть дополнение, то должен быть и скрипт, их объединяющий. Естественно, скрипт — то, с чем работал публикатор — только пример. Другой человек, видя процесс, может написать свой на другом языке, etc, etc.

Второе: текст должен подтверждаться авторитетом специалиста, быть авторитетным. Тексты, которые лежат у меня на `diglossa.ru` — не авторитетны, я не специалист. Пушкинский дом должен выложить текст Пушкина, институт философии — текст Платона (русский перевод), etc, и — отвечать за него. Но специалисты в настоящее время представляют свою работу в каком угодно наукообразном виде, но категорически не хотят просто выложить текст, как основу своих результатов. Тексты выкладывают любители, религиозные организации, кто угодно, кроме науки. Никто, ни в одной стране мира. Это очень плохо. Если бы физики писали статьи, но не публиковали исходные данные — результаты эксперимента, им бы никто и копейки бы не дал.

Ту же задачу выполняет и развивает Диглосса[1] — быстро указать место в параллельном переводе, не обозначая никакой из них как правильный. Они все неправильные — правильно лишь понимать текст, не изменить текст своей/чужой интерпретацией, но позволить тексту изменить вас. (А вот явно неправильные, тоже, конечно, есть).

Все остальное, по моему, от лукавого. В том смысле, что диктуется чем-то извне текста. Текст — источник. Мы сейчас видим множество корпусов — но посмотрите на них, там есть все что угодно, кроме текста. Текст должен быть источником и целью процесса. Он должен лежать в системе контроля версий, на том же гитхабе, например. Как лежат тексты диглоссы[2].

Конечно, возникает масса практических вопросов. Например, если кто-то вносит изменение в текст (исправляет опечатку) — и исправления попали на место якоря, к которому привязано примечание. Комментарии собьются. Наверно, это решается множеством разных способов. Я думаю, что простейший и прозрачный для редактора текста — хук в git-е, а именно pre-commit-hook, который проверит эту коллизию (якоря и исправления), и исправит якорь. Якорь не принадлежит к тексту автора, его поправлять вполне можно.

Это, по-моему, типичная задача для СПО. Код. см. на гитхабе.

В диглоссе, при обновлении текста на гитхабе, автоматически обновляется интерфейс, с которым работает читатель данного текста. Это процесс преобразования источника текста в зримый результат, это процесс. Это автомат. То есть это множество программ, которые

[1] `http://diglossa.ru`
[2] `https://github.com/diglossa`

будут развиваться, дополняться, ветвиться, — это процесс на годы, и даже попросту вечный и бесконечный процесс. Это попросту часть процесса человеческого чтения. В бумажной книге то же самое — одно издание, другое, etc. Этот автомат и нужно создать — как постепенно появилось сообщество разработчиков СПО.

Итак, источник процесса ясен — текст в системе контроля версий. А что есть цель этого процесса? Текст плюс справочный аппарат.

Но для начала я расскажу, что не есть цель. Мы видим огромное количество корпусов текстов — их уже больше, чем песчинок в океане. И их авторы авторы будут убеждать всех, что их корпус как раз чтобы изучить, помочь понять, и помочь прочитать текст. Если не всем, то специалистам. Это, однако, не так.

Корпуса используют современные лингвистические теории, и, соответсвенно, понятия этих теорий. Т.е. существительное, фонема, etc. И теории, и понятия меняются со временем, развиваются. Это природа современной науки, ее метод. Соответсвенно меняется и финальная форма текста, то, что в конце концов видит читатель, интерфейс. Это тоже бесконечный и неизбежный процесс. В любой науке именно так.

Хуже всего то, что разные теории логически несовместимы друг с другом. Разница может быть незаметна, и может не сказаться при написании программы. А может быть заметна очень. Например, могут быть разные определения того, что такое слог. И разбить на слоги корпус, скажем, эпической индийской литературы — это работа, и гигабайты объема. И результат абсолютно ни логически, ни практически не совместим с результатом, полученным при другом определении слога. Я подробно разбираю этот пример в докладе на конференции Моргиналии в Полоцке.

