alt linux
ALT LINUX TEAM

20-DEVCONF

XX

Конференция разработчиков свободного программного обеспечения

4 – 6 октября 2024, Переславль-Залесский

ТЕЗИСЫ ДОКЛАДОВ

Научное издание

Организаторы конференции

base alt

ООО «Базальт СПО»
Институт Программных Систем РАН

Двадцатая конференция разработчиков свободных программ

Переславль-Залесский, 04–06 октября 2024 года

Сборник материалов конференции
Научное издание

Москва,
МАКС Пресс,
2024

УДК 004.91:378
ББК 32.97:74.48
Д22

https://elibrary.ru/lvruju

Д22 **Двадцатая конференция разработчиков свободных программ** : материалы конференции / Переславль-Залесский, 04–06 октября 2024 г. / отв. ред. Чёрный В. Л. — М.: МАКС Пресс, 2024. — 112 с.
ISBN 978-5-317-07257-5
DOI: https://doi.org/10.29003/m4193.978-5-317-07257-5
EDN: https://elibrary.ru/lvruju

В книге собраны материалы конференции, одобренные Программным комитетом Двадцатой конференции разработчиков свободных программ.
Ключевые слова: программное обеспечение, Open Source, свободное программное обеспечение, Linux, разработка.

УДК 004.91:378
ББК 32.97:74.48

The Twentieth Free Software Developers Conference : conference materials / Pereslavl-Zalessky, October 04–06, 2024. / ex. ed. Cherny V. L. — M.: MAKS Press, 2024. — 112 p.
ISBN 978-5-317-07257-5
DOI: https://doi.org/10.29003/m4193.978-5-317-07257-5
EDN: https://elibrary.ru/lvruju
The book contains conference proceedings approved by the Program Committee of the Twentieth Free Software Developers Conference.
Keywords: software, Open source, free software, Linux, development.

ISBN 978-5-317-07257-5

Программа конференции

04 октября, пятница

12.00-15.00 Заселение, обед, регистрация участников

14:30 Автобус от гостиницы «Переславль»

Дневное заседание
15.00–19.00

15.00 Приветственное слово организаторов

05 октября, суббота

9:30 Автобус из гостиницы «Переславль»

Утреннее заседание
10.00–13.00

05 октября, суббота

Вечернее заседание
15.00–19.10

19:10 Автобус в гостиницу «Переславль»

06 октября, воскресенье

Утреннее заседание
10.00–13.30

09:30 Автобус от гостиницы «Переславль»

10.00–10.25 А. А. Андреев

Поддержка целевых платформ в фреймворке Qt на примере ОС Аврора 85

10.25–10.50 Д. Лапшин

Картографический инструментарий MFW 89

10.50–11.15 Д. М. Султаниязов

Прогрессивные веб-приложения: конвергентное дополнение нативным программам в операционных системах семейства Linux 92

11.15–11.35 Кофе-пауза

11.35–12.00 А. А Савченко

ALT Mobile 95

12.00–12.25 В. А. Соколов

Использование ALT Mobile в доменной инфраструктуре ... 98

12.25–12.50 А. А. Быстров

ALT Mobile на игровых консолях Anbernic: зачем и куда? .. 102

13:40 Автобус в Москву

Вне программы

Д. А. Костюк и др.

Разработка мультикомпонентной масштабируемой ГИС на базе виртуализации 104

Анатолий DOC Якушин
Москва, свободный аналитик

Свободное программное обеспечение в здравоохранении. Сегодняшнее состояние и перспективы

Аннотация

Сегодня здравоохранение остаётся одной из немногих областей, где опыт применения СПО крайне незначителен. Это объясняется целым рядом факторов, одним из которых является неправильное представление о роли и месте информационных технологий в современной медицине. Однако возможности сегодняшнего СПО позволяют обеспечить потребности здравоохранения во многих областях и существенно улучшить качество оказания медицинской помощи.

Ключевые слова: *СПО, здравоохранение, медицина.*

Начиная рассматривать использование СПО в современной медицине вообще и в здравоохранении в частности, следует в первую очередь представлять себе, что данное направление деятельности является весьма обширным и крайне неравнозначным в плане потребностей информатизации и требований, предъявляемых к аппаратно-программному обеспечению. В некоторых отраслях медицины большинство задач решаются стандартными базовыми средствами, другие же требуют весьма специализированного обеспечения, разработка которого в силу целого ряда причин практически недоступна свободному разработчику. Однако можно отметить и общие проблемы, снижающие интерес сообщества к деятельности в области здравоохранения.

К ним можно отнести:

Сакрализацию медицины, как области человеческого знания. Принято считать, что медицинская деятельность требует особого подхода, недоступного простому неподготовленному индивиду, что в совокупности со специальной терминологией, устоявшимися «цеховыми» представлениями и правилами делает планку входа в эту область крайне высокой.

Крайне низкую информационную грамотность медицинских работников. Как правило, знания в области вычислительной техники не

выходят за рамки владения на уровне простого пользователя установленным на рабочих местах специализированным программным обеспечением, и крайне поверхностными умениями работы с офисным пакетом, реже с электронной почтой и мессенджерами. И этой грамотности просто неоткуда взятся. За последние 15–20 лет количество русскоязычных руководств по данной тематике не превышает и десятка, при весьма посредственном их качестве.

При этом анализ возможностей свободного программного обеспечения показывает, что при выполнении ряда условий СПО может выступать в здравоохранении в качестве полноценной платформы, и это подтверждается реализацией целого ряда масштабных проектов в нескольких российских регионах, где за последние годы внедрены региональные информационные системы на платформе свободного ПО.

Для решения повседневных рабочих задач (ведение документации, анализ и визуализация данных, планирование деятельности подразделения и т. п.) вполне возможно использование стандартных свободных программ, в том числе и кроссплатформенных, при наличии соответствующих навыков.

Ещё одной областью медицины, где использование СПО, возможно уже сегодня, является визуализация изображений в рентгенологии, травматологии, ангиохирургии. Открытый стандарт графических изображений `DICOM` реализован в нескольких десятках свободных программых продуктов, не уступающих, а зачастую превосходящих по своим возможностям проприетарные аналоги.

Дальнейшее развитие использования СПО в здравоохранении несомненно будет проводится по ряду направлений:

Продолжением работ по более широкому использованию свободных программных платформ и как следствие этого, побуждение разработчиков специализированного ПО к созданию кросплатформенных и/или web решений.

Создание руководств/учебных курсов по использованию СПО для решения типовых задач здравоохранения и адаптация имеющихся программных средств для этого.

Разработка специализированного программного обеспечения под свободными лицензиями в тех редких случаях, когда применение стандартных программных средств не приносит соответствующих результатов.

Владимир Харин
Москва, Университет им. О. Е. Кутафина (МГЮА)

Защита лицензий на свободное программное обеспечение

Аннотация

В статье рассматриваются вопросы защиты лицензий на свободное программное обеспечение: от чего необходимо защищать, какими способами можно защитить, как будет происходить доказывание нарушений, кто будет следить за нарушением лицензий. Автор оценивает, насколько применимы юрисдикционные и неюрисдикционные способы к защите свободных лицензий. Автор предполагает, что техническая и организационная самозащита поможет увеличить количество примеров успешной защиты свободных лицензий.

Ключевые слова: *свободные лицензии, правовая защита СПО, плагиат открытого исходного кода программ.*

Свободное программное обеспечение (далее также — свободное ПО, СПО) — это программное обеспечение, распространяемое на условиях свободного лицензионного договора, на основании которого пользователь получает право использовать программу в любых не запрещённых законом целях; получать доступ к исходным текстам (кодам) программы как в целях её изучения и адаптации, так и в целях переработки, распространять программу (бесплатно или за плату, по своему усмотрению), вносить изменения в программу (перерабатывать) и распространять экземпляры изменённой (переработанной) программы с учётом возможных требований наследования лицензии [1].

Суд по интеллектуальным правам указывал, что «предоставляя свою программу для ЭВМ обществу на условиях открытой лицензии, автор распоряжается своим исключительным правом на нее» (Постановление СИП от 31.05.2019 по делу № А40-151454/2017 [3]. *Важная оговорка:* законодатель отождествляет понятия «открытая лицензия» и «свободная лицензия». Исходя из указанного вывода Суда по интеллектуальным правам, можно сказать, что правообладателями свободных программ выступают их авторы, а также авторы внесённых в них изменений.

Несмотря на достаточно большую историю развития свободного ПО в России, в стране почти не существует примеров защиты прав

на свободные программы, что создаёт ощущение безнаказанности, и, как следствие, соблазн нарушать свободные лицензии. Далее мы рассмотрим вопросы:

- От чего защищать права на свободные программы? Почему это важно?
- Как защитить свои права? Как доказывать факт нарушения?
- Кто будет следить за нарушением лицензий?

Какие могут быть нарушения прав на свободные программы (от чего будем защищаться)

Суть лицензий на свободное ПО заключается в четырёх свободах: использования, адаптации, распространения исходной версии программы, а также распространения улучшенной. Все эти свободы являются неотъемлемыми частями свободной лицензии, причём два из этих критериев (свободы адаптации и распространения улучшенной версии программы) предполагают открытие исходного кода.

Нарушения свободных лицензий, на условиях которых распространяется программное обеспечение, и, собственно, исключительного права на него, происходят несколькими способами: плагиат исходного кода, его перелицензирование, нарушение личных прав автора, а также нарушение условий об оплате при использовании открытого исходного кода в проприетарных программах.

а) Плагиат исходного кода свободной программы

В силу статьи 10 Соглашения по торговым аспектам прав интеллектуальной собственности, а также пункта 1 статьи 1259 Гражданского кодекса РФ (далее — ГК РФ) программы для ЭВМ охраняются как литературные произведения.

Плагиат программы заключается в присвоении авторства на чужую такую работу или её часть. Плагиат свободной программы есть нечто большее: это не просто присвоение авторства себе, но и «закрытие» исходного кода, т. е. разработка на основе открытого исходного кода проприетарной программы, т. е. той, у которой недоступен исходный код для всеобщего обзора. Важно помнить, что не все свободные лицензии позволяют закрывать код (например, GNU General Public License). Таким образом, если лицензия свободной программы

не позволяет использовать находящийся в ней код в проприетарных программах, однако он в ней используется, то это считается плагиатом.

В 2022 году компания Synopsys провела исследование, которое показало, что 98% всех компьютерных программ в мире содержат в себе открытый код, также такие программы состоят в среднем на 75% из открытого кода [2] Несмотря на то, что исследование проводилось по всему миру, статистика по России если и другая, то несильно. Так как процент высокий, есть большая вероятность, что часть открытого кода включена в проприетарные проекты с нарушением лицензий. Пока Вы улучшаете свой проект, кто-то может пользоваться Вашими наработками. В связи с этим существует необходимость защиты свободных программ от плагиата. Из-за сложности обнаружения факта нарушений свободных лицензий, а также отсутствия регистрации таких нарушений, существует вероятность, что масштаб проблемы большой, а мы о нём не знаем.

б) Перелицензирование свободных программ

Перелицензирование заключается в копировании какого-то количества исходного кода в другие свободные проекты, которые распространяются под другими лицензиями, несовместимыми с лицензией копируемого кода. Это тоже считается нарушением условий лицензий, причём обеих.

Статистика по неправомерному перелицензированию программ отсутствует, однако есть несколько примеров несовместимости свободных лицензий между собой. Так, Apache 2.0 несовместима с GPL v2, GPL v2 несовместима с GPL v3 и т. д.

в) Нарушение условий об оплате при использовании открытого исходного кода в проприетарных программах

Существуют проекты, лицензии которых разрешают использовать их код, расположенный в свободном доступе, в проприетарных программах, при внесении оплаты. При этом для свободных программ использование кода бесплатно. Примером такого проекта служит популярная библиотека `Qt`. Существует техническая возможность обращаться к такой библиотеке бесплатно в проприетарных программах. Таким образом, существует соблазн нарушения такой лицензии.

К сожалению, к указанным нарушениям привыкли подходить «спустя рукава», поэтому вопрос о защите прав на свободные программы, а также свободных лицензий развивается, на наш взгляд, недостаточно сильно.

Как защитить свои права?

Способы защиты нарушенных прав делятся на юрисдикционные и неюрисдикционные.

1. Юрисдикционные — это обращение в уполномоченные государственные органы (судебной и исполнительной власти) по защите нарушенных прав.

а) **Обращение в суды** за защитой своих прав представляется необходимым по следующим причинам. Во-первых, это огласка: можно требовать публикацию решения суда за счёт ответчика. Во-вторых, это понуждение к открытию закрытого кода или запрет на использование проприетарной программы. И в-третьих, также можно взыскать компенсацию в пользу автора в размере от десяти тысяч до пяти миллионов рублей (пп. 1 ст. 1301 ГК РФ).

Обоснование факта нарушения свободных лицензий рассмотрим на примере доказывания наличия плагиата. Так, доказать заимствование открытого кода в проприетарных программах можно путём судебной экспертизы. Для этого необходимо при подаче искового заявления или в ходе судебного процесса попросить суд о её назначении, т. е. заявить ходатайство, после чего суд его рассмотрит и назначит экспертизу, которая определит наличие или отсутствие заимствования.

б) **Защита через органы исполнительной власти** видится более перспективной ввиду более широкого инструментария борьбы с нарушителями. Так, методы административной борьбы с правонарушениями в сфере СПО могут быть следующими:

- Пресечение и устранение правонарушений правоохранительными и контролирующими органами;
- Принудительная регистрация всех программ; проверка исходного кода при регистрации — так, по мнению некоторых учёных, может быть обеспечен контроль за плагиатом. На наш взгляд, это трудноосуществимо: этот процесс ресурсозатратен и увели-

чивает нагрузку на государственные органы без видимой пользы для государства;

- Создание органа или подразделения Минцифры РФ, который будет осуществлять надзор в сфере свободного программного обеспечения. Это менее утопичный процесс, но автор считает, что негосударственная организация с этим справится лучше. Об этом будет сказано позже;
- Проведение государством мероприятий по повышению уровня правосознания граждан в рамках использования свободного ПО. Представляется, что эта деятельность будет проводиться эффективнее различными некоммерческими организациями.

Таким образом, из перечисленных методов представляется необходимым лишь защита через полицию, прокуратуру и иные органы, которые могут осуществлять проверку факта нарушений, привлекать к административной и (или) уголовной ответственности, обязывать субъектов устранить совершённые ими правонарушения.

2. Неюрисдикционные — это действия физических и юридических лиц по самостоятельной защите прав без обращения к государственным органам.

Самозащита в нашем случае может быть технической и организационной.

а) **Техническая защита** от плагиата может заключаться в преднамеренно нелогичном кодировании отдельных фрагментов, чтобы такой же код невозможно было написать при самостоятельной разработке другой подобной программы. Важно, чтобы эти особенности кодирования существенным образом не ухудшали качество программы, чтобы не возникала необходимость переписать эти фрагменты.

б) **Организационная защита** заключается в объединении лиц в ассоциацию, которая будет отслеживать нарушения, защищать их права, консультировать по правовым вопросам.

Кто будет следить за нарушением лицензий?

Следить за нарушением прав на свободные программы важно. Безусловно, можно обнаружить нарушения самостоятельно (и здесь главным образом поможет именно техническая защита кода от неправомерного копирования). Однако далеко не всегда можно с лёгкостью обнаружить, а тем более пресечь и устранить нарушения свободных

лицензий. И здесь представляется необходимым создание организации, которая бы осуществляла деятельность по наблюдению и пресечению нарушений прав на свободные программы.

Возможны два варианта организации надзора:

а) государственный контроль и надзор за нарушением прав — для этого необходимо создание органа или подразделения какого-либо органа государственной власти, функции которого будут составлять меры надзора в сфере СПО;

б) негосударственная организация — объединение граждан и организаций, занимающихся разработкой СПО. Представляется, что это объединение будет действовать как организация по коллективному управлению правами: выявлять нарушения, защищать правообладателей, взаимодействовать с государственными органами, в том числе подавать заявления в суд, консультировать по правовым и иным вопросам. Представляется также, что такая организация будет помогать собственным участникам, а также другим лицам регистрировать программное обеспечение и базы данных в Роспатенте, а также в реестре отечественного ПО.

Мы предполагаем, что второй вариант более удобен и эффективен для защиты свободных программ и их авторов. Так, негосударственная организация способна оперативнее помогать гражданам и юридическим лицам в силу более доступного документооборота: например, общение возможно через мессенджеры. Также такая организация способна консультировать не только по юридическим, но и по различным техническим вопросам. Таким образом, объединение способно стать аналогом Фонда Свободного Программного Обеспечения (Free Software Foundation).

Подводя итог сказанному выше, можно сказать, что проблема нарушений свободных лицензий, а вместе с тем исключительного права на свободные программы существует, а потому защищать правообладателей необходимо. Представляется, что наибольшую роль в защите прав на свободное ПО может сыграть как самостоятельная техническая защита разработчиков — авторов исходного кода программ, так и объединение физических и юридических лиц в крупную негосударственную организацию, которая помимо защиты прав может самостоятельно выявлять нарушения, консультировать своих участников по юридическим и иным вопросам. Техническая и организационная самозащита поможет увеличить регистрацию правонарушений и повли-

ять на статистику защиты свободных лицензий, увеличив количество успешных примеров борьбы с правонарушениями.

Литература

[1] Свободное программное обеспечение в госорганах / Министерство цифрового развития связи и массовых коммуникаций Российской Федерации. — [Электронный ресурс]. — URL: `https://digital.gov.ru/ru/activity/directions/106/`

[2] Open Source Security and Analysis Report / Synopsis. — [Электронный ресурс]. — URL: `https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html`

[3] Постановление Суда по интеллектуальным правам от 31.05.2019 по делу № А40-151454/2017 / Система ГАРАНТ.

Николай Непейвода
Переславль-Залесский, ИПС РАН

Условные операторы: чуть теории для реализаторов

> *Огурец, помещённый в правильный рассол, хочешь не хочешь, станет солёным.*
> Бакланов ПГНИУ
>
> *Ничего, через сто лет, может быть, поймёте.*
> Алгебраист, читающий теорию категорий информатикам.

Напомним некоторые принципы понимания практиком теоретика [1].

1. Если теоретик говорит, что это в принципе возможно, понимайте: это практически невозможно, нужно уточнить задачу.
2. Если теоретик говорит, что это не имеет значения, понимайте: это имеет первостепенное значение для реализации.
3. Если вы считаете, что теоретики занимаются сферическими конями в вакууме, говорить с ними бесполезно. Если вы осознали, что их мир другой, но имеет множество выходов на практику, надейтесь на успех.

В своей семье, где знают и теорию, и практику, мы порою называем теоретиков и практиков разными кастами. И у тех, и у других есть свой тайный язык, а общаться они могут лишь на общем, из-за этого понимают друг друга плохо.

Приведём два простейших примера перевода с теоретического на практический. Когда теоретик говорит, что операция коммутативна и ассоциативна (линейная логика Жирара [2]), то практик должен насторожиться и понять, что у него есть прекрасные возможности распараллеливания, организации распределённого поиска и оптимизации. Ведь коммутативность и ассоциативность означает, что отдельные действия независимы, и их можно упорядочить или выполнять как угодно. Если же есть коммутативность без ассоциативности, то возникает неупорядоченное дерево, в котором можно так же распределить или оптимизировать движение, или же наоборот, быстро не задумываясь написать главный цикл или рекурсию, и заняться более интересными вещами.

Заметим, что всё время здесь появляется возможность распределённых вычислений, а свободные распределённые вычисления — это одно из направлений, где свободное программное обеспечение необходимо для слаженной работы волонтёров и решения трудных либо неразрешимых задач.

Здесь мы мельком коснулись ещё одного принципа работы с теоретиками.

4. Если теоретик говорит, что задача неразрешима или доказуемо трудна, то либо ограничьте аппетиты, выделив её часть, которая действительно нужна, либо поинтересуйтесь насчёт алгоритмов приближённого или вероятностного неточного решения.

В случае ограничения наилучшее разобраться с теоретиком, какие же подзадачи можно *в принципе* решить. Например, задача верификации программы доказуемо не проще, чем задача построения её с самого начала, что практически означает, что текст уже существующей программы будет чаще всего лишь мешать. Но задача верификации по типам данных или по областям изменения вполне решаемы и здесь текст программы помогает. Так что приходим к выводу Капитана Очевидность: проверить по настоящему можно лишь программу с открытым кодом, после чего ещё раз обращаемся к логикам и выясняем, что текста программы недостаточно в общем случае, нужно ещё иметь открытую спецификацию. И если это всё есть, практик

может трудиться с большой надеждой на успех либо на обнаружение ловушек. Заметим, что в случае общей задачи верификации открытая спецификация помогает слабо (зато уж точно не мешает!), но зато она помогает аккуратно пересоздать подозрительную программу с самого начала.

А приближённый алгоритм часто прекрасно ложится на распределённые вычисления. Заметим только, что, если вы реализовали недетерминированность либо неточность псевдослучайным алгоритмом, то на самом деле это детерминированный алгоритм, и вы никакого выигрыша не получите.

Теперь рассмотрим один частный случай взаимодействия теории и практики: условные операторы.

Адекватный анализ условных вычислений требует обработки ошибок [3]. Поэтому логика здесь четырёхзначная. T и F нормальные значения, минимально требуемые для условных вычислений. U ошибка, которую можно попытаться обработать и продолжить вычисления, $\perp$ крах.

Традиционный условный оператор **if** A **then** M **else** N **fi**, как показал Маккарти, является базисом алгоритмических вычислений над произвольными абстрактными типами данных. Рекурсивные схемы, состоящие из нескольких определений вида

$$f_i(\vec{x}) \leftarrow \textbf{if } A(\vec{x}) \textbf{ then } M(\vec{f},\vec{x}) \textbf{ else } N(\vec{f},\vec{x}) \textbf{ fi}$$

дают одно из самых удобных, мощных и абстрактных понятий вычислимости. Они создавались для случая последовательных вычислений, где единственный способ преодолеть ошибку — защититься от её появления соответствующим условием.

Дейкстра заметил, что, если задана спецификация программы, то более адекватен условный оператора вида[1]

$$\textbf{if } A_1 \to S_1 \ \square \ \cdots \ \square \ A_n \to S_n \ \textbf{fi},$$

выполняемый недетерминированно, поскольку в случае истинности нескольких условий A_i нам безразлично, какую из возможностей выбрать. Но при реализации данный оператор обычно детерминируют, что дополнительно затрудняет верификацию, проверку и особенно модификацию программ: почему это A_i стоит впереди A_j? Зато на

[1] В оригинальных обозначениях Дейкстра.

распределённые вычисления дейкстровский оператор ложится прекрасно: задать волонтёрам свои $A_i \to S_i$ и организовать «соревнование»: кто первым найдёт и вычислит удачный вариант?

Ещё одно очевидное и используемое, начиная с Фортрана, расширение условного оператора, когда управляющих значений не два $\{T, F\}$, а несколько. Такие переключатели легко обобщаются до дейкстровских и поддаются соревновательным вычислениям.

Логика обычного оператора **if** также была описана Маккарти, и её называют логикой Лиспа. Она по умолчанию используется в большинстве реализаций логических связок. Поэтому $a[i] > 0$ & $i < 100$ потенциальная ошибка, а $i < 100$ & $a[i] > 0$ правильное выражение.

Группа Поттосина в ВЦ СО АН СССР заметила, что при параллельных вычислениях можно интерпретировать условный оператор по-другому. Вычисляем условие и обе альтернативы параллельно. Если условие никак не вычислится до конца либо даёт исправимую ошибку, но обе альтернативы дали один и тот же результат, то его и выдаём. С логической точки зрения это решение красивое и даёт логику Лукасевича, но практик здесь должен задуматься и скорее всего отвергнуть. Во-первых, это решение уменьшает шансы обойти ошибку, так как вычисляются все три части условного оператора. Во-вторых, оно плохо масштабируется на дейкстровский **if** и переключатели.

Если чисто логически попытаться развить новосибирскую идею, то приходим к максимальному **if**. Если условие не вычисляется или даёт устранимую ошибку, и все альтернативы дают одно и то же значение либо устранимую ошибку, то принимаем выдаваемое значение. Этот вариант использован при управлении вычислениями по переполнениям, но он так же плохо обобщается на случаи дейкстровского оператора, переключателей и распределённых вычислений. Зато с логической точки зрения он даёт логику Розоноэра и функционально полон.

Ещё один вариант обработки ошибок — управление по неудачам, как в Прологе. Если условие дало ошибку, выбираем первую из альтернатив, давшую определённый результат. Этот вариант ориентирован на последовательное исполнение и также функционально полон с логической точки зрения. Он отлично обобщается на переключатели. Дейкстровский вариант такого оператора соответствует стандартной реализации **case** в языках, начиная с Лисп.

Таким образом, теоретический анализ в данном случае приводит к двум выводам. Во-первых, подтверждается целесообразность реализации стандартного либо прологовского варианта обработки условий. Во-вторых, в случае стандартного варианта традиционная реализация **case** является одной из *подпорок*, вставляемых для согласования с условностями реализации и затрудняющих анализ и модификацию программы.

Но саркастическая ошибка теоретического анализа состоит в том, что научной истины нет. Есть лишь условный вывод, базирующийся на массе допущений, многие из которых партнёры не высказывают вслух, причём умолчания разные, и поэтому они особенно коварны. В частности, если перейти к рекурсивным схемам, то здесь наиболее мощный результат, а также наибольшие возможности оптимизации, проверки и модификации даёт логически полный третий **if**. Прологовский формальную мощь даёт ту же, но возможности оптимизации и модификации хуже.

И, наконец, заметим, что непосредственно реализовать недетерминированность трудно, если вычисления не распределённые. Но практик может адекватно трактовать недетерминированность как разрешение детерминировать данное место любым способом, и использовать недетерминированный вариант как спецификацию либо даже разрешить его в языке с указанием, что транслятор имеет право детерминировать данный фрагмент любым способом, и оставить на усмотрение оптимизатора. На самом деле в забытом ныне языке Алгол-68 [4] было введено понятие совместного исполнения, когда операторы разделялись запятыми, что означало позволение транслятору вычислять их последовательно в любом порядке либо даже параллельно.

Литература

[1] *Непейвода Н. Н.* Математик и прикладник: о взаимо(не)понимании Вестник Удмуртского университета, 2007, №1, стр. 251–268. EDN: `https://www.elibrary.ru/kfaatp`

[2] *Emmanuel Beffara.* Introduction to linear logic. Master. Italy. 2013. cel-01144229

[3] *Непейвода Н. Н.* Логики условных вычислений с ошибками. // Принята в журнал «Логические исследования» 2024.

[4] *Пейган Ф. Дж.* Практическое руководство по Алголу 68 = A Practical Guide to ALGOL 68 / пер. с англ. А. Ф. Рара, под ред. А. П. Ершова. — М.: Мир, 1979. — 240 с.

Валерий Баканов
Москва, НИУ ВШЭ
http://vbakanov.ru/dataflow/, http://vbakanov.ru/spf@home/

Построение планов параллельного выполнения программ для процессоров со сверхдлинным машинным словом

Аннотация

Обсуждается метод получения планов (расписаний) параллельного выполнения программ для процессоров со сверхдлинным машинным словом, основанный на информационном анализе конкретных алгоритмов. Используется естественный для данной архитектуры метод получения Ярусно-Параллельных Форм (ЯПФ) для Информационного Графа Алгоритма (ИГА) с последующими целенаправленными эквивалентными (не изменяющими информационных связей) преобразованиями к заданным параметрам вычислительной системы. Преобразования выполняются с использованием эвристических методов, которые описываются на встроенном скриптовом языке (применена виртуальная Lua-машина); это обусловливается требованием большой гибкости при поиске рациональных (близких к оптимальным) параметров процедур собственно преобразований. Основными целевыми критериями являлись получение максимальной плотности кода (усреднённый процент использования слотов сверхдлинного слова) и вычислительная трудоёмкость получения данного плана (расписания) параллельного выполнения. Дополнительно анализировались размер и время существования внутренних (локальных) данных и оценивались операторы-претенденты для помещения в кэш процессора. В ходе исследований найдено интересное явление, заключающее в выгодности (в смысле минимизации вычислительной трудоёмкости преобразований) использования ЯПФ в начальной т.н. «нижней» форме (когда все операторы находятся на наиболее приближённых к концу выполнения программы ярусах ЯПФ).

Ключевые слова: *анализ тонкой информационной структуры алгоритмов, целенаправленные эквивалентные преобразования алгоритмов, плотность кода и вычислительная трудоёмкость преобразований, вычислительные системы VLIW-архитектуры.*

Процессоры архитектуры сверхдлинного машинного слова (VLIW — *Very Long Instruction Word*) относятся к специфическим классам архитектур, изначально нацеленным на использование внутреннего параллелизма в алгоритмах (программах), причём параллелизм этот анализируется и планируется к рациональному использованию программными методами (идеологема EPIC — *Explicitly Parallel Instruction Computing*, явный параллелизм уровня машинных команд). При этом собственно аппаратная часть процессора освобождается от процедур распараллеливания (и поэтому потенциально должна стать проще и экономичнее применяющих внутреннее распараллеливание), а степень реализации потенциала параллелизма благодаря использованию (более изобильных) программных ресурсов должна повыситься.

Несмотря на выпуклое преимущество (программным путём дешевле реализовать сложные процедуры параллелизации), работа VLIW-процессоров сопряжена с известными проблемами. Т.к. аппаратно во VLIW-архитектуре уже заложена линейка отдельных вычислительных ядер соответственно каждому слоту сверхдлинного слова, важным требованием является задача максимальной загрузки вычислительной аппаратуры (этих ядер). Есть данные, что одной из серьёзных проблем Itanium'а явилась как раз неприемлемо малая загрузка аппаратуры на некоторых типах алгоритмов.

В данной работе главенствующим принято направления изучения свойств алгоритмов [1] и исследуются пределы использования свойства внутреннего (скрытого) их параллелизма, которые реально получить в ходе разработки планов параллельного выполнения алгоритмов (программ) согласно предлагаемой методике.

Исследования развивают подход акад. Вал. Воеводина и Г. Поспелова, предполагающий получение «квинтэссэции» (*экстракта*) программы в форме её **алгоритма** для анализа информационных (фактически причинно-следственных) связей; отстранение от несущественных характеристик позволяет получать быстрый ответ на вопросы о важнейших количественных показателях программы и производить целенаправленные эквивалентные (не изменяющие информационных связей между действиями) преобразования в направлении

согласования с параметрами архитектуры конкретных вычислительных систем (напр., количеством вычислителей в параллельном вычислительном поле), [2].

Исследовательским инструментом являлась программная система `Практикум DF-SPF` [3], специально созданная для данных целей. Программные модули разработаны с использованием языка C/C++ в стиле GUI для модели Win'32, являются полностью OpenSource [4] и могут быть выгружены для свободного использования (формат инсталляционных файлов)[1]. Вычислительные эксперименты проводились над набором оформленных в виде библиотеки программ (алгоритмов), реализующих наиболее часто применяющиеся (напр., линейной алгебры); библиотека может неограниченно расширяться усилиями участников исследований. Условность выполнения операторов реализуется *предикатным* методом (что позволяет избежать мультивариантности ЯПФ), программные циклы перед выполнением разворачивались («*unrolling*») с использованием макросов.

Естественным является метод представления каждого слота сверхдлинного командного слова соответствующим сечением ИГА с выполнением последовательно по ярусам с начала до конца программы. Для получения рациональных (разумных, стремящихся к оптимальным вследствие *NP*-полноты задачи, [5]) являются целенаправленные эквивалентные преобразования ЯПФ, описываемые с использованием скриптового языка Lua [6]. Основой создания этих сценариев является эвристический подход совместно с итерационным методом движения в сторону повышения качества разрабатываемых планов параллельного выполнения программ.

Усреднённая степень использования параллельных вычислительных ресурсов оценивалась *пространственной плотностью кода*. Для практического применения введена характеристика *плотность кода* (один из близких аналогов — *коэффициент полезного действия*), которую удобно оценивать *коэффициентом использования параллельных вычислителей* (считая общее число таких вычислителей равным ширине ЯПФ).

Основными характеристиками эффективности полученных планов считались следующие:

[1] `http://vbakanov.ru/dataflow/content/install_df.exe`, `http://vbakanov.ru/spf@home/content/install_spf.exe`

- *пространственная плотность кода* (уровень загрузки вычислительных ядер VLIW-процессора) — *коэффициент использования параллельных вычислителей*,
- *вычислительная сложность* целенаправленного преобразования ЯПФ — число элементарных шагов преобразования (в качестве меры выбрано *число перемещений операторов* с яруса на ярус ЯПФ).

Дополнительно анализировалась полученная в результате преобразования высота ЯПФ (*параллельная вычислительная сложность*) — оценка времени выполнения алгоритма (программы).

В ходе исследований были предложены две эвристики преобразования ЯПФ (*WithByWith* и *Dichotomy*), сильно различающиеся по указанным характеристикам. Показано, что для VLIW-процессоров с размером слота сверхдлинного слова в $4 \div 16$ команд плотность кода изменяется в широком диапазоне (может доходить до $0,6 \div 1,0$ с продолжающимся падением при дальнейшим росте слота) и повышается с увеличением размеров обрабатываемых данных.

В целом стартующие от 0,8 значения плотности кода нельзя признать полностью удовлетворительным (вследствие малости). «Светом в конце тоннеля» может являться:

- факт улучшения качества получаемых планов параллельного выполнения программ с возрастанием *размеров обрабатываемых данных* (с ростом порядка матриц резко возрастает число степеней свободы в расположении операторов по ярусам ЯПФ),
- улучшение качества эвристик (автор ни в коей степени не считает, что достигнут оптимум при движении в этом направлении).

Обладающее перспективой является замеченное явление разницы между «верхним» и «нижним» ЯПФ. Оба варианта получения ЯПФ (в первом случае все операторы максимально «прижаты» к началу программы, во втором — к концу оной; назовём их ЯПФ(в) и ЯПФ(н) соответственно) требуют одинаковой вычислительной сложности при получении, но параметры их сильно отличаются. Характер диаграмм интенсивности вычислений для ЯПФ(в) сохраняется; в случае использования ЯПФ(н) имеется явная тенденция к смещению максимума диаграмм ЯПФ к конечной части диаграммы (и собственно неравномерности распределения). А раз вычислительная трудоёмкость получения ЯПФ(в) и ЯПФ(н) одинакова, есть резон начинать процесс це-

ленаправленной модификации с состояния именно ЯПФ(н), при этом основным направлением переноса операторов будет «снизу вверх».

Программная система Практикум DF-SPF показала себя действенным исследовательским инструментом и разработчик заинтересован в всемерном дальнейшем её распространении.

Литература

[1] AlgoWiki. Открытая энциклопедия свойств алгоритмов. Под ред.: Воеводин В., Донгарра Дж. URL: http://algowiki-project.org (дата обращения: 10.02.2024).

[2] *Баканов В. М.* Практический анализ алгоритмов и эффективность параллельных вычислений. — М.: Пробел-2000, 2023. — 198 с. URL: https://www.litres.ru/book/v-m-bakanov/prakticheskiy-analiz-algoritmov-i-effektivnost-parallelnyh-vyc-70184365/

[3] *V. M. Bakanov*. Software complex for modeling and optimization of program implementation on parallel calculation systems. Open Computer Science, 2018, 8, Issue 1, Pages 228–234, DOI: https://www.degruyter.com/document/doi/10.1515/comp-2018-0019/html

[4] *Баканов В. М* Исходные коды проекта Практикум DF-SPF. URL: https://github.com/Valery-Bakanov/data-flow, https://github.com/Valery-Bakanov/spf-home

[5] *М. Гэри, Д. Джонсон*. Вычислительные машины и труднорешаемые задачи. — М.: Мир, 1982. — 416 с.

[6] *Иерузалимски Роберту*. Программирование на языке Lua. — М.: ДМК Пресс, 2014. — 382 с.

Михаил Шигорин
Москва, Базальт СПО

Проект: Разработка на платформе Эльбрус `http://dev.mcst.ru`

Патчи для «Эльбруса»

Аннотация

Патчи, или правки исходных текстов для собираемости и работоспособности на платформе «Эльбрус», успели обрасти целым рядом мифов, будучи под неразглашением. Поскольку летом 2024 года большая часть патчей была опубликована, разберём обстановку — и мифы.

Ключевые слова: *Эльбрус, e2k, исходные тексты, патчи, публикация.*

До 2014

Не погружаясь в историю семидесятых годов прошлого века и далее, начнём с века нынешнего. К его началу в МЦСТ применяли Sun Solaris, а эксперименты с Linux пошли в ранних нулевых — соответственно часть образовавшейся культуры явно идёт ещё от Sun с её смесью жёсткой проприетарности и более поздней поддержкой СПО. Практика работы с МО РФ также способствовала укоренению принципа «как бы чего не вышло», существенно препятствующего инициативной публикации несекретных наработок.

При этом минимальный патч на binutils был предложен в апстрим ещё в 2008 году.

2014—2024

С 2014 года наработки МЦСТ стали в принципе доступны на «гражданке», но лишь организациям: купить оборудование могли только юридические лица, а для получения документации и исходных текстов требовалось подписание соглашения о неразглашении. При этом стоит отметить, что с поставляемыми вычислительными комплексами «Эльбрус» в комплекте предоставлялись и ОС «Эльбрус» с компилятором lcc — для их получения, вопреки одному из мифов, подписание NDA не требовалось.

В это время патчи из МЦСТ стали попадать в более широкий круг апстримов: cmake, meson, flashrom, zstd... Но основной массив правок

оставался закрытым — стоит заметить, что за информацией по e2k и впрямь охотились на Западе; так, осенью 2019 года на IRC-канале #altlinux прозвучали невинные вопросы насчёт реверс-инжиниринга бинарника для e2k в рамках соревнований CTF с упоминанием имеющегося кроссового тулчейна.

При этом в мае 2021 года было опубликовано Руководство по эффективному программированию на платформе «Эльбрус» [4], частично документирующее систему команд.

В 2022 году произошёл ряд утечек исходного кода через нерадивых партнёров МЦСТ, одна из которых документирует систему команд полностью.

2024

3 июля 2024 года произошло событие, которому и посвящён этот доклад: компания МЦСТ объявила о состоявшейся публикации исходных текстов e2k-портов ядра Linux и системной библиотеки glibc, а также массива патчей для прикладных программ и библиотек — и запуске сайта Разработка на платформе Эльбрус [2].

Заметного влияния на сообщество это событие оказать ещё не успело, поскольку у партнёров исходные тексты и так в наличии, у сообщества — утечки; необходимо дальнейшее формирование инфраструктуры обмена текущими наработками.

Нужное

В дальнейшем, как упоминалось на этой конференции в 2023 году, потребуется отход от сложившейся практики, включая архивы вместо гитов и форки вместо соучастия — и переход к известной обкатанной технической инфраструктуре [13]:

1. репозитории разработки кода (git);
2. чат для быстрых обсуждений (telegram);
3. почтовые рассылки для обсуждения по существу (mailman);
4. отслеживание конкретных ошибок (bugzilla);
5. накопление и систематизация опыта (wiki).

для уменьшения затрат времени и сил на постоянно идущий процесс обмена исходными текстами при разработке силами МЦСТ и

партнёров, а также дальнейшего развития и более широкого вовлечения сообщества энтузиастов в деятельность по переносу, оптимизации, обновлению, исправлению и созданию программ.

Литература

[1] Открытие исходных текстов ПО для платформы Эльбрус, URL: http://mcst.ru/open-sourcing

[2] Разработка на платформе Эльбрус, URL: http://dev.mcst.ru/

[3] Wiki ALT Linux о МЦСТ, URL: http://altlinux.org/Эльбрус/upstream#МЦСТ

[4] Руководство по эффективному программированию на платформе «Эльбрус» URL: http://mcst.ru/elbrus_prog

[5] Эльбрус Бабаяна и Pentium Пентковского, URL: http://ixbt.com/cpu/e2k-spec.html

[6] Официальный сайт МЦСТ, URL: http://mcst.ru

[7] Официальный сайт ИНЭУМ, URL: http://ineum.ru

[8] Официальный сайт Новосибирского центра информационных технологий УНИПРО, URL: http://unipro.ru

[9] Исходные коды Эльбруса, URL: http://git.mentality.rip/OpenE2K

[10] Патчи Ильи Курдюкова для e2k, URL: http://github.com/ilyakurdyukov/e2k-ports

[11] Проекты Леонида Юрьева, URL: http://gitflic.ru/user/erthink

[12] Проект lccrt, URL: http://gitflic.ru/project/e2khome/lccrt

[13] Создание Свободного Программного Обеспечения, URL: http://producingoss.com/ru/technical-infrastructure.html

[14] Скрестим вилки, доклад Михаила Шигорина, на Application Developers Days 2011 г. http://0x1.tv/2a9-lets-cross-forks-shigorin

Даниил Исакевич
Владимир, ООО «БизнесСофтСервис»

Проект: metahim, drakoder `https://gitflic.ru/project/dvisa/drakoder`, `https://gitflic.ru/project/dvisa/metahim`

Использование Рефала-М для преобразования текстов на формальных языках

Аннотация

Показаны примеры использования Рефала-М [1] для преобразования текстов на формальных языках: автоматического построения ДРАКОНовских диаграмм (редактор `drakon`) из псевдокода и наборов формул предложенного автором языка в виде схем-графов (пакет `graphviz`). Уделено внимание организации программы и программирования, возникающим ситуациям и приёмам.