Научные теории будут меняться, и следовательно, будет меняться справочный аппарат. Пока будет существовать наука, будет развиваться ПО для обеспечения этого процесса.

Однако чтение текста и изучение текста — существенно разные процессы. Чтение — это понимание текста. А понимание — это изменение, происходящее в нас, в читателях текста. Понимание — это разрешение автору текста изменить меня, читающего. Научное исследование текста вообще не ставит своей задачей понимание текста. Какая разница, что там пишет отсталый Аристотель в своей Физике. Для науки важно лишь как он пишет, сколько в тексте глаголов несовершенного вида, например, etc.

Но древний автор не для того писал свой текст, чтобы его изучали, он писал в надежде найти в веках читателя, а не вивисектора.

Следовательно, справочный аппарат быть должен, но это должен быть аппарат, с которым автор был бы согласен. Аппарат, современный автору текста. В идеале, для чтения текста читатель должен получить образование, адекватное знаниям автора. Современное автору образование. Это идеальная утопия, но существенные черты древнего знания справочный аппарат для современного читателя отражать обязан. Особенно расхождения с современной научной картиной мира.

Особенная опасность — в том случае, когда современная наука лингвистика говорит: это глагол, прошедшего времени, etc. И древний автор в этом месте сказал бы дословно то же самое, глагол, прошедшего времени, etc. Но само понимание того, что такое глагол, вовсе другое для него. Скажем, первое-главное в глаголе для автора — то, что он произнесен (вслух), сказан, про-глаголен, и рассказывает о событии, является рас-сказом. А вовсе не то, что вы подумали, или подумал современный лингвист.

Но это сложный случай, а в простых — попросту справочный аппарат должен дать справку, современную автору текста, адекватную тексту. Древнее образование всегда начинается с грамматики. Недаром в средневековье первым из семи свободных искусств идет тривиум (науки о слове), а в нем первым — граматика. То же и в веданге — первыми идут шикша — фонетика, чхандас — метр, и вьякарана — грамматика. Там вводятся понятия, фундаментальные смыслы, как звук, слог, имя, глагол, время, etc. Они появляются в грамматике, риторике — но лишь для того, чтобы тут же быть употреблены в философии, музыке, астрономии и богословии. Это фундаментальные понятия. Подменить из современными — для понимания текста — все равно, что подменить младенца в роддоме.

Научные институции вовсе не ставят себе такой задачи. Понимание текста вообще не является научной задачей. Поэтому нет надежды, что научные институции могут способствовать решению этой задачи. Для них древний учебник не фундаментальная основа понимания авторского текста, а устаревшая бесполезная теория (а это и не теория, кстати, вовсе), хлам, ветошь. Никто не станет изучать физику по Ньютону. Поэтому на научные институции в этом плане надежды у меня нет, и быть не может.

Также эта задача не под силу ни одному человеку, ни коммерческой организации. Но она под силу сообществу разработчиков свободного ПО. При условии если, или когда, найдется достаточное количество разработчиков, одновременно являющихся и людьми, заинтересованными в чтении древнего текста. Есть исключение — тексты религтозного содержания будут сопровождаться именно современным автору и адекватным аппаратом при электронном издании их религиозными же организациями. Для текстов любого иного типа (медицинских, математических, музыкальных, филологических) — надежда может быть только на саморганизацию читателей-разработчиков.

Утешает одно. В калейдоскопе мелькающих теорий древний учебник грамматики — константа. Это напоминает то, как в древнем мире в калейдоскопе и круговращении возникновений и уничтожений есть лишь одна неизменная, нетленная константа — Полярная звезда, arktos. Так и сейчас для нас, автор и его справочный аппарат — инвариант, он есть видимая и очевидная общая цель, telos, причина движения, наша Полярная звезда.

Александр Боковой
Эспоо, Финляндия, Red Hat Ltd.

Проект: FreeIPA `https://www.freeipa.org/`

Гномы на производстве

Аннотация

Доклад-демонстрация текущего состояния интеграции графической среды GNOME и интегрированной системы управления инфраструктурой предприятия FreeIPA.