Ключевые слова: *Рефал, язык, преобразование.*

Введение

Рефал — язык, созданный для написания программ, пишущих программы. Его наличие поощряет создание языков под своё понимание и нужные классы задач. Часто перевод с одного формального языка высокого уровня на другой более низкого уровня осуществим достаточно быстро. Выходной язык низкого уровня может быть достаточной для решения задачи частью более богатого языка, сделанного кем-то другим. Среди востребованных задач такого рода — построение содержательной графики.

Укрощение drakon'а

Автоматизация формирования ДРАКОНовских диаграмм понадобилась из-за того, что для редактора `drakon` диаграмма — лишь куча графических элементов; он не имеет представления об их связях, и при редактировании диаграмма «расползается». А результат (штук пятнадцать красивых картинок) понадобился если не послезавтра, то хотя бы на следующей неделе. Поэтому было решено прекратить тесное общение с `drakon`ом, скормив ему готовые диаграммы.

Файл `имя.drn`, в котором `drakon` хранит диаграммы, является базой данных SQLite и может быть создан последовательностью SQL-запросов; собственно диаграммы и графические элементы создаются `INSERT`-ами в соответствующие таблицы.

Для описания диаграмм человеком создан язык, позволяющий вводить последовательности действий, выделять и именовать блоки кода, в том числе ветвления с проверкой условия и циклы с переменной-счётчиком. Краткое описание синтаксиса языка:

```
текст ::= строка переводстроки текст | конецфайла
строка ::= описаниешага '//' комментарии | описаниешага
комментарии ::= комментарий '//' комментарии | комментарий
описаниешага ::= заголовок '{' | '}' | описаниедействия
заголовок ::= условие '?' | перем '=' нач ':' кон |
              названиеблокакода
```

Псевдокод на этом языке переводится в SQL-запросы Рефал-программой [2] за три основных этапа (каждый этап — несколько более мелких шагов):

1. Ввод текста, распознавание и преобразование его структуры
2. Размещение блоков на диаграммах
3. Дорисовка соединительных линий и вывод SQL-запросов

Для размещения графических элементов потребовалось вычислять их размеры и координаты. Координаты X зависят от вложенности блоков кода и удобнее вычисляются, если программа представлена как лес. Координаты Y на диаграмме вычисляются в порядке следования по программе как плоскому списку.

От набора формул к схеме

Другая задача — перевод набора формул ЯСП (описан в отдельном докладе) в команды языка `dot` пакета `graphviz`. Перевод выполняется Рефал-программой [3]. Основные его этапы:

1. Ввод и разбор формул и построение леса, перевод выражений в префиксную форму
2. Выделение представляющих интерес общих подвыражений
3. Представление леса как списка узлов (корни и листья именованы) и дуг

4. Склейка одноимённых узлов
5. Нумерация узлов, формирование команд выходного языка.

Замечания о программировании

Последовательность преобразований реализована функцией

```
ЦепьПреобразований {
    .вхарг (.прочие (?функция .допарг)) =
        <Mu ?функция .допарг
            <ЦепьПреобразований .вхарг (.прочие)>>;
    .вхарг () = .вхарг;
}
```

— используя которую удобно наращивать цепочку вызовов; порядок вызовов слева направо, поэтому код сохраняет читаемость.

Особое внимание уделяется построению плана преобразований, одни из которых изменяют структуру данных в целом, а другие обрабатывают локальные ситуации (обычные или особые). Документирование такой программы — отдельная задача.

Чтобы хорошо выделить модули, лучше решать несколько похожих задач. Поэтому сейчас модули выделены только для ЯСП.

При расчётах размеров и координат терм нагружается подтермами, и было бы удобно искать значения по «ключу» — что требует реализовать функцию поиска. Сейчас такой функции нет, и программист следит за расположением подтермов сам.

Литература

[1] *Трусов С.* strusov/refal-machine: Исполнитель диалекта Рефал. `https://gitflic.ru/project/strusov/refal-machine`

[2] *Исакевич Д.* Преобразователь псевдокода в диаграммы ДРАКОН. `https://gitflic.ru/project/dvisa/drakoder`

[3] *Исакевич Д.* Преобразователь формул ЯСП в схемы GraphViz. `https://gitflic.ru/project/dvisa/metahim`

Даниил Исакевич
Владимир, ООО «БизнесСофтСервис»

Проект: metahim `https://gitflic.ru/project/dvisa/metahim`

Формальный язык соединений и превращений для описания развития и его реализация

Аннотация

Язык записи химических реакций обобщён для описания развития. Показан пример представления высокоуровневого понятия на полученном «метахимическом» языке. Предложена графическая нотация, изображающая наборы формул языка в виде графов и реализованная средствами Рефал-М и `graphviz`.

Ключевые слова: *язык, реакция, развитие, визуализация, Рефал.*

Предыстория

Классической логики для описания развития недостаточно; нужна неклассическая [1] со своим языком — например, общеизвестным языком записи химических реакций. Попытки расширить его применения: учение о «специальной химии», которая «рассматривает как элемент каждую субстанцию, имеющую отдельную функцию, даже самую сложную» [2]; утверждение о том, что «связи в ландшафтах напоминают цепочку реакций, которые совершаются в них» [3]; переусложнённая линейная логика Жирара [4].

Описание языка

Две связки, достаточные для записи реакций — соединение «+» (коммутативное и ассоциативное) и превращение «→»; «+» имеет приоритет перед «→», так что выражение

$$H_2 + H_2 + O_2 \to H_2O + H_2O$$

пишем без скобок. Чтобы строить формулы-определения, добавим в язык символ «=». Пример:

$$\text{Горение водорода в кислороде} = H_2 + H_2 + O_2 \to H_2O + H_2O.$$

Формула $C = A + B$ определяет C — соединение A и B. Формула $C = A \to B$ определяет «элемент» C — «субстанцию, имеющую отдельную функцию» [2] превращать A (при его наличии) в B. Эти «субстанции» сами могут участвовать в соединениях и превращениях. Так мы получаем «химическую» надстройку над языком реакций — «метахимический» язык соединений и превращений (ЯСП). Соединения этих «субстанций» могут повлечь недетерминированность их функции, например

$$(X \to A_1) + (X \to A_2)$$

есть превращение X в одну из альтернатив A_1 и A_2. Это превращение можно использовать, чтобы формализовать понятие выбора A_1 и A_2:

$$\begin{aligned}
\text{Переход к } A_1 &= X \to A_1,\\
\text{Переход к } A_2 &= X \to A_2,\\
\text{Переход к } A_1 \text{ либо к } A_2 &= \text{Переход к } A_1 + \text{Переход к } A_2,\\
\text{Выбор } A_1 &= \text{Переход к } A_1 \text{ либо к } A_2 \to \text{Переход к } A_1,\\
\text{Выбор } A_2 &= \text{Переход к } A_1 \text{ либо к } A_2 \to \text{Переход к } A_2.
\end{aligned} \tag{7.1}$$

Полученный набор формул — второго уровня (содержит превращения превращений).

Графическая нотация

Набор формул ЯСП изображается схемой. Пример — на рисунке 1.

Схема — граф, имеющий узлы имён (с текстовыми метками), соединения «+» (частей в целое), превращения «→» (входа в выход). Дуги входа и выхода снабжены стрелкой, направленной соответственно к узлу превращения и от него. Ещё одна дуга при узле превращения не имеет стрелок, определяет само превращение и может быть связана другим концом с узлом имени. Одна из дуг при узле соединения изображает соединение, а остальные дуги — его составляющие.

При разных инструментах обозначения могут различаться. Например, узел соединения естественно рисовать как слияние линий, но пакет `graphviz` не умеет этого делать правильно. В `graphviz` узел соединения обозначим точкой, а дугу соединения пометим дужкой,

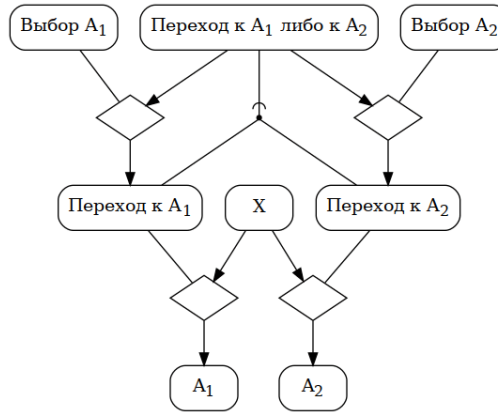


Рис. 1: Схема набора формул «Выбор» (7.1)

раскрытой к этому узлу. Узел превращения изобразим ромбом; входы и выходы — стрелками; дугу превращения — без стрелок.

Катализаторы

Взаимодействие — реакция вида $A + B \to C + D$. Если одна из взаимодействующих частей неизменна, то реакция — действие неизменной части на меняющуюся. Например, в формуле

$$\text{Чтение} = \text{Читатель} + \text{Книга} \to \text{Умный читатель} + \text{Книга}, \qquad (7.2)$$

учтено лишь воздействие книги на читателя. Неизменную действующую часть («катализатор» превращения) будем выделять явно, что делает формулы

$$\text{Чтение} = \text{Читатель} \xrightarrow{\text{Книга}} \text{Умный читатель}. \qquad (7.3)$$

и схемы (рисунок 2) выразительней. На схеме катализатор будем изображать при узле превращения как отдельную дугу с двумя стрелками.

Программная реализация

Для ввода связок ЯСП с клавиатуры используются `+`, `->`, `=:`

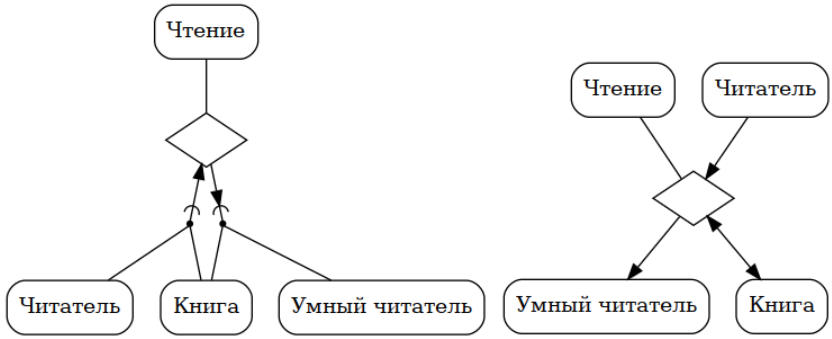


Рис. 2: Схемы формулы «Чтение»: слева без явного выделения катализатора (7.2), справа с явным выделением катализатора (7.3)

```
Чтение = Читатель + Книга -> Умный читатель + Книга.
```

Имена можно снабжать HTML-разметкой. Формулу можно размещать на последовательных строках и нужно завершать точкой. Конец набора формул — пустая строка либо конец файла. Программа [5] на Рефале [6] читает набор формул и превращает его в описания узлов и дуг, которые включаются в описание графа схемы для программы `dot` пакета `graphviz`.

Литература

[1] *Непейвода Н. Н.* Прикладная логика: учебное пособие. 3-е изд., существ. перераб. и доп. — Москва, Берлин: Директ-Медиа, 2019. — 575 с.: ил. DOI: 10.23681/561272

[2] *Успенский П. Д.* В поисках чудесного. Фрагменты неизвестного учения. Перевод Н. В. фон Бока. — СПб.: Издательство Чернышева, 1992. `https://lib.ru/URIKOVA/USPENSKIJ/poiski.txt`

[3] *Кружалин В. И., Симонов Ю. Г., Симонова Т. Ю.* Человек, общество, рельеф: Основы социально-экономической геоморфологии. — М.: Диалог культур, 2004. — 120 с., 8 с. цв. вкл.

[4] *Girard J.-Y.* Linear Logic: Its Syntax and Semantics. `http://girard.perso.math.cnrs.fr/Synsem.pdf`

[5] *Исакевич Д.* Преобразователь формул ЯСП в схемы GraphViz. `https://gitflic.ru/project/dvisa/metahim`

[6] *Трусов С.* strusov/refal-machine: Исполнитель диалекта Рефал. `https://gitflic.ru/project/strusov/refal-machine`

Михаил Чернигин
Саратов, ООО «Базальт СПО»

Проект: Конфигурирование операционной системы
`https://www.altlinux.org/Alterator_на_D-Bus`

Опыт разработки нового Альтератора

Аннотация

Доклад посвящён текущему состоянию разработки нового Альтератора — конфигуратора операционных систем «Альт». Будут рассмотрены новые возможности и подходы разработки подсистем Альтератора, которые позволяют вести разработку на различных языках программирования с различными графическими и консольными фреймворками.

Ключевые слова: *Альтератор, Qt, DBus.*

Зачем мы разрабатываем новый Альтератор?

Альтератор — это инсталятор и конфигуратор операционных систем «Альт». Первоначальная его реализация предполагает возможность написания модулей на языке Scheme для фронтендов и на языке Bash для бэкендов. В этой реализации возникали следующие проблемы:

- отсутствие API модулей для приложений;
- Scheme с Qt значительно усложняет написание фронтендов;
- запуск в графическом интерфейсе доступен только по паролю root.

Целями создания нового Альтератора являются:

- спецификация интерфейсов бэкендов для создания фронтендов в различных графических и консольных приложениях;
- обеспечение свободы в выборе технологий для реализации фронтендов и бекэндов;
- предоставление разработчикам фреймфорка для создания собственных приложений;

- обобщение механизмов, используемых как для локального, так и удалённого конфигурирования системы.

Как это работает?

В основе нового Альтератора лежит служба alterator-manager, которая и занимается управлением всеми объектами на шине DBus.

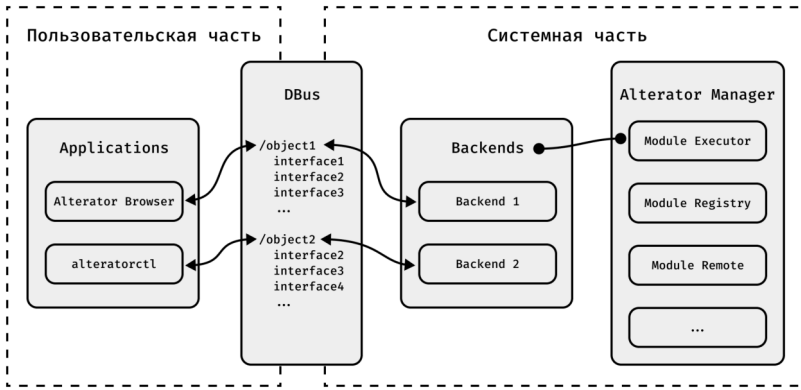


Рис. 1: Архитектура нового Альтератора

Создание новой подсистемы Альтератора состоит из двух частей.

1. Реализация бэкенда, предоставляющего свой интерфейс на DBus в объекте `/ru/basealt/alterator`.
2. Реализация фронтенда, то есть пользовательского приложения, которое использует этот интерфейс на DBus.

Бэкенды

Модули Альтератора создают интерфейсы на DBus. В том числе модуль executor упрощает этот процесс, путём использования `.backend` файлов. Executor парсит эти файлы и генерирует по ним соответствующие интерфейсы на DBus. Например, рассмотрим инструмент диагностики сети:

```
[numbers=left]
```

```
[Alterator Entry]
Type = Backend
Module = executor
Name = diag_network
Interface = diag1

[Info]
execute = cat /usr/share/alterator/diag/diag-network.diag
stdout_bytes = enabled
action_id = Info

[Run]
execute = diag-network {test}
stdout_signal_name = diag1_stdout_signal
stderr_signal_name = diag1_stderr_signal

[List]
execute = diag-network -l
stdout_strings = enabled
```

При запуске службы alterator-manager, можно увидеть на DBus новый объект:

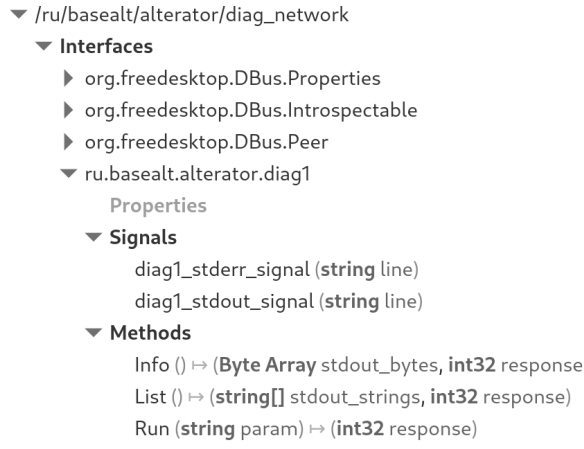


Рис. 2: Получившийся из `.backend` файла интерфейс на DBus.

Фронтенды

Фронтенд можно реализовать в виде графического приложения на любом языке и фреймворке, которое будет обращаться к предоставляемому бекэндом интерфейсу на DBus. Например, для работы с инструментами диагностики существует приложение Alterator Diagnostic Tool, которое с помощью alterator-manager получает все объекты, реализующие интерфейс `diag1` и отображает их в своём интерфейсе.

Чтобы реализованное приложение появилось в Alterator Browser, ему так же необходимо создать объект на DBus с интерфейсом `application1`. Например, для Alterator Diagnostic Tool:

```
[numbers=left]
[Alterator Entry]
Type = Backend
Module = executor
Name = adt
Interface = application1

[Info]
execute = cat /usr/share/alterator/applications/adt.application
stdout_bytes = enabled
```

Текущие недостатки и планы по их исправлению

Как уже было сказано в начале, новая архитектура определённо упрощает разработку и даёт больше свободы при выборе технологического стека, но были замечены и некоторые недостатки такого подхода.

Больше всего переживаний вызывает количество кода, до которого разрастаются фронтенды. В виду того, что все фронтенды являются независимыми, они часто требуют повторной реализации обращений и обработки сообщений через DBus и построения объектов в рамках ООП.

На данный момент решением данной проблемы видится содание библиотеки, условно называемой libalterator, предоставляющей инструменты для работы с alterator-manager на шине DBus, отправки сообщений и парсинга ответов.

Скачедубов Данила
Саратов, «Базальт СПО»
https://github.com/danila-Skachedubov/gpupdate

Архитектурный подход к хранению системных настроек

Аннотация

Доклад посвящён особенностям хранения системных настроек в отдельных конфигурационных файлах, что является текущей практикой в большинстве систем. В условиях современной разработки такое раздельное хранение становится громоздким и сложноуправляемым. В докладе будет предложено решение этой проблемы путём систематизации хранения конфигурационных параметров с использованием инструмента `Dconf`. Основное внимание будет уделено преимуществам централизованного подхода, включая упрощение управления настройками и возможности динамического обновления параметров для запущенных приложений без необходимости их перезагрузки.

Ключевые слова: *dconf, хранение системных настроек.*

В фоновых сервисах возникает необходимость обновления конфигурационных параметров в режиме реального времени. Обычно это достигается перезапуском сервиса или отправкой специализированных сигналов, например, `SIGUSR1` или `SIGUSR2`, чтобы сервис перечитал конфигурационные файлы. Однако подобная логика обновления параметров должна быть реализована в каждом приложении заново, что усложняет разработку и поддержание кода.

Для того чтобы избежать необходимости отправки сигналов, таких как `SIGUSR1` или `SIGUSR2`, приложения могут использовать шину D-Bus и единообразный интерфейс, позволяющий уведомлять о внесённых изменениях конфигурации в реальном времени. `Dconf` предоставляет решение этой задачи, позволяя распределять ключи конфигурации по группам, каждая из которых имеет соответствующий префикс, относящийся к конкретному приложению. Эта конфигурация может храниться как в централизованном наборе баз данных, так и в специализированных, обеспечивая гибкость в управлении. Примером такого подхода является использование `Dconf` в IBus для управления его настройками.