Виталий Липатов
Санкт-Петербург, Etersoft
https://www.altlinux.org/Etersoft-build-utils

Как упрощается сборка пакетов в ALT Linux с помощью etersoft-build-utils

Аннотация

Чтобы прийти к качественной сборке пакетов за минимальное время, требуется пройти длительный путь, при этом учитывая происходящие изменения в процедуре сборки. Набор команд из etersoft-build-utils и предлагаемые типовые практики помогают сократить порог вхождения и облегчить затраты на первичную и последующую сборку пакетов.

Чтобы собрать даже простой пакет, нужно знать и учесть множество нюансов. С приобретением опыта сборка пакетов начинает казаться несложной, но рутинной задачей. С целью упростить сборку пакетов, то есть скрыть мелкие подробности и выполнить за человека рутинные задачи, и создаётся уже более 10 лет набор утилит `etersoft-build-utils`. Стоит упомянуть, что у каждого мантейнера с длительным опытом сборки пакетов собраны вспомогательные скрипты, которые ведомы только им, и не распространяются по причине их кустарности.

На самом деле собрать пакет очень просто. Берём git-репозиторий с пакетом и запускаем

```
$ rpmbb
```

если мы хотим собрать пакет в своей системе, или

```
$ rpmbsh
```

если мы хотим собрать пакет в hasher, или

```
$ rpmbs -u
```

если мы хотим отправить пакет на сборку в ALT Linux Sisyphus, или

```
$ rpmbph -b p7 -u
```

если мы хотим отправить пакет на сборку для ALT Linux p7.

При этом вам не нужно беспокоиться о временных коммитах, создании рабочего каталога для `hasher`, публикации репозитория, создании и публикации тэга для сборки, создании задания на сборку.

Но где же взять git-репозиторий с пакетом? Помимо очевидного решения с созданием такого репозитория с нуля самостоятельно, есть дополнительные пути:

Можно склонировать репозиторий из ALT Linux, если такой пакет уже собран:

```
$ rpmgp -g normalize
```

а можно посмотреть, нет ли такого пакета в других системах:

```
$ rpmgp -a normalize
List for altautoimports:
    nodejs-normalize-package-data-0.2.1-alt1_1.src.rpm
    ...
List for rosa2012:
    normalize-0.7.7-11.src.rpm
```

и скачать его, указав полное название файла, а потом преобразовать в git-репозиторий:

```
$ rpmgp -a -d normalize-0.7.7-11.src.rpm && rpmgp -m
    normalize-0.7.7-11.src.rpm
```

Если нужную вам программу вы вчера написали сами, и она ещё не собрана ни в одну из популярных систем, придётся писать спек для сборки самому. Но ни в коем случае не пишите спек с нуля. Возьмите подходящий пример отсюда: `https://www.altlinux.org/SampleSpecs` или загрузите репозиторий другого, уже собранного в ALT Linux Sisyphus пакета и используйте его в качестве образца.

Если сборка пакетов всё ещё кажется вам сложной, просто создайте задачу на добавление нового пакета: `https://bugzilla.altlinux.org/enter_bug.cgi?product=New%2Fproposed%20packages` И когда кто-то соберёт пакет, вы сможете его скачать и улучшать, чтобы им можно было пользоваться. Или просто соберёте новую версию пакета командой

```
$ rpmrb 4.32
```

которая скачает и закоммитит новую версию исходников, произведёт изменения в спеке, обновив версию и добавив запись в changelog, проверит собираемость в локальном hasher и отправит на сборку в Сизиф в случае успеха.

Виталий Липатов
Санкт-Петербург, Etersoft
http://wiki.etersoft.ru/Utils

Утилиты от Etersoft: упрощение с помощью обобщения и не только

Аннотация

По разным причинам в арсенале компании Etersoft оказался набор инструментов, разработанных внутри: менеджер пакетов epm, архиватор erc, инструмент для ведения параллельного upstream — gitum,средство создания резервных и архивных копий eterbackup, оболочка над git — giter. В докладе будет рассказано о каждом из этих инструментов,причинах их появления и пользе от их использования.