Архитектура и принципы работы Dconf

`Dconf` основан на концепции «профиля», который представляет собой список баз данных конфигурации. Профиль определяет порядок загрузки этих баз, задавая приоритеты при чтении и записи данных. Базы данных бывают системные, хранящиеся в /etc/dconf/db/, и пользовательские, расположенные в ~/.config/dconf/user. Пользовательские настройки перезаписывают системные для конкретного пользователя. В профиле базы данных расположены в порядке приоритета вычитывания, с первой в списке имеющей наивысший приоритет. `Dconf` также поддерживает «блокировки», которые предотвращают изменение ключей в базах данных, следующих за заблокированным ключом, если блокировка выставлена в нескольких базах, то приоритет вычитывания будет инвертирован.

Например, традиционный базовый профиль может выглядеть так:
user-db:user
system-db:local
system-db:site

В этом примере пользовательская база данных (user-db:user) имеет наивысший приоритет, позволяя пользователю переопределять локальные системные настройки (system-db:local). Однако настройки в базе system-db:site могут содержать блокировки, которые запрещают изменение определённых параметров, даже если пользователь пытается их изменить через свою базу данных.

Для разработки и применения групповых политик в `Dconf` предусмотрены два типа расширенных профилей: пользовательские и системные. Был добавлен слой Policy, который располагается между слоями Local и User, что позволяет отделить управляемые политиками параметры от локальных настроек. Это даёт возможность точно контролировать, какие параметры будут регулироваться политиками, а какие останутся на усмотрение локальных настроек.

Введён также слой Default, который обеспечивает наличие базовых настроек по умолчанию, используемых системой, если они не переопределены другими слоями. Это гарантирует, что всегда будут применены какие-то настройки, даже если остальные слои не задействованы. Дополнительно в Dconf используется возможность блокировки (Lock), которая защищает важные параметры от изменений. Совмещение слоёв Policy и Local с блокировками позволяет точно

определить, какие параметры могут быть изменены пользователем, а какие защищены от изменений.

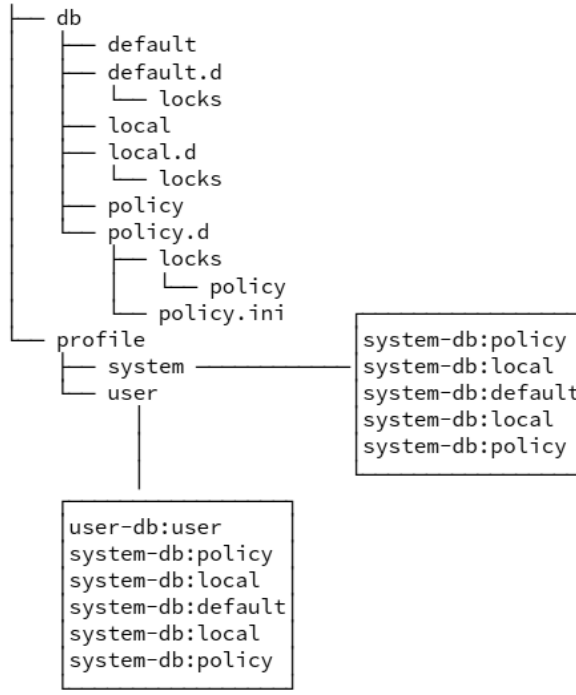


Рис. 1: Схема расширенных профилей.

Текущее использование Dconf и планы на дальнейшую разработку

В версии 0.10.0-alt1 в утилите gpupdate был изменён способ хранения ключей реестра, полученных из шаблонов групповой политики (GPT), переместив их из SQLite в `Dconf`. Это улучшило прозрачность и управляемость ключами политик, что позволило простыми методами реализовать утилиту, отображающую текущее состояние применённых политик.

Dconf теперь используется для хранения как системных, так и пользовательских политик, а также для хранения конфигурационных настроек самого Alterator, который управляет настройками системы. Механизм Dconf планируется использовать для централизованного хранения настроек различных сервисов и приложений, разрабатываемых в рамках проекта Alterator, ADMC и других приложений. Это позволит унифицировать управление конфигурациями и обеспечить централизованное хранение параметров для всех компонентов системы.

Литература

[1] ALT Linux Wiki: Dconf, [Электронный ресурс] URL: https://www.altlinux.org/Dconf

[2] Dconf System Administrator Guide, [Электронный ресурс] URL: https://wiki.gnome.org/Projects/dconf

Мария Алексеева
Саратов, ООО «Базальт СПО»

Проект: GPResult https://github.com/alxvmr/gpresult

GPResult: реализация инструмента для отображения и анализа групповых политик

Аннотация

Анализ применяемых групповых политик является важным этапом в своевременной отладке проблем, возникающих в доменной инфраструктуре. Утилита GPResult, разработанная для ALT Linux, предоставляет системным администраторам возможность проведения подобного анализа. Архитектурные особенности приложения и связанные с ним разработки изложены в докладе.

Ключевые слова: *Групповые политики, администрирование, домен, gpresult*

Система управления групповыми политиками позволяет администраторам применять определённые правила и ограничения, что обеспечивает стандартизированную безопасную вычислительную среду. Однако в момент, когда инфраструктура разрастается и становится

сложнее, определение применимости политик к пользователям и машинам проходит через определённые трудности. В частности, такое поведение обусловлено неочевидностью порядка применения групповых политик.

И в такой ситуации возникает необходимость эффективного устранения возникших неполадок. Первоначально для этого требуется отчёт, отражающий сведения об объектах групповой политики: имя политики, её ключи и значения, GUID, версия и прочее. Все эти данные отображает утилита `GPResult`, аналога которой ранее не существовало в Linux.

Архитектурные особенности

Как показано на рисунке 1, работа `GPResult` связана с артефактами работы инструмента применения групповых политик — утилиты `GPUpdate`. `GPUpdate` (до версии 0.10.0-alt1) кэшировал обработанные объекты групповой политики (GPO) в базе данных SQLite. Позже архитектура сменилась — с версии 0.10.0-alt1 ключи реестра, полученные из GPO, хранятся в базе `Dconf`. Такое решение позволило внешним приложениям получить доступ к информации о применённых GPO.

`GPUpdate` применяет групповые политики как на системном уровне, так и для отдельных пользователей. В результате, информация о применённых политиках (в виде списка ключей и значений) сохраняется в неструктурированной бинарной базе данных формата GVDB — `/etc/dconf/db/policy<UID>`, где UID — идентификатор пользователя. Для машины информация сохраняется по пути `/etc/dconf/db/policy`. Отличительной особенностью этих файлов является хранение записей об уже применённых групповых политиках (в отличие, например, от `.ini-файлов`).

Указанные базы данных предоставляют два набора ключей, относящихся к одному ключу реестра — один предоставляет метаданные, другой — значение ключа. Причём ключ с метаданными отличается от ключа со значением добавлением префикса `/Source`. Так например, для ключа `/Software/KeyName` ключ с метаданными будет иметь название `/Source/Software/KeyName`.

Ключи метаданных включают в себя необходимую информацию для GPResult:

Рис. 1: Взаимодействие GPResult и GPUpdate

- имя групповой политики (`policy_name`), к которой относится ключ;
- тип данных значения (`type`);
- сведения о перезаписи ключа (`reloaded_with_policy_key`);
- является ли значение ключа списком (`is_list`).

Фактически, для реализации основной задачи `GPResult` необходимо вычитать и правильно интерпретировать информацию из указанных баз данных. Однако предоставляемое API для чтения базы GVDB применимо к проектам на C. `GPResult` в свою очередь написан на Python. Решением возникшей проблемы стало использование GObject Introspection, поскольку в реализации API используются объекты GObject.

Данный механизм является промежуточным слоем между библиотеками C (использующими GObject) и языковыми привязками. Библиотеку можно сканировать во время компиляции и генерировать файлы метаданных в дополнение к ней. Затем языковые привязки могут считывать полученные метаданные и автоматически предоставлять привязки для вызова библиотеки C (см. рис. 2). В результате, дополнительно был собран пакет, предоставляющий метаданные о проекте.

Первым шагом из исходного кода и заголовков генерируется `.gir`-файл, содержащий информацию о самоанализе библиотеки. После, при помощи компилятора из этого файла генерируется бинарный

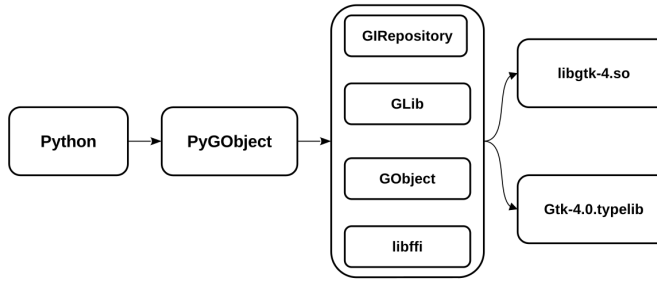


Рис. 2: Принцип взаимодействия языковой привязки и файлов самоанализа библиотеки

файл `.typelib`, предоставляющий метаданные для языковой привязки (PyGObject).

Функциональные возможности GPResult

Актуальная версия утилиты `GPResult` поддерживает сбор и отображение сведений о применённых групповых политиках для текущего пользователя и локальной машины. Приложение является консольным и поддерживает следующий набор опций:

- выбор формата вывода: отображение применённых ключей и значений (raw); сводных данных, включающих только имена групповых политик (common); вывод списка, соотносящего имена GPO с их GUID (list); отображение подробных сведений о политиках (verbose);

- выдача информации о применённой групповой политике по её имени или GUID;

- указание типа объекта (`user` или `machine`).

Также утилита поддерживает два языка — русский и английский. Выбор языка происходит автоматически и зависит от языка системы.

Перспективы развития

На данный момент `GPResult` предоставляет возможность отображения данных о применённых групповых политиках для текущего пользователя и локальной машины. Проект активно развивается, и в будущем планируется расширить функционал: добавить поддержку удалённых пользователей и компьютеров; реализовать предоставление информации о перезаписанных ключах и приоритетах политик; разработать механизм генерации html-отчёта.

Елена Дятленко, Антон Абрамов
Москва, Саратов, Базальт СПО

Проект: ALT Diagnostic Tool https://gitlab.basealt.space/alt/adt

Перспективы развития инструментов диагностики в ОС «Альт»

Аннотация

Alterator Diagnostic Tool (ADT) — графический инструмент диагностики системы, приложение для запуска наборов с тестами в графическом или консольном интерфейсе. В докладе рассматривается опыт ближайших наработок и долгосрочные планы изменений в утилите.

Ключевые слова: *Диагностика, D-Bus, система, Альтератор.*

Опытный администратор как правило использует широкий набор известных утилит для диагностики проблем. Почти всегда работа происходит в терминале, поскольку большинство приложений не имеют графический интерфейс. Для необычных случаев приходится изучать документацию к имеющимся утилитам или искать новые, удовлетворяющие требования по исследованию неполадок. Весь этот процесс может занимать достаточное время, особенно для начинающих пользователей Linux. Дополнительные трудности появляются когда опытному администратору недоступна связь с удалённой машиной, а оператор рабочего места не владеет навыками работы в терминале. В таких случаях будет удобно запустить на рабочем столе окно программы, выполнить заранее подобранные тесты, получить отчёт и отправить его детали, например, по электронной почте.

Задуманная цель ADT — упростить проверку локальной системы и инфраструктуры. Предполагается, что базовые наборы с диагностическими тестами пойдут в комплекте с графическим фрэймворком. Хотя утилита допускает достаточно простое подключение собственных наборов проверок, которые «под себя» подготовит системный администратор или IT-служба для внутреннего контура.

Сейчас утилита развивается по нескольким направлениям: расширение возможностей для тестирования, развитие отчётов и журналов, удобство интерфейса (UI/UX), ограничения доступа для пользователей, документирование, базовые инструменты диагностики.

Ближайшие разработки

В версии 0.1.4-alt1 обновлен внешний вид и появилась возможность работать с объектами на сессионной шине D-Bus (ранее была только системная шина). Написана документация проекта.

При работе через системную шину D-Bus диагностические скрипты выполняются с правами суперпользователя, что бывает не всегда необходимо. Поддержка сессионной шины позволяет запускать инструменты диагностики от пользователя в текущей сессии. Например, это позволит выяснить kerberos-билет пользователя, запустившего тест, или состояние пользовательской службы SystemD.

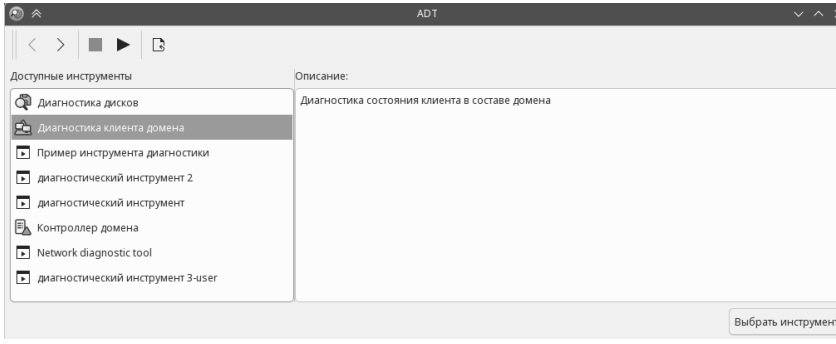


Рис. 1: ADT — выбор инструмента диагностики.

Обновился внешний вид утилиты, появилась поддержку тёмной темы. Элементы управления инструментами теперь располагаются на панели инструментов для удобства работы и масштабирования окна.

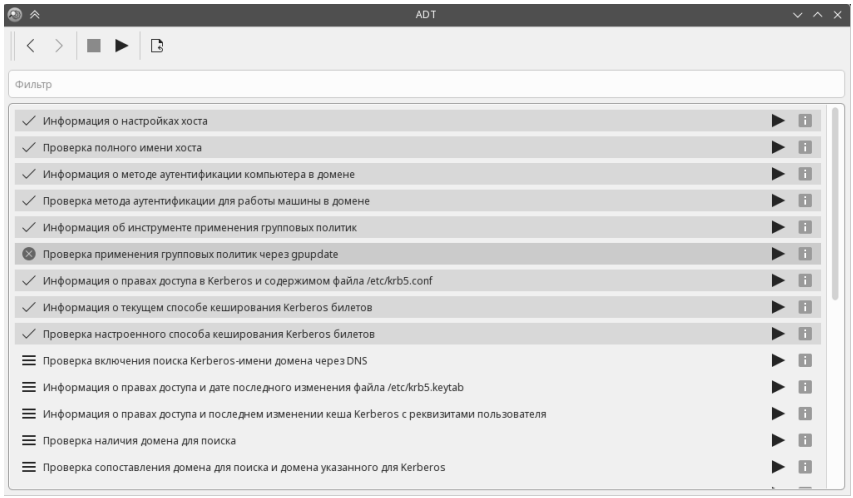


Рис. 2: ADT — Запуск тестов инструмента диагностики.

Описание к инструментам диагностики можно готовить на языке разметки HTML со стилями CSS. Архитектура утилиты позволяет обрабатывать файлы с разметкой HTML и CSS, и даёт возможность подробно описывать наборы диагностических тестов. Разметка должна соответствовать спецификации HTML 4. ADT частично поддерживает CSS разметку в соответствии с документацией Qt.

Планы развития

Поскольку одна из задач ADT — упростить взаимодействие оператора с технической поддержкой, будет развиваться система журналов и отчётов. У каждой проверки есть собственный журнал с деталями, и в рамках одного инструмента можно получить обобщённый журнал всех проведённых тестов. Помимо общего журнала, утилита сформирует архив отчёта (report), в котором будет содержаться текст общего журнала и системные конфигурационные файлы.

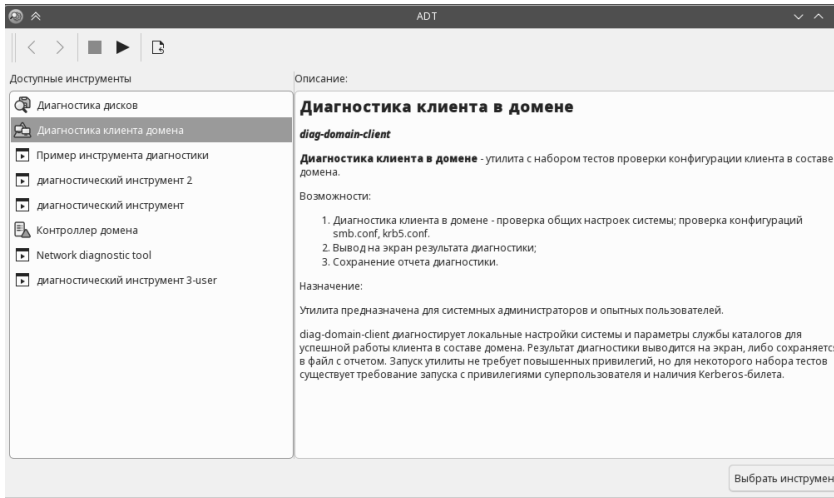


Рис. 3: ADT — Пример многострочного описания инструмента диагностики.

Семен Фомченков, Максим Слипенков
Обнинск, Донецк, ИАТЭ НИЯУ МИФИ, ДонНТУ

Проект: ALT Gnome https://gitflic.ru/project/alt-gnome/karkas, `https://t.me/alt_gnome`

Модульная платформа для разработки Telegram-ботов «Каркас»

Аннотация

Открытая платформа для создания Telegram-ботов, упрощающая создание self-hosted решений с модульным функционалом для различных сфер использования. От администрирования Telegram чатов до создания простой CRM-системы для нужд сообществ. Разработка ведётся командой ALT Gnome Infrastructure.

Ключевые слова: *чат-бот, администрирование Telegram, модульность, CRM.*

Проблемы существующих платформ для разработки чат-ботов

Современные платформы для создания чат-ботов часто представляют собой монолитные системы, ограничивающие возможности разработчиков в кастомизации и расширении функциональности. Такие системы затрудняют адаптацию ботов под специфические требования проектов, снижают эффективность разработки и усложняют внедрение новых возможностей. В подавляющем большинстве случаев чат-боты распространяются в формате готового решения, без возможности его изменения и разворачивания на своей инфраструктуре.

«Каркас»: модульная платформа для Telegram-ботов

«Каркас» — это платформа для разработки модульных Telegram-ботов, призванная упростить взаимодействие с чатами, разрабатываемая под лицензией GNU GPL v3. Она предоставляет возможность расширять функциональность бота с помощью интеграции различных блоков. Такой подход позволяет разработчикам легко добавлять, удалять и изменять компоненты без затрагивания основной структуры приложения. А возможность самостоятельного развёртывания обеспечивает полный контроль над данными и функциональностью бота. В основе проекта «Каркас» лежит язык программирования Python 3 и библиотека aiogram3 для асинхронной работы с Telegram API.

Блоки: расширение функциональности через модульность

Блоки в «Каркасе» — это независимые компоненты, добавляющие различные функции боту. Стандартные блоки предоставляют базовые возможности, такие как автоматизированное модерирование чата, ведение статистики, информирование участников чата о правилах и другой информации, создание графических интерфейсов на базе платформы Telegram Mini Apps и работа с базой данных (функциональность необходимая для работы других модулей). Дополнительные официальные блоки, созданные командой разработки платформы «Каркас», расширяют функциональность бота, например позволяя другим блокам работать с нейросетевыми моделями или добавляя функциональность составления стандартизированных отчётов об ошибках для bugzilla.altlinux.org. Разработчики также могут созда-

вать и интегрировать собственные блоки, что обеспечивает гибкость и адаптивность платформы.

Преимущества использования «Каркаса»

- **Гибкость и масштабируемость**: модульная архитектура позволяет легко добавлять и удалять функциональные компоненты.
- **Упрощение разработки**: использование готовых блоков снижает сложность при создании и поддержке ботов.
- **Сообщество и сотрудничество**: открытый исходный код способствует коллективной работе и обмену знаниями.

Обучение и развитие через участие в СПО-проектах как основа проектов ALT Gnome

«Каркас» служит отличной платформой для обучения студентов и начинающих разработчиков. Участие в проекте позволяет приобрести практические навыки работы над реальными задачами, научиться писать поддерживаемый код на языке Python, использовать современные инструменты и технологии, а также развить навыки командной работы.

Заключение и перспективы развития

«Каркас» стремится стать универсальной платформой для разработки Telegram-ботов, способной удовлетворить потребности широкого круга пользователей. Мы уверены, что модульный подход и открытость проекта внесут вклад в направление разработки чат-ботов. В ближайших обновлениях «Каркас» мы планируем начать интеграцию функциональности CRM-систем, а также реализовать возможность работы с такими протоколами как CalDAV. Кроме создания новых блоков, так же планируется и улучшение анти-спам защиты благодаря интеграцией открытой Telegram спам-базы «Combot» и решений на базе математической классификации и локальных нейронных сетей.