Количество инструментов для решения одной и той же задачи, а также скорость их изменения или смены не может не поражать. Для решения задачи сначала нужно заняться поиском, потом из многого выбрать подходящий или работающий инструмент, потом научиться им пользоваться. А ведь для этого нужно знать, что искать, а если часто пользуешься — запоминать, как использовать. А речь может идти о десятках, сотнях, а порой и тысячах программ.

При этом создатели инструментов обычно забывают о Unix-way: не используют совокупность инструментов для решения сложной задачи, а пишут свой велосипед, решающий все задачи сразу. И у них даже есть на это оправдания.

Возьмём к примеру простую задачу разархивирования файла. Да, расширение файла обычно подсказывает, какую программу задействовать. Её всего лишь надо установить, потом почитать справку и узнать, с каким ключом запускается разархивация. Можно попробовать пользоваться 7-zip, который поставляется с плагинами, поддерживающими некоторые форматы. А можно использовать erc. Вне зависимости от формата архива, создание и разархивирование происходит всегда одинаково, одними и теми же командами. Причём теми, к которым вы привыкли. Причём достигается это простым способом:

```
$ erc -h
erc - universal archive manager
Usage: erc [options] [<command>] [params]...
Commands:
```

```
a|-a|create|pack       create archive
e|x|-e|-x|u|-u|extract|unpack  extract files from
    archive
l|-l|list             list archive contents
t|-t|test|check       test for archive integrity
```

То есть программа поддерживает различные варианты параметров, один из которых точно используется в вашем привычном архиваторе.

На самом деле `erc` не самостоятельная программа, а оболочка над программой `patool`, которую разрабатывает `Bastian Kleineidam`, привносящая универсальную обработку параметров.

Такая же история с установкой пакетов в систему. Идея иметь один инструмент для управления пакетами на любой платформе пришла не сразу, но настойчиво стучалась в дверь. Когда ей открыли, раздалось множество голосов: зачем столько систем, нужно использовать одну, и не нужно будет изобретать ещё один пакетный менеджер; в MacOS нет пакетов, поэтому там не нужен пакетный менеджер; опять изобрели велосипед. Всё это говорило о неравнодушии к поднятой проблеме. Зато теперь можно установить пакет командой `epm install` или `epmi` (как вам удобнее), или определить принадлежность файла в системе к пакету:

```
$ epmqf vim
Note: vim is placed as /usr/bin/vim
 $ rpm -qf /usr/bin/vim
 предупреждение: файл /usr/bin/vim не принадлежит ни одному из
     пакетов
Note: /usr/bin/vim is link to /usr/bin/vim-neXtaw
 $ rpm -qf /usr/bin/vim-neXtaw
 vim-X11-neXtaw-7.3.353-alt4.1
```

не задумываясь, какую команду нужно применять, и какой сейчас в этой системе моден пакетный менеджер.

Последняя наша разработка на эту тему — набор утилит eterbackup, служащих для создания и обслуживания резервных и архивных копий. Там удалось применить несколько интересных наработок: начиная от журнального инкрементального дедублицирующего архиватора zpaq, создаваемого Matt Mahoney, до прореживания старых архивных копий с логарифмическим интервалом. Так же удалось найти компромисс между простотой доступа к архиву и количеством

файлов в одном архиве для таких данных, как почтовые ящики или домашние каталоги: для каждого каталога, начиная с определённого уровня вложенности ведётся отдельный инкрементальный архив.

Основная суть все этих разработок, примитивных по сути и написанных на Shell — фиксирование накопленного опыта. При этом это намного больше, чем «однострочник», решающий какую-либо задачу.

Применённый подход обвязок на Shell вокруг существующих команд также сглаживает различия между версиями и позволяет при необходимости компенсировать ошибки в используемых командах.

Вадим Кузнецов
Москва, Калужский филиал МГТУ им. Н.Э. Баумана, Московский институт электроники и математики НИУ «Высшая школа экономики»

Обзор свободного ПО для моделирования радиоэлектронных устройств и новых возможностей симулятора электронных схем Qucs.