Кирилл Изместьев
Киров, ООО «Базальт СПО»
https://git.altlinux.org/people/felixz/packages/?p=localization-tt.git;a=summary

Татарская локализация ОС «Альт Образование»

Аннотация

В статье рассматривается проблема распространения татарского языка в информационном пространстве. А зачем вообще переводить компьютерные программы? В первую очередь, компьютер — это важное устройство, которое мы используем повседневно. Во-вторых, это элемент языковой среды. Язык — основа идентичности народа. Для сохранения языка важно, чтобы он постоянно развивался, адаптировался к новым реалиям, чтобы в языке появлялись современные слова, связанные с компьютерной тематикой. Описывается опыт по локализации оболочки XFCE 4 операционной системы «Альт Образование» с открытым исходным кодом на татарский язык.

Ключевые слова: *локализация, татарский язык, XFCE*

Введение: для чего это надо?

Локализация программного обеспечения — процесс адаптации программного обеспечения к культуре какой-либо страны. Как частность — перевод пользовательского интерфейса, документации и сопутствующих файлов программного обеспечения с одного языка на другой.

Зачем вообще переводить компьютерные программы? В первую очередь, компьютер — это важное устройство, которое мы используем повседневно. Во-вторых, это элемент языковой среды. Язык — основа идентичности народа. Для сохранения языка важно, чтобы он постоянно развивался, адаптировался к новым реалиям, чтобы в языке появлялись современные слова, связанные с компьютерной тематикой.

С помощью СПО-инструментов можно переводить приложения на родной язык и делиться результатом с мировым сообществом.

Всё начинается с добавления локали — набора параметров, определяющего региональные настройки пользовательского интерфейса,

такие как язык, страна, часовой пояс, набор символов, формат вывода даты, времени, используемая денежная единица и пр.

Установка региональных параметров является частью большого процесса локализации. При этом изменение региональных настроек часто доступно конечному пользователю системы без перепрограммирования программного обеспечения (ПО), в отличие от перевода сообщений пользовательского интерфейса, который часто требует изменения программного кода разработчиками ПО. Обычно код языка — это 2 символа.

Что имеется

1. Дистрибутив ОС «Альт Образование» 10.2. В библиотеке `glibc` существует 2 локали:
   `tt_RU` — для кириллицы, и
   `tt_RU@iqtelif` — для латинского написания.

2. Татарская клавиатура. Её можно добавить выбрав из списка «Русская» — «Татарская».

3. Стандартная установка «Альт Образование» включает графическую оболочку XFCE. Альтернативную оболочку KDE можно установить при установке системы. Для оболочки KDE была сделана татарская локализация, работа велась с 2005 года. Для XFCE 4 — не было. ОС «Альт Образование» используется в различных учебных заведениях, парк компьютеров в которых не везде самый свежий. Оболочка XFCE 4 из коробки потребляет несколько меньше ресурсов компьютера, и при выборе: что установить на не самое свежее железо? — лучше устанавливать XFCE.

Что было исследовано и добавлено

1. Формат переменной:

```
LANGUAGE=tt:ru
```

Смысл в том, что можно в переменной задать не один язык, а два и более. Если нет перевода для первого языка, то выбирается следующий.

В состав операционной системы входит большое количество пакетов, локализация добавляется поэтапно, поэтому компьютерные термины на языке какого-либо народа могут отсутствовать.

Народы России хорошо знают русский язык, а потому если не будет перевода терминов на родной язык, удобнее будет их видеть на русском, чем на английском языке.

Для этого добавляем строку в файле `/etc/profile.d/lang.sh`

```
export LANGUAGE=tt:ru
```

2. В пакете `libnatspec` татарская локаль уже была, но называлась `tt_ru.UTF8`, и если прописать иначе, то в Thunar вместо надписей на татарском языке отображаются не читаемые символы.

Задача: подготовить минимальный комплект для показа визуального ряда на татарском языке в базовой поставке «Альт Образование».

1. В обновлённой системе определяем элементы интерфейса, которые будут показываться:

— XFCE (включая панели, плагины и сеанс)

— Меню (категории и пункты меню)

— NetworkManager-applet-gtk

2. Исходим из того, что берём файлы, переведённые на русский, и декомпилируем .mo в исходный .po. Они лежат в каталоге `/usr/share/locale/ru/LC_MESSAGES/`

```
mkdir ~/tmp/tt
cd /usr/share/locale/ru/LC_MESSAGES/
for i in xfce4-*.mo NetworkManager*.mo;
             do msgunfmt $i -o ~/tmp/tt/${i/.mo/.po}; done
```

3. Подготавливаем строки с категориями (поле `Name` в `/usr/share/desktop-directories/*.directory`) и названиями приложений (GenericName и Comment) в `/usr/share/applications` (и ниже) в файлах `.desktop`:

```
sed -n 's/^Name=//p' /usr/share/desktop-directories/*.directory > ~/tmp/tt/DIRECTORY
sed -ne 's/^\(GenericName\|Comment\)=//pi' $(find /usr/share/applications
                                          -name \*.desktop) > ~/tmp/tt/DESKTOP
```

Патч в `libxfce4uti` для отображения надписей в меню на Татарском языке, а в случае отсутствия перевода — на русском (вместо английского) [2]. Результат на рис. 1 и рис. 2.

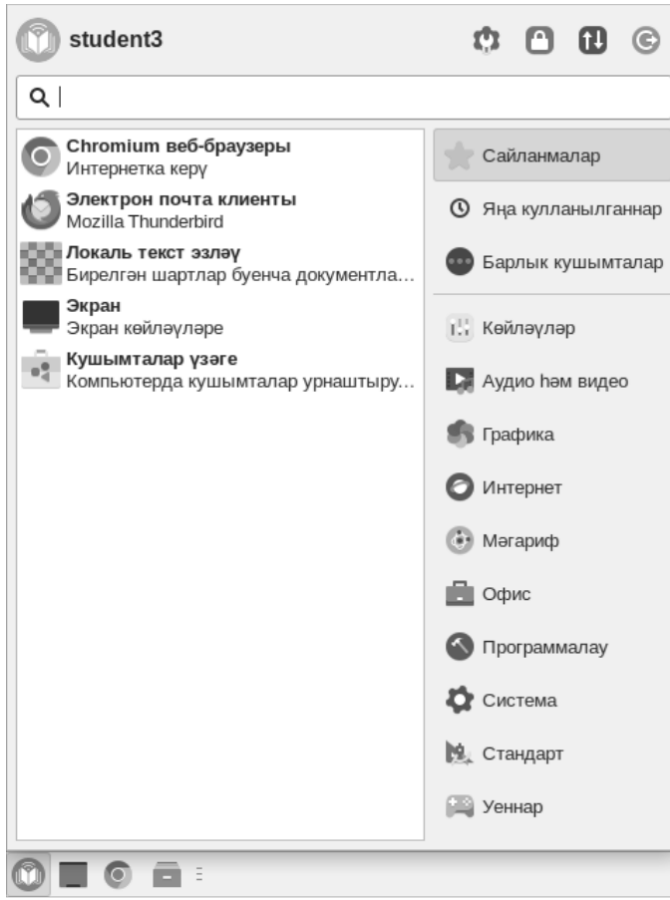


Рис. 1: Локализованное Меню.

3. Неочевидность перевода после локализации пакета lightdm-gtk-greeter: «Password» остаётся на английском, перевод хранится в пакете Linux-PAM. Он передавался на перевод отдельно после исследования.

4. Трудности, с которыми столкнулись переводчики — это, например, обратный порядок слов.

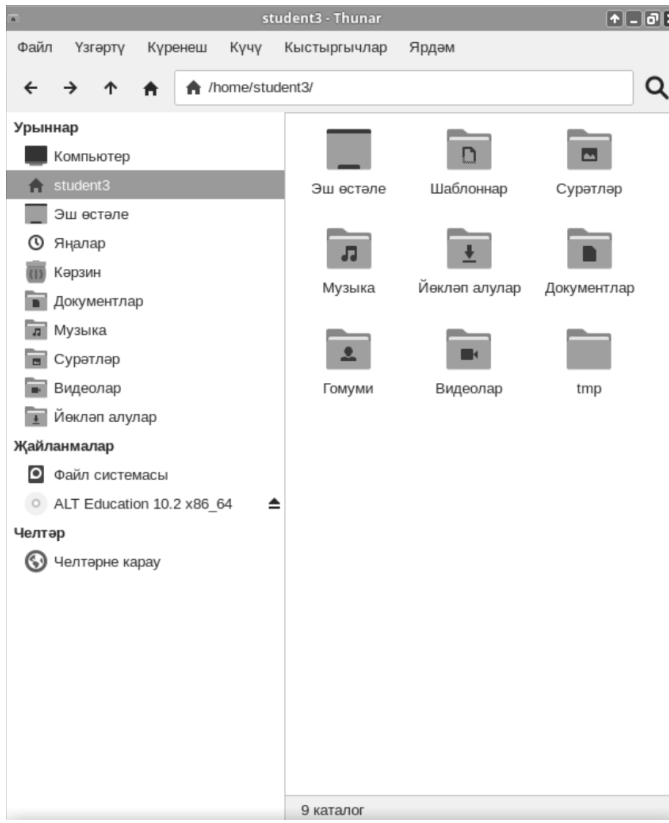


Рис. 2: Локализованный файловый менеджер Thunar.

В результате собран пакет `localization-tt`, содержащий `.po` файлы базовых приложений `XFCE`, которые видны на рабочем столе и в меню.

Тестировался (из сборочного задания) следующим образом:

`apt-repo test 355257`

в файле `/etc/locale.conf` меняем строку на:

`LANG=tt_RU.UTF8`

в файле `/etc/sysconfig/i18n` меняем строки на:

`LANG=tt_RU.UTF8`

`SUPPORTED=tt_RU.UTF8`

Добавляем строку в файле `/etc/profile.d/lang.sh`

`export LANGUAGE=tt:ru`

Устанавливаем пакет `libxfce4util` с патчем для отображения меню на татарском языке.

Скачиваем файлы `.desktop` и `.directories` [3].

Заменяем файлы с татарским переводом `/usr/share/desktop-directories` и `/usr/share/applications`.

Добавляем раскладку клавиатуры из списка Русская — Татарская.

Планы по развитию проекта

1. Дополнение и сопровождение перевода приложений, использующихся в ОС «Альт Образование» на татарский язык;
2. Сотрудничество с языковыми активистами и другими заинтересованными людьми, которые желают видеть перевод приложений на родном для себя языке.

Литература

[1] Перевод программ с помощью GNU gettext URL: `https://docs.weblate.org/ru/latest/devel/gettext.html`

[2] Патч в libxfce4uti для отображения надписей в меню на Татарском языке. URL: `https://gitlab.xfce.org/xfce/libxfce4util/-/commit/b2a2aee40956e194944031432d7147c0d5613e93`

[3] Архив подготовленных данных URL: `https://cloud.mail.ru/public/GxDX/C2oFYkapk`

Александр Речицкий, Брагин А.В.
Ставрополь, Москва, АНО ДПО Академия ТОР, МГТУ им. Баумана

Проект: ReactOS `https://reactos.org`

Операционная система ReactOS: сквозь тернии от альфы к бете

Аннотация

ReactOS — это свободная операционная система с открытым исходным кодом, основанная на архитектуре Windows, стремящаяся к высокой совместимости с существующими драйверами устройств и программами для ядра Windows NT 5.2 и в перспективе его более новыми версиями

Ключевые слова: *ReactOS*.

ReactOS — это свободная операционная система с открытым исходным кодом, основанная на архитектуре Windows, обеспечивающая поддержку существующих приложений, драйверов и опыта пользователя наиболее распространённой потребительской операционной системы.

Проект ReactOS развивается с середины 1990-х годов, однако на момент середины 2024 года, несмотря на наличие полнофункционального прототипа, он всё ещё находится в стадии альфа-версии. Это означает, что система всё ещё не является полностью стабильной для повседневного использования, и в ней могут встречаться ошибки.

Последняя версия ReactOS — это **0.4.14**, которая вышла в декабре 2021 года. В этой версии было внесено множество улучшений, в частности, исправления в ядре системы и оболочке, добавлена поддержка новых драйверов и устройств, а также усовершенствована работа с сетевыми протоколами и файловыми системами. Также продолжается разработка x64-версии системы, хотя она пока менее стабильна по сравнению с x86.

В версию ReactOS 0.4.15, которая находится в разработке и возможно будет выпущена в конце 2024 года, попадёт более 8600 изменений. Среди прочих планируются следующие ключевые улучшения:

Поддержка загрузки через UEFI

В эту версию будет добавлена экспериментальная поддержка UEFI для современных устройств, таких как x86, AMD64 и ARM. Она поз-

воляет запускать ReactOS на устройствах с прошивкой UEFI, где нет возможности запуска в режиме совместимости с BIOS

Улучшения отладчика ядра

Будет обновлён встроенный отладчик ядра (KDBG), что облегчает процесс отладки системы. ReactOS теперь поддерживает формат отладочных символов PDB, что улучшает совместимость с инструментами отладки, такими как WinDbg.

Улучшения в интерфейсе (Shell)

Пользовательский интерфейс был улучшен, в том числе для работы с ZIP-архивами с UTF-8 символами, а также исправлены ошибки и проблемы отображения иконок инструментов в старых программах, таких как MS Office 2000.

Улучшения Win32-подсистемы

Обновлено управление графикой, добавлена поддержка поворота изображений, улучшено управление памятью и повышена стабильность работы с приложениями DirectX.

Улучшения установщика и загрузчика

ReactOS FreeLoader теперь поддерживает загрузку с NTFS, управление памятью с использованием расширения физических адресов (PAE), а также улучшены обнаружение и обработка устройств при инициализации системы.

Литература

[1] *Брагин А. В., Иванов И. П.* Операционные системы. Лабораторные работы : учеб. издание. Москва : Изд. МГТУ им. Н. Э. Баумана, 2024. 36 с.

Аркадий Шейн
г. Москва, ИНФЕРИТ

Проект: ОС «МСВСфера» `https://msvsphere-os.ru/`, `https://git.inferitos.ru/explore/repos`

Как сделать хороший дистрибутив, чтобы не получился BolgenOS

Аннотация

Докладчик поделится личным опытом создания различных дистрибутивов Linux в России и рассмотрит проблему дистрибутивов, основанных на Red Hat Enterprise Linux.

Ключевые слова: *Red Hat, МСВСфера, Linux, дистрибутив.*

В 2011 году на просторах российского сегмента сети Интернет появилась новость о выходе «Принципиально новой революционной операционной системы», которая получила название BolgenOS [1]. Главным и единственным разработчиком дистрибутива был школьник из Нижнего Тагила. Новость облетела различные новостные ленты, включая Linux.org.ru и Хабр, а на местном ТВ даже появился репортаж.

Однако при ближайшем рассмотрении BolgenOS оказалась всего лишь немного изменённым дистрибутивом Ubuntu с удалёнными копирайтами и «нескучными обоями».

С тех пор название BolgenOS стало нарицательным, а любой новой ОС, основанной на пакетной базе другого дистрибутива Linux, злыми языками неизменно навешивается ярлык BolgenOS.

Автор, основываясь на личном опыте, проанализирует создание различных дистрибутивов, основанных на пакетной базе Red Hat, начиная с 90-х годов, и попытается ответить на вопрос, чем же самостоятельный дистрибутив отличается от «BolgenOS».

Первые дистрибутивы Linux не имели не только никакой русификации, но и никакой локализации. Если вам удалось поставить условный Red Hat Linux 5.0, то после установки вы понимаете, что не можете ни писать, ни читать на русском языке.

Проблемы была намного глубже, чем кажется на первый взгляд. В документе The Linux Cyrrilic HOWTO [2] от 1998 года насчитывается двенадцать глав. Это и проблема со шрифтами, и с раскладками клавиатуры, с TeX'ом и т. п.

На этой почве стали появляться первые исправленные версии Red Hat Linux. Сначала это был Urban Soft, потом появились Красная Шапочка и Red Hat MSIU, затем Black Cat Linux, который превратился в ASPLinux.

В Московском государственном индустриальном университете был создан респин MSIU ASPLinux который содержал большое количество обновлений и изменений под учебный процесс.

Постепенно изменения попадали в upstream, и острая нужда в подобных дистрибутивах отпала. Однако в 2008 году появился ещё один респин — Russian Fedora. Сперва дистрибутив возник как ответ на проблему с установкой Fedora 9 на русском языке, а затем как позыв к образованию сообщества в рамках инициативы Red Hat по созданию сертифицированного центра в России. В итоге дистрибутив выдержал 30 релизов.

На сегодняшний день за границей существует большое количество дистрибутивов как платных, так и бесплатных, основанных на пакетной базе Red Hat Enterprise Linux. Это Oracle Linux, AlmaLinux, Rocky Linux, EuroLinux, MiracleLinux, CloudLinux. В России же до недавнего времени не существовало ни одного дистрибутива основанного на 8 или 9 версии RHEL, хотя спрос на подобные дистрибутивы существует, например, на рынке хостинга.

С чем это было связано? В основном с тем, что в отличии от принципиально лёгкой пересборки RHEL 6/7 (обычно это делали основываясь на пакетной базе CentOS), собрать 8 или 9 версию совсем не так просто. На сборку «МСВСфера» 8 мы потратили четыре месяца. Это связано и с модульными пакетами и с тем, что в актуальном окружении старые пакеты уже не собираются.

ОС «МСВСфера» на данный момент полностью повторяет пакетную базу RHEL, а также содержит различные дополнения и улучшения, например, возвращена поддержка технологии SPICE, обновлён LibreOffice, добавлена возможность прослушивать музыку и смотреть фильмы и т. д.

Так что же нужно, чтобы у вас получился хороший самостоятельный дистрибутив?

- Система контроля версий для хранения исходных текстов доступная всему сообществу. Это важный шаг в открытой разработке.

- Система сборки (Koji) + Modular Build System (для сборки модульных пакетов).
- Багтрекер для регистрации ошибок от сообщества.
- Репозиторий (в том числе для исходников и debug-пакетов).
- Errata/OVAL (для сканеров уязвимостей).
- Документация.
- Система тестирования дистрибутива.
- Запись в реестре и сертификация ФСТЭК.
- «Нескучные обои», куда без них.

Но главное нужна команда, которая понимает, что нужно делать.

Таким образом, брать за основу другие дистрибутивы не зазорно, главное придерживаться принципа открытой разработки и отталкиваться от текущих реалий. Также ничто не мешает вам вносить улучшения, чтобы отличаться от оригинала в лучшую сторону.

Вы можете делать респины (без пересборки всех пакетов) какого-либо дистрибутива (будь то Fedora или ALT Linux) для удобства использования скажем в образовательной среде или под конкретные внутренние задачи, главное не забывать указывать источник пакетов.

И главное нужно стараться всегда отправлять важные изменения в upstream.

Литература

[1] Денис Попов, `https://neolurk.org/wiki/Денис_Попов`

[2] Alexander L. Belikoff, The Linux Cyrillic HOWTO, 1998,
URL: `http://tldp.yolinux.com/HOWTO/Cyrillic-HOWTO.html`

Антон Мидюков
Москва, ООО «Базальт СПО»

Проект: mkimage-profiles https://www.altlinux.org/Mkimage-profiles

И снова об mkimage-profiles: инструментарий сборки дистрибутивов ОС «Альт» (5 лет спустя)

Аннотация

Нововведения в инструментарии сборки дистрибутивов ОС «Альт» за 2020-2024

Ключевые слова: *дистрибутив, сборка, iso-образ.*

За прошедшие пять лет с момента предыдущего доклада [1] по `mkimage-profiles` было немало важных изменений.

Чтобы поддерживать несколько бранчей была введена переменная BRANCH (дефолтное значение соответствует бранчу хоста). Кроме того, это позволило удобно подключать нужный apt.conf в зависимости от заданного бранча, настроив `~/.mkimage/profiles.mk`.

Сборка `initrd` и добавление ядер в iso образы была перенесена из mkimage в `mkimage-profiles`. Были сделаны фичи initrd-propagator (для p10) и initrd-bootchain (замена propagator для p11, sisyphus). Теперь `initrd` собирается самим `make-initrd`, все модули прописываются в `initrd.mk`, больше никакого full.cz не собирается, на выходе обычный `initrd.img`.

Появилась возможность собирать iso-образы с несколькими ядрами.

Isolinux стал legacy, по дефолту в iso-образах используется grub-pc. Для isolinux было невозможно реализовать возможность выбора ядер при загрузке. Это пришлось бы делать на уровне gfxboot, который требует поддержку графики.

Поддержка rEFInd для загрузки iso в режиме EFI была прекращена. Вместо него используется `grub-efi`. Это позволило поддерживать загрузку в EFI в текстовом режиме и через последовательную консоль, а также использовать один конфиг grub для legacy и uefi, реализовать поддержку выбора ядер.

Для каждой стадии сборки готовятся списки rpm и srpm пакетов и сохраняются в `.disk/` у iso, и в `/root/.install-logs/` у `rootfs` и контейнеров. Что позволяет легко отслеживать изменения пакетной базы и выявлять баги.

Каждый iso теперь частично самодокументируется. В `.disk/` есть такие файлы: `mkinitrd` (команда для сборки `initrd`) и `initrd.mk` (конфиг для сборки `initrd.img`).

Появилась возможность использовать классический инсталлятор из live. Это цель use/live-install для установки распаковкой live. При этом «лишние» пакеты удаляются (те, что уже есть в live). Если добавить цель use/live-install/pkg, то установка будет происходить аналогично install2 из пакетов.

Появилась возможность использовать режим `rescue` в обычном live. Для этого нужно задействовать цель use/live/rescue. При этом в live должна использоваться система инициализации systemd, а в kernel cmdline будет прописан параметр `systemd.unit=rescue.target`.

Таким образом, стало возможным собирать дистрибутивы с одним сквошом live вместо трёх, что сильно уменьшает размер образа в современных реалиях.

Появился OEM-режим установки [3] для altinst use/install2/oem и для live use/live-install/oem.

Главным нововведением для нового стабильного бранча p11 стало изменение системы загрузки первой стадии (initrd) у iso-образов с `propagator` на `make-initrd-bootchain-altboot` [2], о котором вскользь упоминалось вначале. Об этом будет подробно в докладе.

Литература

[1] *А. П. Мидюков* Mkimage-profiles — гибкий инструмент сборки дистрибутивов для множества платформ. Доклад на OSSDEVCONF-2019. URL: `https://0x1.tv/20190828E`

[2] altboot URL: `https://www.altlinux.org/Installer/common/altboot`

[3] Установка в режиме OEM URL: `https://www.altlinux.org/Installer/OEM`

Виталий Липатов
Санкт-Петербург, Этерсофт

Проект: EPM https://eepm.ru

Перспективы epm play как средства установки приложений

Аннотация

Реализация `epm play` как средства скачивания пакетов с сайтов разработчиков имеет свои ограничения, которые планируется преодолеть техническими и организационными мерами.

Ключевые слова: *репозиторий, магазин приложений, бинарная совместимость*.

По мере увеличения числа поддерживаемых в `epm play` приложений (на данный момент около 250) стали острее проявляться недостатки нынешнего осторожного подхода (по сути, автоматизация скачивания с сайта, не более).

Анализ созданных для скачивания сценариев показывает, что их возможно преобразовать из скриптов в данные с информацией об источнике приложения, что сделает подход к доставке похожим на применяемый в AUR, winget или snap. Информация о приложениях будет отделена от `epm play` и доступна через API сервера.

Это решит частые проблемы со сторонним ПО, которые требуют нетривиальной логики и отдельных запросов к сайтам:

- распространение без пакетов (в архивах) с непростой инструкцией по разворачиванию;
- отсутствие информации о версии приложения в названии архива и даже в самом архиве (иногда помогает наличие версии в URL к архиву).

В ближайших версиях `epm play` будет выделен в подпакет `eepm-play`, что позволит убрать претензию к тому, что наличие пакета `eepm` позволяет легко установить пакеты из непроверенных источников. При этом `epm repack` (без которого `epm play` не имеет смысла) будет выделен в подпакет `eepm-repack`.

Переход к использованию GUI планируется в нескольких направлениях: приложение для установки и удаления приложений,

сайт (с API) со списком и описанием доступных приложений, интеграция с существующими менеджерами приложений, такими как `gnome-software` и `discover`, также один из вариантов доставки — создание репозитория перепакованных приложений, для которых это разрешено лицензией.

В разработке находится управление оконными средами с помощью `epm`: командами `epm desktop install / remove` можно устанавливать и удалять DE.

Полезным оказалась полная автоматизация действий по переключению используемых драйверов (`epm play switch-to-nvidia`, `switch-to-nouveau`, `switch-to-pipeware`), где учтены многие нюансы.

Добавлены сценарии для установки `Waydroid` и `flatpak`: `epm play waydroid` и `flatpak`.

В рамках добавления ведения истории всех (успешных) релизов приложения (многие вендоры на своём сайте предоставляют только последнюю версию) будет добавлены градации разрешений на распространение (по результатам анализа лицензий).

Возможны различные варианты разрешений от производителя, которые влияют на механизм доставки приложения пользователю:

- разрешение на получение дистрибутива только после регистрации на официальном сайте с выдаваемой уникальной ссылкой (получение дистрибутива не автоматизируемо);
- разрешение на скачивание по прямой ссылке только с официального сайта;
- разрешение на зеркалирование (распространение копий в Интернете);
- разрешение на распространение в составе дистрибутива;
- отсутствие запрета на перепаковку.

В ряде случаев, если присутствует такое требование от производителя, при установке приложения должен выдаваться запрос на согласие с лицензией.

Помимо проверки контрольных сумм будет добавлен статус приложения (стабильная версия, бета версия, была ли выполнена проверка работоспособности для данной системы).

Отдельного внимания требует поддержка драйверов устройств (в основном принтеров и сканеров): в настоящий момент нет чёткой свя-

зи между USB ID имеющегося устройства, названием модели и требующимся драйвером.

При этом концепция не меняется: результатом работы epm play должен быть rpm-пакет(ы), устанавливаемый в систему (и удаляемый) штатными средствами.

EPM и epm play это свободные проекты, новые сценарии можно присылать в виде pull request на https://github.com/Etersoft/eepm.

Роман Алифанов, Кирилл Уницаев
Санкт-Петербург, ООО «Этерсофт»

Проект: Ximper Linux https://ximperlinux.ru/

Ximper Linux: создание, применение и перспективы

Аннотация

Доклад посвящён Ximper Linux — современному дистрибутиву ОС на базе ALT Sisyphus, который объединяет в себе удобство, эффективность и поддержку актуального программного обеспечения. Рассмотрены основные характеристики и назначение дистрибутива, проанализированы его особенности, включая пользовательский интерфейс и частоту обновлений. Затронуты причины создания и планы развития дистрибутива.

Ключевые слова: *Ximper Linux, дистрибутив, Единый переключатель тем.*

Введение

Ximper Linux — роллинг-релиз дистрибутив, основанный на базе репозитория ALT Sisyphus. Система ориентирована преимущественно на домашнее использование, предлагая широкий набор инструментов для различных целей.

Сборка образа дистрибутива производится с помощью mkimage-profiles, как и сборка всех дистрибутивов ALT.

Основные особенности Ximper Linux

1. *Модель обновления:* В основе Ximper Linux лежит роллинг-релиз модель обновления. Система постоянно получает послед-

ние обновления из репозитория без необходимости делать полную переустановку системы. Пользователи всегда могут быть уверены, что у них установлено самое актуальное программное обеспечение и драйверы.

2. *Среда рабочего стола GNOME:* GNOME — это один из наиболее популярных и стабильных рабочих столов, предлагающий чистый и минималистичный интерфейс. Он создан с целью обеспечения максимального удобства и простоты взаимодействия с системой. Пользователи `Ximper Linux` смогут быстро адаптироваться, даже если до этого работали с другими операционными системами.

3. *Игры из коробки в PortProton:* Одной из уникальных особенностей `Ximper Linux` является встроенный PortProton — инструмент, позволяющий запускать игры, разработанные для Windows, прямо на Linux-системе. С его помощью пользователи могут наслаждаться множеством современных игр без сложных настроек и сторонних программ.

4. *Удобный пакетный менеджер EPM:* `Ximper Linux` использует EPM (Etersoft Package Manager) — простой в использовании пакетный менеджер, который облегчает установку и управление программами. С EPM даже пользователи, не обладающие глубокими знаниями Linux, смогут легко находить и устанавливать нужное ПО.

5. *Оптимизация для домашнего использования:* Система настроена таким образом, чтобы обеспечить максимальный комфорт для использования дома.

6. *Открытость:* `Ximper Linux`, как и большинство Linux-дистрибутивов, придерживается философии свободного программного обеспечения. У пользователей есть полный доступ к исходному коду системы, они могут модифицировать его под свои нужды, а также пользоваться системой бесплатно.

Преимущества Ximper Linux

- Свежие обновления благодаря роллинг-релизу;
- Поддержка игр из коробки через PortProton;

- Репозиторий Sisyphus гарантирует доступ к огромному выбору программного обеспечения;
- Интуитивно понятный рабочий стол GNOME, который обеспечивает лёгкость в использовании и подходит как для новичков, так и для опытных пользователей;
- Простой в использовании EPM, который упрощает управление программами и установку ПО;
- Оптимизация для домашнего использования;
- `Ximper Linux` — это идеальный выбор для пользователей, которые ищут стабильную, удобную и готовую к использованию систему с поддержкой игр и приложений прямо «из коробки».

Планы развития

1. *Единый переключатель тем*

   Для `Ximper Linux` разрабатывается некоторый софт, чтобы сделать пользовательский опыт гораздо удобнее. Единый переключатель тем — один из таких проектов.

Рис. 1: Единый переключатель тем.

Программа состоит из двух частей:

- *GUI.* Позволяет поменять в одном месте тему qt и kvantum для тёмного и светлого режимов.
- *Сервис.* Благодаря ему темы kvantum и GTK3 переключаются одновременно с системным режимом (тёмным/светлым) GNOME. (Сервис мониторит dbus сигналы).

В отличие от других сред рабочего окружения, GNOME отправляет два сигнала вместо одного. Для корректной работы в любых условиях сервис автоматически определяет среду и адаптируется к ней (проверено на Hyprland).

2. *Ximper Hyprland*

   Новый проект, который с начала июля находится в активной разработке. Это ответвление Ximper, созданное специально для более опытных пользователей и энтузиастов, которые любят полную настройку своей системы. Сборка основывается на динамическом оконном менеджере Hyprland и полностью ориентирована на кастомизацию, максимальную производительность и эффективность работы.

   Особенности:

   - Управление с клавиатуры: Вся система оптимизирована для управления с клавиатуры, что повышает скорость и эффективность работы. Пользователь может настроить горячие клавиши для любого действия, что сокращает время на переключение между приложениями и выполнение повседневных задач.
   - Модульная структура компонентов: Все элементы системы, включая панель задач, меню приложений и уведомления, полностью независимы и настраиваемые. Пользователь может заменить любой компонент на свой вкус, используя альтернативные решения или свои собственные разработки, что обеспечивает гибкость в создании уникальной рабочей среды.
   - Высокая степень кастомизации: Ximper Hyprland позволяет изменять практически все аспекты интерфейса: от оформления окон и панели до мелких деталей, таких как анимации и поведение элементов интерфейса. Это даёт пользователям возможность создать действительно персонализированное рабочее пространство.
   - Быстрая настройка и адаптация: Ximper Hyprland предлагает пользователю готовые шаблоны конфигураций для быстрого старта, а также инструкции и шаблоны для создания своих собственных настроек без необходимости глубоких знаний.

Дополнительные возможности:

- Для повышения удобства пользователей разрабатываются специальные скрипты, которые добавляют функции, отсутствующие в оконных менеджерах из-за их особенностей. Например, уже написан скрипт wm-xdg-autostart, обеспечивающий поддержку XDG Autostart в оконных менеджерах, что позволяет автоматически запускать программы при входе в систему.

  Эти инструменты будут изменять поведение оконных менеджеров в пользу пользовательского комфорта, при необходимости их можно будет легко выключить через файл настроек.

3. *Собственный репозиторий*

   Планируется переход `Ximper Linux` на собственный репозиторий, что позволит избежать критических ошибок. Он будет состоять из двух частей:

   - *Основная часть.* Эта часть будет протестированным резом Сизифа без ошибок при обновлении.
   - *Дополнительная часть.* Эта часть будет содержать в себе наши разработки более новых версий и важные исправления, которые, по каким-либо причинам не попали в основную часть.

4. *net-install*

   Планируется переместить выбор и установку рабочего окружения и программ в post-install установщик, это позволит уменьшить вес образа.

Станислав Богатырев
Санкт-Петербург, Yadro

Проект: FrostFS `https://frostfs.info`, `https://git.frostfs.info/`

FrostFS: свободная децентрализованная объектная СХД

Аннотация

FrostFS — это децентрализованная объектная система хранения данных, интегрированная с блокчейном Neo N3. Мы стремимся создать децентрализованную платформу, способную решать реальные, полезные на практике, задачи, которые сейчас решаются в централизованном варианте.

Ключевые особенности СХД:

– Возможность роста сети до планетарного масштаба и десятков тысяч узлов;
– Расчёт на работу в нестабильной недоверенной среде дикого интернета;
– Гибкая система политик хранения, позволяющая пользователю полностью контролировать где и как хранятся его данные;
– Поддержка разных протоколов (gRPC, S3, HTTP) доступа к объектам через протокольные шлюзы и работы Web3 приложений на традиционном Web2 стеке;
– Работа с объектами в сетевой изоляции.

Ключевые слова: *децентрализованные СХД, блокчейн, объектное хранилище.*

О проекте

FrostFS — Свободное ПО под лицензией GPLv3+ и SDK под лицензией Apache 2.0. Схемой лицензирования мы хотели предотвратить превращение основного кода проекта в проприетарное ПО и появление разных версий продукта с разной функциональностью, но обеспечить возможность использовать SDK где угодно. Комбинация пермиссивной лицензии для библиотечных компонентов и свободной лицензии для основного кода оправдала наши ожидания на практике.

Проект начался в 2018 году как заявка на грант Neo Foundation. В 2019 году мы начали разработку под именем NeoFS, а в 2022 нам пришлось сделать форк самих себя и продолжить разработку под именем FrostFS.

После проблем с GitHub мы перешли на self-hosted инфраструктуру на Forgejo. Сейчас все исходные коды и артефакты релизов доступны на `http://git.frostfs.info`.

В 2022 году вышел коммерческий продукт Yadro Tatlin.Object, базирующийся на FrostFS. В продукте используется свободный код с полным соблюдением лицензии, а все изменения происходят напрямую в основных репозиториях проекта.

Сейчас мы работаем над запуском и тестированием публичной сети. Помимо объектной СХД, мы позволяем использовать DNS с хранением данных в блокчейне и слой HTTP-шлюзов, позволяющих обеспечить доставку данных. Все элементы платформы являются СПО и могут быть развёрнуты на собственной инфраструктуре. Мы хотим прийти к набору децентрализованных сервисов для разработки и запуска приложений, способных заменить централизованные облачные решения для подавляющего большинства современных задач от хранения данных, до их обработки.

Модель данных

На уровне родного протокола FrostFS, данные представлены в виде неизменяемых объектов, т.е. совокупности данных и метаданных.

Объекты объединяются в контейнеры — структуры, состоящие из политики хранения и дополнительных атрибутов в формате Ключ-Значение.

Из хэша структуры контейнера и хэша заголовка объекта формируется адрес объекта. Таким образом, адрес зависит от содержимого (Content Addressed Storage) и подмена данных при передаче невозможна.

Сеть FrostFS поддерживает и постоянно обновляет список активных узлов хранения сети — карту сети. Узел хранения представлен идентификатором его ключа и набором атрибутов.

С помощью политики хранения пользователь может гибко настроить правила, по которым его данные будут располагаться в сети. Правила можно писать на SQL-подобном языке. Политика применяется к карте сети и на выходе детерминированно получается вектор узлов

хранения, пригодных для размещения объекта, отсортированный в порядке уменьшения вероятности найти объект на узле.

Поиск и размещение объекта происходят по одному и тому же алгоритму, что позволяет без центрального индекса или метабазы находить сохранённые объекты, обладая адресом и актуальной или прошлой версией карты сети.

Архитектура

Во FrostFS используется блокчейн, как распределённая реплицированная база данных. Можно считать, что смарт-контракты в блокчейне это аналог таблиц, а методы контрактов — хранимые процедуры. Блокчейн используется как источник дискретного времени и хранилище небольшого объёма управляющей информации.

Управляющую функцию выполняет небольшое количество Узлов Внутреннего Кольца (Inner Ring Nodes). Опираясь на блокчейн, они подтверждают статусы узлов хранения, формируют карту сети, утверждают создание контейнеров и следят за соблюдением политик хранения другими узлами. Для возможности контролировать корректность хранения данных реализован алгоритм ZKP на гомоморфных хэшах, с оптимизацией на goasm, для достижения скоростей записи на HDD.

Непосредственно хранением данных занимаются Узлы Хранения (Storage Nodes). Они образуют p2p сеть и принимают запросы на размещение объектов и доступ к ним. Каждый узел обеспечивает соблюдение политики хранения, заданной пользователем. фоновый процесс опрашивает узлы, на которых должны храниться оставшиеся копии, проверяет и передаёт при необходимости копию объекта.

Родной протокол FrostFS строится на базе gRPC с сообщениями в формате protobuf. Все узлы и клиенты общаются по одному и тому же протоколу в p2p манере.

Для работы с уже существующими приложениями в FrostFS есть протокольные шлюзы s3 и HTTP. Проект стремится предоставлять доступ к одним и тем же объектам по разным протоколам одновременно, сохраняя контроль правил доступа. Гарантии по безопасности и контролю доступа проходят и через протокольные шлюзы. Можно даже на уровне HTTP BearerToken предать подписанные ключом владельца контейнера правила доступа и они будут переданы с запросом и исполнены на нижнем уровне родного протокола узлами хранения.

Кроме протокольной трансляции шлюзы добавляют специфичную для конкретной предметной области функциональность. Так HTTP шлюз может формировать списки объектов по определённым префиксам в виде HTML страницы, как это делают традиционные веб-серера или выгружать группы объектов в zip-архиве.

Такой подход позволяет применять при разработке децентрализованных приложений уже существующие приёмы и наборы инструментов.

Результаты

Сейчас проект разрабатывает Свободное ПО и коммерческий продукт на его базе, взаимодействует со студентами в рамках учебных курсов, практики и руководства ВКР. Мы продолжаем исследования проблем децентрализованных систем и публикуем результаты в научных статьях. Достигнут паритет по функциональности с другими решениями, при этом в децентрализованной и недоверенной среде.

Антон Политов
Смоленск, филиал ФГБОУ ВО «НИУ «МЭИ» в г. Смоленске

Проект: ALT Gnome `https://vitepress.dev/`, `https://alt-gnome.wiki/`

VitePress: создание современных библиотек знаний для операционных систем семейства Linux в стеке Vue3 и Markdown

Аннотация

В докладе рассмотрен фреймворк VitePress [1] — современный инструмент для создания документации, который позволяет упростить и ускорить разработку. Будут рассмотрены ключевые составляющие VitePress, такие, как Vue и Vite, возможности стека, поддержка написания собственных компонентов на Vue. В рамках доклада будет представлен реальный пример использования VitePress на базах знаний ALT. Будет затронуто управление репозиторием и взаимодействие с сообществом документации, а также возможности общения с апстримом для улучшения функционала и совместной работы над проектом.

Ключевые слова: *VitePress, ALT Gnome Wiki.*

VitePress как современный инструмент для создания документации

VitePress — это современный статический генератор сайтов, использующий открытую лицензию MIT и специально разработанный для создания баз знаний. Он основан на Vite, высокопроизводительном инструменте сборки, и использует Vue.js для создания интерактивных компонентов. VitePress предлагает интуитивно понятный интерфейс для пользователю и обширные возможности организации и управления контентом для разработчика.

Ключевые особенности и преимущества использования VitePress

- **Лёгкость в использовании**: Простой процесс конфигурации и сборки с использованием современных инструментов CI/CD, интуитивно понятный язык разметки Markdown для написания статей и парсер Markdown-it и его расширения делают VitePress доступным инструментом для разработчиков с разным уровнем опыта;
- **Гибкость:** Легко настраиваемые интерфейс и темы позволяют адаптировать внешний вид под нужды проекта. Также VitePress предоставляет адаптивную тему по умолчанию;
- **Использование Vue.js**: Возможность интеграции компонентов Vue позволяет создавать динамичные и интерактивные элементы в документации, значительно расширяя её функциональность;
- **Сообщество и открытый исходный код**: Открытое развитие привлекает сообщество разработчиков и авторов, которые могут вносить свой вклад, исправлять ошибки и добавлять новые функции в продукт.