Аннотация

Приведён обзор программных средств с открытым исходным кодом, предназначенных для моделирования различных радиоэлектронных устройств. Кратко рассмотрены основные возможности следующего ПО: Ngspice, Qucs, XYCE, TkGate, Gnucap и openEMS. Все данные проекты активны в настоящее время. Приведён обзор новых возможностей САПР для моделирования электроники Qucs, добавленных в версиях 0.0.19 и 0.0.19S. Рассмотрены перспективы развития САПР Qucs и вопросы интеграции проектов Qucs, Ngspice и XYCE. Автор статьи входит в команду разработчиков проекта Qucs. Данная информация будет интересна инженерам-разработчикам электроники, планирующим переход на open-source ПО и ОС Linux.

Введение

Существует не так уж и много open-source САПР. САПР электроники (Electronic Design Automation — EDA) являются исключением. Важной разновидностью данных САПР являются симуляторы электронных схем и электромагнитные симуляторы. За последнее время было сообществом было создано достаточно большое количество весьма достойных проектов. На сегодняшний день наиболее значимыми

из них являются: Ngspice, Qucs, XYCE, TkGate и openEMS. Ниже приведён их краткий обзор.

Сначала рассмотрим симуляторы цифровых и аналоговых схем. На сегодняшний день индустриальным стандартом описания моделей и списков цепей для аналоговых схем является SPICE, разработанный в университете Беркли в 80-е годы прошлого века. Исходный код многих симуляторов основан на оригинальном коде Berkeley spice3f5. Для цифровых схем стандартом является поддержка языков описания аппаратуры VHDL и Verilog.

SPICE-совместимые симуляторы аналоговых схем

На сегодняшний день полную совместимость со SPICE имеют следующие open-source симуляторы:

1. Ngspice [1] — консольный симулятор аналоговых и цифроаналоговых схем. Позволяет производить моделирование на постоянном и переменном токе, анализ переходных процессов, шумовой анализ, анализ нелинейных искажений, Фурье-анализ. Имеет полную совместимость со SPICE-моделями и существующим библиотеками электронных компонентов. Разработка проекта начата более 10 лет назад. Совместим по видам моделирования с популярным коммерческим пакетом PSpice. Существует версия с поддержкой расчётов на GPU — CUDAspice. Движок моделирования Ngspice написан на C и использует исходный код spice3f5, написанный университетом Беркли. Ngspice имеет постпроцессор Ngnutmeg со встроенным скриптовым языком, позволяющим выполнять сложный анализ электронных схем. Лидерами проекта являются Francesco Lanutti и Paolo Nenzi.
2. XYCE [2] — консольный кроссплатформенный SPICE-совместимый симулятор электронных схем. Xyce разрабатывается Сандийскими национальными лабораториями (США). Существуют для версии симулятора: с поддержкой параллельных вычислений через OpenMPI (XyceParallel) и без неё (XyceSerial). Версия XyceParallel доступна только для Linux. Отличается от Ngspice тем, что не имеет постпроцессора и не содержит некоторых видов моделирования (шумовой анализ, анализ нелинейных искажений), но зато содержит дополнительный вид моделирования — гармонический баланс. Симулятор написан на C++, с исполь-

зованием набора библиотек функций линейной алгебры Trilinos, разработанного в Сандийских лабораториях.

3. Gnucap — данный симулятор основан на движке моделирования, написанном с нуля. Пока не реализованы все виды моделирования и постпроцессор. Симулятор написан на C++. Сейчас проект разрабатывает Felix Salfelder.

Симуляторы цифровых схем

Среди симуляторов цифровых схем следует отметить TKGate. Это симулятор только цифровых схем на базе Verilog. Он работает только в ОС Linux. Симулятор написан на связке C и Tk/Tcl. Автором проекта является Jeffery P. Hansen (неактивен). Сейчас разработкой занимается наш соотечественник Андрей Скворцов. После шестилетнего перерыва в разработке в этом году вышел релиз TKGate-2.0.

Ключевые возможности симулятора следующие:

1. Интерактивное моделирование цифровых схем
2. Доступные типы компонентов: логические элементы, триггеры, регистры, счётчики, дешифраторы, мультиплексоры, ОЗУ, ПЗУ, АЛУ, ключи на МОП-транзисторе, имитаторы входных сигналов, светодиоды, семисегментыне индикаторы, шкалы, генераторы синхросигналов.
3. Использование пользовательских скриптов Verilog.