Удобство управления проектами на VitePress

Редактирование и хранение данных в формате Markdown вместо классической базы данных позволяет просматривать и редактировать необходимые данные, не используя дополнительные инструменты, а

благодаря использованию Vite можно локально запускать среду для разработки и отладки VitePress.

Также данный фреймворк предоставляет свободу выбора организации CI/CD для тестирования и поставки (деплоя) проекта в зависимости от инфраструктуры. К примеру, можно использовать платформы GitHub Actions или конвейеры GitFlic, чтобы упростить задачи проверки внесённых сообществом и командой изменений.

Статическая генерация VitePress заметно упрощает размещение проекта в рамках различных инфраструктур — это могут быть как сервисы наподобие GitHub Pages, так и собственный веб-сервер NGINX, Apache или любой другой.

Практическое использование VitePress на базах знаний систем семейства «Альт»

Яркими примерами практического использования VitePress являются базы знаний по системам семейства «Альт» [2, 3, 4, 5], которые в полной мере раскрывают особенности VitePress.

Во многом благодаря простоте написания контента, проект ALT Gnome Wiki активно развивается и дополняется новыми статьями, которых, благодаря усилиям сообщества, уже более 290. За счёт поддержки Vue активно внедряются новые компоненты, написанные как участниками проекта, так и сообществом. Например, история изменений статей и улучшение читабельности (плагины набора Nólëbase Integrations), боковой бар приложений, динамическая генерация страницы участников и др.

Простота использования и поддержки позволяет в кратчайшие сроки создавать новые проекты, ярким примером чего служит ALT Mobile Wiki и ALT Packaging Guide (Vue), созданные силами нескольких человек и уже сейчас помогающие пользователям.

Взаимодействие с сообществом разработки VitePress

Разработчики VitePress и Vue.js активно развивают собственный проект с ростом его популярности. С каждым днём вводится все больше новых функций, разработчики и участники сообщества активно делятся решением проблем и задач, готовы помочь с вопросами по использованию VitePress и внедрению любого функционала.

Взаимодействие с сообществом разработки компонентов

Поддержка компонентов Vue в VitePress позволяет не только заниматься написанием собственных расширений, но и использовать наработки сообщества — одним из ярких представителей сообщества является команда Nólëbase.

В частности набор плагинов Nólëbase Integrations, а также активное взаимодействие с разработчиком напрямую упростили процесс разработки упомянутых ранее баз знаний по системам семейства «Альт».

Литература

[1] VitePress — `https://vitepress.dev/` (Репозиторий: `https://github.com/vuejs/vitepress`)

[2] ALT Regular Gnome Wiki — `https://alt-gnome.wiki/` (Репозиторий: `https://github.com/OlegShchavelev/ALTGnomeWiki`)

[3] ALT KDE Wiki — `https://alt-kde.wiki/` (Репозиторий: `https://github.com/OlegShchavelev/ALTKDEWiki`)

[4] ALT Mobile Wiki — `https://altmobile.org/` (Репозиторий: `https://github.com/OlegShchavelev/ALTMobileWiki`)

[5] ALT Packaging Guide (Vue) — `https://sokolovvaly.github.io/alt-packaging-guide-vue/` (Репозиторий: `https://github.com/SokolovValy/alt-packaging-guide-vue`)

[6] Nólëbase Integrations — `https://nolebase-integrations.ayaka.io/pages/en/` (Репозиторий: `https://github.com/nolebase/integrations`)

Владимир Васьков
г. Москва, ALT Gnome

Проект: Vala https://vala.dev/

Vala — современный язык программирования GNOME

Аннотация

Доклад посвящён языку программирования Vala. В докладе будут рассмотрены его возможности, стандартная библиотека и принцип работы «под капотом». Будет презентовано в каких проектах Vala используется.

Ключевые слова: *разработка, GNOME, язык программирования, Vala.*

Что такое Vala?

Vala — компилируемый объектно-ориентированный язык программирования с синтаксисом Java/C#. Однако в это же время Vala не совсем язык в стандартном понимании. Это компилятор в более низкоуровневый код на C, который можно скомпилировать в машинный код желаемым C-компилятором. В качестве стандартной библиотеки язык использует обширную библиотеку GLib, в частности GObject для реализации ООП и механизма управления ссылками, дополняя их:

- асинхронными функциями в виде корутин, что позволяет не блокировать основной поток;
- лямбда-функциями;
- строками в виде отдельного класса в C-строки — массив символов.

Какие библиотеки есть у Vala?

Vala может использовать любые C-библиотеки, для которых написаны VAPI файлы. VAPI — инструкция трансляции C-кода в Vala. В файле необходимо описать C-код в Vala стиле, дополняя в аттрибутах различную информацию: название функции или структуры в C,

описание аргументов функции, добавляя ключевые слова out или ref или добавляя значение по умолчанию. Есть возможность автоматической генерации VAPI, используя GObject Introspection файл. Vala имеет VAPI для всех библиотек стека GNOME. Также остальные библиотеки, которые используют GObject, включая те, что написаны на других языках с использованием языковых привязок, можно использовать после генерации VAPI (интероперабельность).

Для каких целей Vala был создан?

Написание программы на C с использованием GObject налагает дополнительные трудности в виде написания большого количества шаблонного кода для реализации ООП. Языковые привязки к GObject могут иметь проблемы со своей системой управления памятью или более низкой производительностью относительно C. Компилятор Vala оставляет управление памятью за собой на этапе компиляции, предоставляя читаемый и лёгкий синтаксис, из-за чего время разработки сокращается.

Где Vala используется?

На Vala можно писать консольные утилиты:

- valac — компилятор Vala;
- vala-language-server — языковой сервер Vala;
- vala-lint — утилита для форматирования Vala-кода.

Графические приложения:

- GNOME Clocks — приложение часов, входящее в GNOME Core;
- Tuba — клиент matrix;
- Planify — приложения для задач;
- Foldy — приложение для управления папками приложений;
- Baobab — приложение для анализа занятого места на компьютере;
- Black Box — приложение виртуального терминала;
- Cassette — неофициальный клиент Яндекс Музыки.

Библиотеки:

- libgranite — графическая библиотека, разработанная командой Elementary OS;
- libgee — библиотека с высокоуровневыми коллекциями;
- libgxml — GObject надстройка над libxml2.

Это не все проекты на Vala и их количество постоянно растёт.

Семен Фомченков
Обнинск, ИАТЭ НИЯУ МИФИ

Проект: ALT Gnome https://gitlab.gnome.org/GNOME/libadwaita

Libadwaita — инструмент для создания современного UI

Аннотация

Libadwaita является современной библиотекой для разработки графических интерфейсов приложений в среде GNOME. В эпоху стремительного развития технологий пользователи ожидают от приложений не только функциональности, но и современного, адаптивного интерфейса с единообразным пользовательским опытом (UX). Libadwaita предоставляет разработчикам инструменты для создания интерфейсов, придерживаясь стандартов дизайна GNOME. Стандартизация пользовательских интерфейсов помогает создать единое и предсказуемое взаимодействие, делая работу с приложениями проще и интуитивнее.

Ключевые слова: *GNOME, Адаптивность, UI/UX.*

Libadwaita как инструмент для создания современного UI

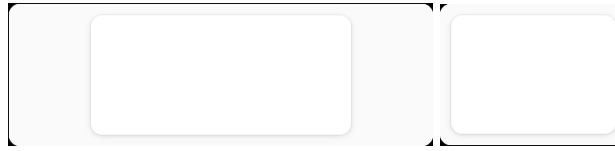
Libadwaita — это мощный инструмент для разработчиков, которые хотят создавать современные и адаптивные приложения. Она выросла из библиотеки Libhandy, которая когда-то была создана для работы с адаптивными интерфейсами в GTK 3. С появлением GTK 4 Libadwaita получила новые возможности, набор виджетов и стилей, соответствующих последним рекомендациям по дизайну GNOME Human Interface Guidelines (HIG). Эта интеграция не просто помогает соблюдать единый стиль, но и улучшает общее впечатление от работы с приложениями. Она облегчает разработку интерфейсов, обеспечивая единообразие внешнего вида и поведения приложений.

Адаптивный интерфейс и стандартизированный UX в Libadwaita

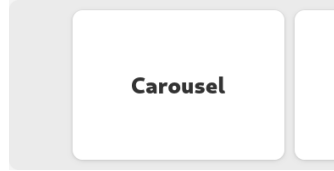
Одна из самых полезных особенностей Libadwaita — это автоматическое подстраивание интерфейса под разные размеры и ориентации экранов. Это существенно облегчает жизнь разработчикам: они могут сосредоточиться на функциональности, не думая о том, как приложение будет выглядеть на разных устройствах. Современные виджеты и стили делают приложения не только удобными, но и эстетически приятными. На данный момент Libadwaita предлагает широкий набор виджетов для создания адаптивных интерфейсов, рассмотрим несколько из них:

- `AdwNavigationSplitView` — позволяет создавать многостраничные интерфейсы, которые автоматически переключаются между режимами стека и панели в зависимости от размера экрана.

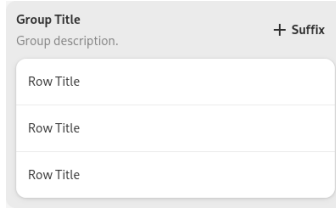
- `AdwOverlaySplitView` — обеспечивает адаптивные боковые панели, которые могут скрываться или отображаться в зависимости от доступного пространства.
- `AdwClamp` — ограничивает максимальную ширину содержимого, улучшая читаемость на широких экранах.



- `AdwCarousel` — предоставляет возможность создания прокручиваемых списков элементов с поддержкой жестов, что особенно полезно на сенсорных устройствах.



- `AdwViewSwitcher` и `AdwViewStack` — облегчают создание интерфейсов с переключаемыми представлениями, адаптирующимися под размер экрана.



- `AdwActionRow` и `AdwPreferencesGroup` — помогают создавать адаптивные страницы настроек с единообразным стилем и поведением.

Использование этих виджетов обеспечивает комфортное взаимодействие пользователя с приложением независимо от устройства. Стандартизированный UX достигается благодаря строгому следованию GNOME HIG, что упрощает освоение новых приложений и повышает удовлетворённость пользователей средой GNOME.

Развитие фреймворка Libadwaita

История развития Libadwaita показывает, как сообщество стремится к совершенствованию инструментов. От решения конкретных задач адаптивности в Libhandy до более универсального инструмента в Libadwaita — этот путь демонстрирует важность эволюции, когда технология должна отвечать требованиям как разработчиков, так и пользователей. На текущий момент основные направления развития фреймворка включают:

- **Интеграция современных дизайнерских паттернов**: постоянное обновление виджетов и стилей в соответствии с актуальными рекомендациями GNOME HIG;
- **Расширение функциональности**: добавление новых адаптивных виджетов и улучшение существующих для повышения гибкости разработки;
- **Оптимизация производительности**: использование возможностей GTK 4 для ускорения работы интерфейсов и снижения потребления ресурсов;
- **Поддержка новых платформ**: обеспечение корректной работы приложений на различных устройствах, включая мобильные и сенсорные экраны;
- **Совместимость и миграция**: предоставление инструментов и документации для облегчения перехода разработчиков с предыдущих версий GTK и связанных библиотек на Libadwaita.

В итоге, Libadwaita — это современный инструмент, который значительно облегчает разработку и улучшает пользовательский опыт. Её будущее выглядит многообещающим: сообщество активно работает над новыми функциями и улучшениями. Стандартизация, адаптивность и участие сообщества — это основа успешной разработки сегодня.

Алексей Андреев, Максим Костерин, Кирилл Чувилин
Санкт-Петербург, Ярославль, Москва, ООО «Открытая Мобильная Платформа», ООО «ФРУКТ»

Проект: auroradeployqt `https://gitlab.com/omprussia/qt`

Поддержка целевых платформ в фреймворке Qt на примере ОС Аврора

Аннотация

Кросс-платформенный фреймворк Qt позволяет создавать приложения для различных окружений: операционных систем для персональных компьютеров и мобильных устройств, WebAssembly. Это достигается благодаря слою адаптации, который обеспечивает взаимодействие публичных классов Qt с программными интерфейсами, предоставляемыми целевой платформой. В докладе рассматривается опыт портирования Qt на публичный API операционной системы Аврора, описываются необходимые для изменения в компонентах слоя адаптации и их взаимодействие с системным API. Доклад будет интересен как разработчикам операционных систем и устройств, так и прикладным разработчикам, которые хотят ознакомиться с устройством системных API.

Ключевые слова: *qt, auroraos, linux, программные интерфейсы операционных систем.*

Qt — это мультиплатформенный фреймворк, предоставляющий широкий набор библиотек и инструментов разработки ПО [4]. Приложения на Qt используются в различных окружениях: персональные компьютеры, мобильные устройства, медицинское оборудование, автомобильные системы, браузеры. Примеры ПО, написанного на Qt: KDE, VLC, Telegram, Virtual Box, Cura и др. Многие дистрибутивы Linux используют Qt как основной фреймворк, в том числе Ubuntu, Fedora, Red Hat и отечественные Astra Linux, ROSA, ALT Linux. Не исключением является и мобильная операционная система Аврора [1], разрабатываемая компанией «Открытая мобильная платформа».

Для разработки прикладного ПО фреймворк Qt предоставляет платформонезависимый API. Это позволяет иметь общую кодовую базу для версий приложения под различные целевые платформы:

Windows, Android, macOS, iOS, дистрибутивы Linux, ОС Аврора. Однако для взаимодействия с каждой платформой сами библиотеки Qt используют системные API, которые могут различаться для разных платформ. Набор связующего ПО, который позволяет запустить Qt на конкретной платформе называют слоем адаптации или, для краткости, адаптацией Qt под эту платформу.

Qt состоит из большого количества отдельных библиотек, которые сгруппированы по модулям [5]: базовые модули, мультимедиа, работа с датчиками, работа с сетью, работа с базами данных и др. Адаптация может выполняться для модулей независимо.

Поддержка платформы может быть реализована на стороне самого фреймворка Qt [6]. Это означает, что работы по адаптации Qt включены в состав основной ветки исходных кодов Qt, разработчики могут использовать уже готовые модули. В случае, когда целевая платформа не поддержана в составе исходных кодов Qt, разработчики, желающим предоставить или использовать фреймворк на такой платформе, могут провести работы по адаптации самостоятельно.

Среди открытых проектов можно обнаружить адаптации Qt для Tizen [7], Haiku [8], managarm [9], SerenityOS [10], OpenHarmony [11].

Мы выполнили открытую адаптацию Qt для ОС Аврора [12]. Могут возникнуть два вопроса:

- В составе ОС Аврора уже есть библиотеки Qt. Зачем потребовалось делать новую адаптацию?

  По ряду причин в составе ОС Аврора сейчас используются библиотеки Qt версиии 5.6. Несмотря на то, что они вполне применимы для разработки прикладного ПО, версия уже изрядно устарела. Это приводит к тому, что разработчикам становится сложнее поддерживать общую кодовую базу, если их приложение на другой платформе использует более новую версию Qt.

- ОС Аврора близка к дистрибутивам Linux для ПК. В чём особенность портирования Qt для неё?

  В составе ОС Аврора можно найти многие распространённые библиотеки, такие как GStreamer, systemd, и др. Адаптация Qt в составе ОС Аврора их и использует. Не смотря на это, непосредственное использование системных библиотек в прикладном ПО ограничено механизмами безопасности. Для сторонних приложений ОС Аврора предоставляет собственный API [13]. И получается, что адаптация Qt для ОС Аврора — это новая

задача, заключающая в использовании публичного API ОС Аврора для реализации функций, необходимых интерфейсам Qt.

Как и для многих других проектов, выстраивание взаимодействия между вносящими вклад участниками и передача наработок в основную ветку — отдельная непростая задача. Более простой организационно шаг: начать адаптацию с изменений в отдельном репозитории.

Для этого требуется подготовить связку между абстрактными классами, предлагаемыми как универсальные со стороны Qt, и API, предоставляемым платформой. Чтобы это реализовать, понадобится сверить наборы доступных и требуемых функций и проработать конвертацию форматов данных. Сложность может заключаться в том, что эти характеристики могут изменяться как между различными версиями Qt, так и между различными выпусками целевой платформы. При этом, придётся изучать исходный код Qt [14], поскольку документация не покрывает по его приватные классы.

В ходе проведения адаптации мы выяснили, какие функции должна предоставлять платформа для реализации отдельных модулей Qt. Кроме того, разработали инструмент auroradeployqt. Который по аналогии с другими утилитами *deployqt позволяет скопировать в поставку приложения необходимые зависимости: библиотеки модулей Qt и QML-компоненты. А также проверяет доступность для линковки и запуска с учётом механизмов валидации пакетов ОС Аврора.

Актуальный статус разработки и руководство по использованию доступны на странице [15].

Литература

[1] Аврора — первая российская мобильная операционная система, URL: `https://auroraos.ru` (дата посещения 12.09.2024).

[2] Открытая мобильная платформа — официальный сайт, URL: `https://www.omp.ru` (дата посещения 12.09.2024).

[3] Qt: Cross-platform software development for embedded & desktop, URL: `https://qt.io` (дата посещения 12.09.2024).

[4] Qt: Development Framework for Cross-platform Applications, URL: `https://www.qt.io/product/framework` (дата посещения 12.09.2024).

[5] Qt Documentation: Modules, URL: `https://www.qt.io/product/framework` (дата посещения 12.09.2024).

[6] Qt Documentation: Supported Platforms, URL: https://doc.qt.io/qt/supported-platforms.html (дата посещения 12.09.2024).

[7] Qt Wiki: Tizen (legacy) URL: https://wiki.qt.io/Tizen (дата посещения 12.09.2024).

[8] Haiku Community: The Next Generation of Qt, URL: https://discuss.haiku-os.org/t/the-next-generation-of-qt-6-is-now (дата посещения 12.09.2024).

[9] GitHub: Patches and build scripts to build a managarm distribution, Qt, URL: https://github.com/managarm/bootstrap-managarm/tree/master/patches/qt6 (дата посещения 12.09.2024).

[10] GitHub: Port Qt to SerenityOS, URL: https://github.com/SerenityOS/serenity/issues/152 (дата посещения 12.09.2024).

[11] Gitee: building project with Qt for OpenHarmony DevEco, URL: https://gitee.com/openharmony-sig/qt (дата посещения 12.09.2024).

[12] GitLab: Open Mobile Platform, Qt related repositories, URL: https://gitlab.com/omprussia/qt/ (дата посещения 12.09.2024).

[13] Документация ОС Аврора: Публичные API, URL: https://developer.auroraos.ru/doc/software_development/reference/public_api (дата посещения 12.09.2024).

[14] GitHub: Official mirror of the Qt Git repositories, URL: https://github.com/qt (дата посещения 12.09.2024).

[15] GitLab: AuroraDeployQt, руководство по быстрому началу работы, URL: https://gitlab.com/omprussia/qt/auroradeployqt/-/wikis/Quickstart-Guide (дата посещения 12.09.2024).

Дмитрий Лапшин, Даниил Маркевич, Константин Звягин, Кирилл Чувилин
Москва, ООО «Открытая мобильная платформа»

Проект: Картографический инструментарий MFW
`https://gitlab.com/omprussia/examples-extra/MfwMap`

Картографический инструментарий MFW

Аннотация

MFW — это набор открытых программных библиотек, инструментов и примеров для решения задач картографии и навигации. Поводом для его создания послужили запросы от разработчиков прикладного ПО для ОС Аврора на функции, которые не поддержаны в фреймворке Qt. MWF обеспечивает работу с векторными картами, включая маштабирование, вращение, стилизацию, а также геокодирование, навигацию и ведение по маршруту. В докладе представлены доступные функции, архитектура и особенности реализации.

Ключевые слова: *OS Aurora, Карты, разработка приложений.*

ОС Аврора — это операционная система для мобильных устройств, разрабатываемая компанией «Открытая мобильная платформа». Важную часть её целевой аудитории составляют корпоративные заказчики. Работа с картами является распространённым запросом для пользователей этого сегмента. Зачастую возникает необходимость внедрить соответствующие функции в ПО, которое разрабатывается для конкретного проекта. Для этого требуется API со стороны ОС и программные библиотеки. Основные требования к таким библиотекам:

- набор API должен покрывать отрисовку карт, управление отображениями карт, прямое и обратное геокодирование, построение и ведение по маршрутам;
- должны поддерживаться векторные карты;
- должна поддерживаться работа как с данными онлайн, так и в офлайн режиме;
- должны поддерживаться актуальные версии ОС Аврора;
- формат поставки библиотек должен позволять использование в составе коммерческих программных продуктов.