Код Verilog можно потом передавать из TkGate в САПР для синтеза ПЛИС.

Qucs — post-SPICE симулятор аналоговых и цифровых схем

Теперь рассмотрим симуляторы несовместимые со SPICE. К этому классу относится популярный симулятор Qucs.

Qucs [4,6] является симулятором аналоговых и цифровых схем, основанном на вновь разработанном движке моделирования Qucsator. Qucs написан на C++, имеет свой GUI, основанный на Qt4. Лидером проекта является Guilherme Brondani Torri. Qucs поддерживает все базовые виды моделирования: анализ на постоянном и переменном токе, моделирование переходного процесса [7]. Отличительной особенность данного симулятора является расширенные функции для моделирования схем СВЧ-устройств: моделирование S-параметров, специ-

альные модели СВЧ-компонентов и постпроцессор с расширенными функциями анализа комплексных сопротивлений в частотной области [8]. По возможностям анализа СВЧ-устройств Qucs приближается к таким проеприетарным аналогам как AWR MicrowaveOffice. Но симулятор Qucsator несовместим со SPICE и поддерживает SPICE только через слой совместимости, что значительно осложняет использование существующих библиотек моделей электронных компонентов. Несмотря на отличные характеристики движка Qucsator для моделирования СВЧ-схем в частотной, во временной области имеются многочисленные баги.

Возможно моделирование цифровых схем с применением языков описания аппаратуры Verilog и VHDL.

Электромагнитные симуляторы

Единственным представителем класса open-source электромагнитных симуляторов является openEMS [5]. Он не имеет своего графического интерфейса, а встраивается в Octave или Matlab. Задание на моделирование является скриптом Octave или Matlab, а для просмотра результатов служит программа Paraview.

Электромагнитный симулятор предназначен для расчёта распределения электромагнитного поля в различных системах и имеет следующие возможности.

1. Расчёт электромагнитного поля (электродинамика) методом FDTD (метод конечных разностей во временной области)
2. Расчёт S-параметров электромагнитных систем
3. Расчёт электромагнитного поля в ближней и дальней зоне
4. Расчёт диаграмм направленности антенн
5. Визуализация результатов

Преимуществом openEMS является то, что он встраивается в Matlab/Octave, и следовательно мы имеем неограниченные возможности по параметризации моделей. Недостатком openEMS является повышенный порог вхождения. Требуется обязательное знание Matlab/Octave.

Qucs-0.0.19S — новая версия Qucs с полной поддержкой SPICE

Qucs имеет интуитивно понятный интерфейс и развитую подсистему визуализации результатов. Но Qucs использует свой собственный формат списка цепей и моделей, несовместимый со SPICE. Это сильно осложняет использование существующих библиотек моделей полупроводниковых компонентов, распространяемых производителями. Слой совместимости со SPICE имеет многочисленные ограничения. Также ядро моделирования Qucs имеет многочисленные баги (зависание и крах симулятора) при моделировании во временной области, не позволяющие моделировать ключевые схемы и силовую электронику.

Ngspice, напротив, полностью совместим со стандартом SPICE и не имеет проблем при моделировании во временной области. Но Ngspice не имеет GUI и собственной системы визуализации. Поэтому возникает задача объединить Qucs и Ngspice так, чтобы можно было схему Qucs моделировать в Ngspice и использовать Qucs для визуализации результатов.

Данную задачу решает набор патчей spice4qucs [9,10,11], который в 2014-2015 году разработали я и Mike Brinson (London Metropolitan University). Spice4qucs позволяет моделировать схемы Qucs с использованием внешних консольных симуляторов Ngspice и XYCE. С нашим набором патчей Qucs может служить интерфейсом для Ngspice.

Исходный код находится в главном репозитории Qucs в ветке `rebase_spice4qucs`: `github.com/Qucs/qucs/tree/rebase_spice4qucs`
Итак, spice4qucs выполняет следующие функции:

- Преобразует схему Qucs в SPICE-netlist
- Запускает сторонний симулятор
- Преобразует вывод от симулятора в XML-формат данных Qucs

Spice4qucs не принят в upstream проекта Qucs. Поэтому было принято решение выпустить параллельно два релиза Qucs:

- Qucs-0.0.19 — Без spice4qucs. Используется только ядро моделирования Qucsator.
- Qucs-0.0.19S — С интегрированной функцией моделирования схем при помощи Ngspice и XYCE

Версия Qucs-0.0.19S позволяет решать следующие задачи, недоступные базовой версии Qucs-0.0.19:

1. Spice4qucs позволяет моделировать схемы, недоступные дефолтному симулятору Qucsator. Прежде всего это силовая электроника, ключевые схемы, схемы на полупроводниковых приборах, работающих с заходом в режим отсечки, и схемы с большим количеством компонентов.
2. Добавлено множество SPICE-совместимых компонентов и видов моделирования. Они расположены в группах: Spice components, Spice sections, и Spice simulations. Подробнее о них написано в документации. Все эти компоненты не будут работать с дефолтным симулятором Qucsator. Они реализуют задание параметров, начальных условий и команды построцессора в стиле Ngspice. Также добавлены модели полупроводниковых компонентов с полным описанием модели в формате SPICE. Это позволяет просто копировать модель из существующих библиотек, минуя конвертеры.
3. В библиотеках теперь разрешено включение SPICE-нетлиста. Он напрямую передаётся SPICE-совместимому симулятору, минуя конвертации.

Таким образом Qucs-0.0.19S может быть использован в промышленности инженерами-разработчиками электронной техники не только для моделирование СВЧ-устройств, но и любой другой электроники и составить конкуренцию симуляторам MicroCAP и Multisim.

Заключение

В данном обзоре были рассмотрены наиболее популярные активные opens-source проекты в области моделирования электронных схем. Можно сделать вывод, что СПО реализует все базовые виды моделирования, применяющиеся при разработке электроники: схемотехническое моделирование цифровых и аналоговых схем, синтез схем электронных устройств, электромагнитное моделирование. Но по возможностям СПО существенно отстаёт от проприетарных аналогов. Недостатками являются: консольный интерфейс, бедные библиотеки компонентов, недостаточный уровень поддержки промышленных стандартов. Все данные недостатки преодолимы при условии привлечения к данным проектам дополнительных разработчиков. Преимуществом является нетребовательность к ресурсам ПК и лучшая производительность, чем у коммерческого ПО. Таким образом можно ре-

комендовать применение рассмотренного СПО в учебном процессе на приборостроительных специальностях, разработчикам-фрилансерам и на небольших предприятиях.

Литература

[1] Сайт проекта Ngspice `http://ngspice.org/`

[2] Сайт проекта XYCE `https://xyce.sandia.gov/`

[3] Сайт проекта TkGate `http://tkgate.org/`

[4] Сайт проекта Qucs. `http://qucs.sf.net/`

[5] Сайт проекта openEMS `http://openems.de/`

[6] Brinson M. E., Jahn S. Qucs: A GPL software package for circuit simulation, compact device modelling and circuit macromodelling from DC to RF and beyond // International Journal of Numerical Modelling (IJNM): Electronic Networks, Devices and Fields. 2008. — September. Vol. 22, no. 4. Pp. 297 – 319.

[7] Кузнецов В. В. Симулятор электронных схем с открытым исходным кодом Qucs: основные возможности и основы моделирования, Компоненты и технологии, 2015, №3 (164)., с. 114 – 120

[8] Кузнецов В.В. Моделирование высокочастотных схем в частотной области при помощи САПР Qucs, Компоненты и технологии, 2015, №8 (169)., с. 120 – 127 `http://www3.interscience.wiley.com/journal/121397825/abstract`

[9] Brinson M., Kuznetsov V. Qucs equation-defined and Verilog-A RF device models for harmonic balance circuit simulation, in Mixed Design of Integrated Circuits & Systems (MIXDES), 2015 22nd International Conference, vol., no., pp.192-197, 25-27 June 2015

[10] M. Brinson, R. Crozier, V. Kuznetsov, C. Novak, B. Roucaries, F. Schreuder, G. B. Torri. Qucs: improvements and new directions in the GPL compact device modelling and circuit simulation tool., MOS-AK Workshop, Grenoble, March, 2015 `http://www.mos-ak.org/grenoble_2015/presentations/T4_Brinson_MOS-AK_Grenoble_2015.pdf`

[11] M. Brinson, R. Crozier, V. Kuznetsov, C. Novak, B. Roucaries, F. Schreuder, G. B. Torri. Qucs: An introduction to the new simulation and compact device modelling features implemented in release 0.0.19/0.0.19Src2 of the popular GPL circuit simulator., MOS-AK Workshop, Graz, September, 2015 `http://www.mos-ak.org/graz_2015/presentations/T_5_Brinson_MOS-AK_Graz_2015.pdf`

Константин Рыбас, Дмитрий Солдатенков, Александр Епифанов
Москва, TAU Technologies
Проект: TAU Platform `http://tauplatform.com/`

Проект TAU Platform. Кросс-платформенная разработка мобильных приложений

Для мобильных/встроенных устройств имеется набор различных платформ(операционных систем) с разной спецификой — это ощутимая проблема. Чтобы приложения работали на всех основных платформах приходиться нанимать специалистов с совершенно разными навыками.

Так например в зависимости от задач может понадобиться:

- набрать программистов под разные OS (iOS, Android, Windows Phone, Blackberry, Windows Embedded, Linux Embedded)
- создать проектную команду с синхронизацией разработки и тестирования под разные операционные системы

Открытый программный продукт TAU Platform[1] является решением для кросс-платформенной разработки мобильных/встроенных, а также настольных приложений, то есть позволяет написать код один раз, который будет работать везде.

Решение TAU Platform это «гибридное» решение. Исходный код приложения представляет из себя набор HTML/CSS/JS файлов + кода на языке Ruby (опционально). Приложение при запуске открывает WebView (возможно несколько в отдельных Tab-ах), в котором отображается контент.

При использовании Ruby, архитектура похожа на решение Ruby on Rails, только при этом веб сервер с виртуальной Ruby машиной запущен прямо на устройстве.

Как из JavaScript, так и из Ruby доступны различные API для доступа к функционалу устройства — файловой системе, камере, системным функциям, данным (календарь, контакты) и т.п. Также имеется набор модулей для доступа к различным capabilities, например

[1] `http://tauplatform.com/`

распознаванию штрих-кодов или работе с картами. Разработчики могут легко дополнять функционал новыми модулями с помощью соответствующего инструментария.

Вызовы различного API из WebView происходят путем отправки запросов локальному HTTP серверу, который в свою очередь в случае использования Ruby-кода обращается к виртуальной Ruby машине, а в случае использования JavaScript непосредственно к реализации соответствующего API через внутренние врапперы.

На данный момент мы ведем работу по кастомизации свободного движка для отображения веб-страниц WebKit, что даст продукту неоспоримые преимущества в части полного контроля над отображением контента.

Также в ближайшее время в решение будет добавлена виртуальная JavaSciprt машина, что даст возможность вынести бизнес-логику, написанную на JavaScript из WebView и поместить в отдельные контроллеры в отдельном потоке. Также это даст возможность комфортно отлаживать JavaScript код из IDE.

Если говорить о такой актуальной теме, как синхронизация данных между мобильными устройствами и хранилищами данных, то в TAU Platform включен клиентский модуль, позволяющий синхронизировать данные через специальный интеграционный сервер.

Также наша команда планирует работу по созданию облачных сервисов. Облачные сервисы подразумевают под собой такие возможности, как сборка, тестирование и дистрибуция приложений в облаке, а также разворачивание интеграционных серверов в облаке для синхронизации данных с такими бэкендами, как например SAP, Oracle, SalesForce,1C, Парус и любыми другими. Также возможна интеграция с IoT сервисами, например с Zatar(zatar.com). В этом направлении возможно наше партнерство с компанией IBM, от который мы получили соответствующее предложение.