ОС Аврора предоставляет API для получения координат устройств и функции для работы с картами на основе Qt Location[1]. Однако, для использования доступны только растровые тайлы, и не обеспечены задачи геокодирования.

Были проанализированы также другие решения, и выявлены их ограничения:

- OSM Scout server[2]

  Ограничения: сложная интеграция со сторонними решениями для ОС Аврора, не предоставляет API для работы с тайлами.

- Navitel [3]

  Ограничения: стороннее решение, не предоставляющее средства интеграции.

- Yandex [4]

  Ограничения: отсутствует API для ОС Аврора.

Поскольку доступны варианты обладали существенными ограничениями, было принято решение предоставить новое решение. Набор библиотек MFW предоставляет следующие основные компоненты:

- *MfwMap* — основная библиотека[5].

  В качестве основы компонента отображения векторных карт в ней используется библиотека MapLibre GL Native, для которой реализованы интерфейсы Qt Quick.

  Основные озможности *MfwMap*:

  – подключение карт и API для управления ими, карты легко интегрируются в код пользовательских приложений;

  – работа в режиме онлайн/оффлайн;

  – векторные (плюсы: компактность и хорошая масштабируемость) и растровые карты;

  – геокластеризация;

  – аппаратный рендеринг.

[1] `https://developer.auroraos.ru/doc/software_development/guides/webtiles`
[2] `https://rinigus.github.io/osmscout-server/en/`
[3] `https://navitel.ru/ru`
[4] `https://yandex.ru/dev/mapkit/doc/ru`
[5] `https://gitlab.com/omprussia/libraries/mfw-map`

- *MfwOfflineRouting*[6] — библиотека для построения маршрутов.
  Основные возможности *MfwOfflineRouting*:
  - построение маршрута по двум или нескольким точкам;
  - получение необходимой информации для ведения по маршруту;
  - обновление ранее полученного объекта маршрута;
- *MfwOfflineGeocoding*[7] — библиотека для геокодинга.
  Основные возможности *MfwOfflineGeocoding*:
  - получение адреса по координатам;
  - получение координат по адресу;
  - автоматическое получение подсказок при наборе адреса.
- *MfwExportScripts*[8] — docker-образ со скриптами генерации оффлайн инофрмации.
  Основные возможности *MfwExportScripts*:
  - генерация данных для оффлайн-карт;
  - генерация данных для построения маршрутов;
  - генерация данных для геокодинга.

В совокупности разработанные компоненты предоставляют необходимые библиотеки, инструменты, документацию, а также примеры как для разработки отдельных картографических решений, так и для включения необходимых функций в состав специализированного ПО. Для удобства включения в состав сторонних приложений MWF предоставляется в виде открытого исходного кода с разрешительными лизцензиями.

[6] `https://gitlab.com/omprussia/libraries/mfw-offline-routing`
[7] `https://gitlab.com/omprussia/libraries/mfw-offline-geocoding`
[8] `https://gitlab.com/omprussia/tools/mfw-export-scripts`

Давид Султаниязов
Санкт-Петербург, Санкт-Петербургский государственный институт кино и телевидения

Проект: ALT Gnome

Прогрессивные веб-приложения: конвергентное дополнение нативным программам в операционных системах семейства Linux

Аннотация

В докладе исследуется роль прогрессивных веб-приложений (PWA) как дополнения к традиционным нативным программам в системах семейства Linux. Рассматривается использование веб-приложений в рабочих окружениях, их отличия по сравнению с десктопными и мобильными приложениями, а также особенности их внедрения в операционные системы семейства «Альт». Будет рассмотрена тема интеграции в корпоративные процессы и какие инструменты и фреймворки поддерживают создание и развертывание таких приложений в Linux-системах.

Доклад охватывает различные варианты распространения веб-приложений и их интеграцию в существующие системы дистрибуции программного обеспечения. В завершение будет проведён сравнительный анализ жизненных циклов десктопных, мобильных и веб-приложений в рамках различных платформ, что позволит выявить ключевые отличия в их разработке, обновлении и поддержке. Это поможет понять, как PWA могут стать удобным и эффективным решением для пользователей Linux и улучшить взаимодействие между веб и нативными приложениями.

Ключевые слова: *прогрессивные веб-приложения, конвергентность, нативные приложения.*

Введение в концепцию и преимущества PWA

Прогрессивные веб-приложения (PWA) представляют собой технологию создания сайтов с доступом к функциям операционной системы (например: отправка уведомлений, получение доступа к геолокации, WebGL), предлагая пользователям возможность использовать приложение как нативное. Они обеспечивают улучшенную производительность за счёт технологии кэширования и работы в офлайн-режиме,

возможность установки на рабочий стол без необходимости традиционной загрузки через магазин приложений, а также кроссплатформенность, что упрощает разработку и дальнейшую поддержку единого приложения на разных устройствах и операционных системах, что делает PWA хорошим методом создания разнонаправленных решений, так как они могут обеспечить быстрый доступ к функциям и информации при минимальных затратах на разработку и поддержку.

PWA в корпоративных и частных пространствах

Внедрение PWA в рабочие процессы может значительно улучшить общую доступность и удобство использования корпоративных систем. PWA способны интегрироваться в существующие корпоративные среды, предлагая пользователю постоянный доступ к необходимым инструментам и данным. Однако внедрение PWA может вызывать трудности в связи с необходимостью интеграции с уже существующими корпоративными сервисами, не готовыми предоставлять необходимые данные PWA, что может привести к проблемам с безопасностью и конфиденциальностью данных при нахождении вне корпоративного контура.

Использование PWA в частной жизни может облегчить доступ к необходимому набору программного обеспечения и даёт возможность использовать веб-сайт как приложение. Можно заменить некоторые нативные приложения на их PWA-версию в угоду быстрого и беспрепятственного обновления.

Примером уже реализованного свободно распространяемого PWA в докладе будет представлен сервис Penpot [1] — инструмент проектирования с открытым исходным кодом для совместной работы над дизайном и кодом, адаптированный под технологии PWA

Создание и интеграция PWA в операционные системы, основанные на ядре GNU/Linux

Операционные системы семейства Linux подходят для создания PWA, предоставляя свободные инструменты (VSCodium, Zed, Neovim и др.), и использования открытых фреймворков (React Native, Vue.js, Svelte, Flutter, Django и т. д.), что позволяет разрабатывать и тестировать веб-приложения в режиме оперативной разработки (Live Development Mode). При интеграции PWA в операционные системы

в большинстве случаев не требуется знать специфику этих систем в контексте адаптации приложения под различные устройства — данные о системе зачастую предоставляются фреймворками. Из-за разнообразия операционных систем на базе ядра GNU/Linux и поставляемых этими системами программного обеспечения для работы PWA (браузеры), могут возникнуть некоторые сложности при адаптации приложения к конкретной версии браузера.

В рамках доклада будет сделан обзор процесса перевода сайт, реализованного с помощью фреймворка React, в веб-приложения. Для примера будет взят сайт с открытым исходным кодом, предоставляющий простые инструменты для различных задач [2]

Сравнительный анализ жизненных циклов десктопных, мобильных и веб-приложений

Жизненные циклы приложений могут значительно различаться в зависимости от их типа. Десктопные приложения требуют установки и регулярных обновлений пакетов, тогда как мобильные приложения могут обновляться через магазины приложений. PWA, в свою очередь, обновляются на серверной стороне, что позволяет пользователям получать последние версии без необходимости устанавливать обновления на стороне клиента. Понимание этих различий важно для эффективного управления приложениями, так как они влияют на способы обновления, поддержку и взаимодействие с пользователями.

Фреймворки и инструменты для создания и развёртывания PWA: ретроспективы и текущие тренды

С момента появления прогрессивных веб-приложений в 2015 году, инструменты и фреймворки, такие как Service Workers, Web App Manifests и Workbox, претерпели значительные изменения, отражая рост и развитие этой технологии. Ранние инструменты (полифиллы, специализированные библиотеки для кэширования) были важны для обеспечения базовой поддержки, тогда как современные фреймворки (Angular, React, Vue.js) и утилиты для тестирования и сборки (Lighthouse, Webpack) теперь играют ключевую роль в создании и развёртывании PWA, обеспечивая более глубокую интеграцию, высокую производительность и удобство разработки.

Литература

[1] Сервис penpot URL: https://github.com/penpot/penpot

[2] Исходный код к сайту x1z53.ru URL: https://gitverse.ru/x1z53/x1z53.ru

Андрей Савченко
Москва, Базальт СПО
Проект: ALT Mobile https://www.altlinux.org/ALT_Mobile

ALT Mobile

Аннотация

В докладе делается обзор проекта ALT Mobile, позволяющего использовать обычный Linux на мобильных устройствах. Обсуждается развитие проекта, поддержка и особенности использования на новых типах устройств: планшетах и игровых консолях.

Ключевые слова: *СПО, мобильные устройства, Sisyphus, Alt.*

Введение

В то время, как Linux покорил сервера и достойно представлен на десктопах, ситуация на мобильных устройствах куда скромнее. Android хоть и основан на ядре Linux, но ключевые компоненты системы закрыты, либо не управляемы сообществом и обладают сомнительной легальностью.

Поэтому всё большее развитие получают альтернативные проекты, позволяющие использовать современный телефон без проприетарного ПО. Команда ALT Linux Team является частью этого движения в рамках проекта ALT Mobile [1].

Аппаратная платформа

Основным требованием к совместимым устройствам является требование необходимости и достаточности свободного ПО для базовой функциональности устройства. Исключения могут быть сделаны для

прошивок периферийных устройств, поскольку это обособленное оборудование. При этом драйвера ядра должны быть открыты во исполнение GPLv2.

Это жёсткие ограничения, но именно они позволяют обеспечить свободу и безопасность пользователей. На данный момент поддерживаются устройства на базе СнК `rk3399`, `rk3566`, `imx8mm`, но ведётся работа и над поддержкой других СнК.

В рамках проекта доступны образы [2] для телефонов, планшетов и игорвых консолей, а так же для виртуальных машин архитектур `x86_64`, `aarch64` и `riscv`.

Графический интерфейс

На текущем этапе развития проекта основным графическим окружением выбрана оболочка Phosh (phone shell), основанная на библиотеках Gnome, в частности, технологии адаптивного интерфейса `libadwaita` [3].

В ходе портирования возникли нетривиальные проблемы, связанные с использованием в Альте механизма `tcb` [4] вместо обычного `shadow`, которые были успешно решены путём доработки [5] `phosh` и `acountsservices` [6] в Gnome.

Ведётся активная работа с апстримом по доработке оболочки и ряда приложений. За год большая часть основных компонент была адаптирована.

Прикладное ПО

Представлен широкий спектр прикладного ПО как для связи (`gnome calls`, `chatty`, `contacts`, `telegram`), так и для повседневных задач: карты, мультимедиа (`livi`, `mpv`, `amberol`, `lollypop`), работа с камерой (`megapixels`, `cheese`), браузеры (`chromium`, `firefox`), почта (`geary`), календари, калькулятор, погодный виджет и т. п.

При необходимости можно использовать механизм контейнерной изоляции `Waydroid` для запуска Android приложений. Но этот механизм не рекомендуется вследствие худшего контроля над ним и закрытости типовых приложений. Грубо говоря, ситуация аналогична использованию `wine` для запуска приложений Windows в Linux.

Доменная интергация

Реализована доменная интеграция, что является основой для MDM. Устройствами можно удалённо управлять, при этом устройства на ALT Mobile могут быть как управляемыми, так и управляющими.

Проблемы

Основной проблемой является нарушение GPLv2 ядра со стороны многих вендоров, когда код ядра и всех его драйверов не публикуется. Это ограничивает возможности по портированию, но прогресс в данном вопросе наблюдается.

Есть системная проблема, когда системно значимые приложения существуют только в закрытом виде, например, клиенты для работы с Системой Быстрых Платежей, да и то лишь в виде Android приложений. Здесь помогает переориентация многих производителей на web-приложения. Ведь web для того и создавался, чтоб обеспечить совместимую среду между совершенно разными системами.

Планы

Рассматривается возможность поддержки других оболочек в будущем (`hyprland`, `gnome shell mobile`, `swmo`). Возможна поддержка устройств на другом оборудовании. Требуется взаимодействие с индустрией по её переориентации на создание свободных решений под Linux.

Литература

[1] ALT Mobile. URL: `https://altmobile.org`

[2] Образы ALT Mobile. URL: `http://beta.altlinux.org/mobile/latest/`

[3] Построение адаптивных приложений в Gnome. URL: `https://gnome.pages.gitlab.gnome.org/libadwaita/`

[4] TCB authentification system. URL: `https://www.openwall.com/tcb/`

[5] Проблема разблокировки экрана в phosh. URL: `https://bugzilla.altlinux.org/46389`

[6] Проблема разблокировки экрана в phosh. URL: `https://bugzilla.altlinux.org/47499`

Валентин Соколов
Саратов, «ООО Базальт СПО»

Проект: ALT Mobile `https://github.com/SokolovValy/alt-mobile-auth`, `https://github.com/SokolovValy/admx-altmobile`

Использование ALT Mobile в доменной инфраструктуре

Аннотация

В докладе рассматриваются специфические аспекты применения мобильной ОС семейства «Альт» в доменной инфраструктуре, разработка средств для её управления и настройки. Разрабатывая ОС «ALTMobile», мы начали включать в возможность её конфигурирования применение групповых политик. Задача заключается в организации централизованного управления мобильными устройствами. В рамках этой задачи существуют два ключевых направления: централизованное управление устройствами и централизованное управление настройками пользователей, которые используют данные мобильные устройства.

Ключевые слова: *ALT Mobile, Phosh, Групповые политики, доменная инфраструктура.*

Архитектурные особенности ALTMobile

Главной особенностью ALTMobile является то, что это — тот же Альт на базе GNU Linux. Благодаря общей базе с десктопным Альт Linux, можно легко настраивать и управлять мобильными устройствами теми же методами, что и десктопами. Это особенно важно для организационного сектора, где можно использовать единую политику управления устройствами.

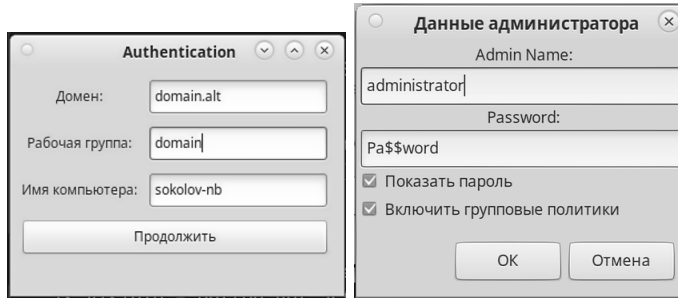
Реализация интеграции с доменом

Для введения дистрибутивов «Альт» в домен используется модуль аутентификации приложения Центр управления системой (ЦУС) — Alterator. В случае с ОС «ALTMobile» на текущий момент для введения в домен графические средства alterator не применимы. Для

решения данной задачи было разработано графическое приложение, являющееся аналогом, и сохраняющим функционал модуля аутентификации ЦУС.

Архитектура и принцип работы приложения аутентификации

Графический интерфейс, разработанный на Python с использованием PyGObject, в котором пользователь вводит данные о домене, компьютере, рабочей группе, логин и пароль администратора домена. Данные формируются в строку и передаются на системную шину Dbus в метод, который запускает скрипт system-auth с аргументом, строкой, переданной из GUI.



(a) Ввод данных о домене (b) Ввод данных администратора домена

Основные отличия механизма аутентификации в Phosh

Процесс логина в Phosh (Phone Shell) имеет несколько особенностей, которые отличают его от традиционного механизма логина на десктопе. Эти отличия связаны с его предназначением для мобильных устройств и упрощённым подходом к аутентификации:

Отсутствие дисплейного менеджера (DM)

- В Phosh обычно не используется отдельный дисплейный менеджер, как это делается на десктопе. Вместо этого Phosh напрямую управляет процессом запуска графической среды.

- Phosh запускается как часть системного процесса и автоматически инициирует сессию пользователя без необходимости отображения экрана выбора пользователя (Greeter).

Ориентация на одного пользователя

- Phosh ориентирован на работу с одним пользователем, так как мобильные устройства, для которых он предназначен, обычно настроены на одного владельца.
- Нет необходимости в переключении между пользователями или выборе рабочей среды, как на десктопе. Пользователь сразу входит в свою единственную сессию.

Краткое сравнение

- **Desktop:** Запуск DM → Greeter → Аутентификация → Запуск окружения рабочего стола.
- **Phosh:** Прямой запуск Phosh с заданным пользователем → Экран блокировки → Аутентификация → Отображение интерфейса Phosh.

Phosh использует упрощённый процесс, ориентированный на одного пользователя и мобильные устройства, без необходимости в дисплейном менеджере и полноценном greeter. Пользователь, под которым Phosh запускается по умолчанию, определяется на уровне машинной настройки. Если задать UID пользователя для загрузки системы, это позволит удалённо изменять рабочее окружение. Благодаря настройкам, управляемым через групповые политики, становится возможным переключение между окружениями разных пользователей. Загрузка графического интерфейса под доменным пользователем позволяет применять пользовательские групповые политики, обеспечивая централизованное управление настройками пользователей.

Разработка групповых политик для конфигурирования системы

Phosh, как часть экосистемы GNOME, использует GSettings для управления своими настройками, включая такие вещи, как конфигурация интерфейса пользователя, параметры энергосбережения, на-

стройки панели задач и других элементов среды. Пользователи и администраторы могут настраивать эти параметры через `gsettings`, изменяя соответствующие схемы и ключи.

На данный момент в ALTMobile присутствует около 1200 ключей gsettings, редактируя значения которых мы можем гибко настраивать систему средствами групповых политик.

Для управления графическим окружением был разработан ряд групповых политик, позволяющих гибко настраивать внешний вид приложений, тему оформления окружения, режимы электропитания, управление отображения времени, фон рабочего стола.

Литература

[1] Проект Phosh, `https://gitlab.gnome.org/World/Phosh/phosh`

[2] Проект Alterator-manager, `https://gitlab.basealt.space/alt/alterator-manager/-/blob/master/docs/README-ru.md?ref_type=heads`

[3] Проект Alterator-module-executor, `https://gitlab.basealt.space/alt/alterator-module-executor`

[4] Проект ALT Mobile, `https://www.altlinux.org/ALT_Mobile`

[5] Групповые политики [ОС «Альт»], `https://docs.altlinux.org/ru-RU/domain/10.2/html/group-policy/index.html`

Артём Быстров
п. Туймебаева, Алма-Атинская обл., Казахстан, ALT Linux Team

Проект: ALT Mobile
https://www.altlinux.org/ALT_Mobile/Портативные_приставки_Anbernic

ALT Mobile на игровых консолях Anbernic: зачем и куда?

Аннотация

Анализ истории, причин и следствий попыток запуска ОС ALT Mobile на устройства, отличные от смартфона Pinephone Pro, среди которых — линейка игровых портативных приставок от Anbernic.

Ключевые слова: *ALT Mobile, Anbernic.*

Первопричина

На протяжении всего процесса разработки проекта ALT Mobile мы чаще всего слышали три вопроса:

- На каких устройствах можно протестировать нашу ОС?
- Почему эти устройства такие дорогие?
- Почему их так мало?

В данном докладе автор попробует ответить на эти вопросы, а также попробует предложить возможную альтернативу дорогостоящим устройствам в виде китайских портативных приставок Anbernic тем, кому охота пощупать отечественную ОС ALT Mobile на железе, а не на виртуальной машине, за более скромную сумму.

На текущий момент ALT Mobile протестирован на совместимость со следующими моделями игровых приставок:

- RG 552;
- RG Arc D (S — в качестве побочного результата);
- RG 353M;

Почему приставки?

- Шаговая доступность — на текущий момент устройства можно приобрести в розничных магазинах бытовой электроники;
- Потому что конкретно у этих моделей есть сенсорный экран — ключевой момент при выборе модели;
- Потому что данные устройства выходят дешевле и не требуют серьёзной физической доработки напильником (изготовление корпуса, доп. органов управления и прочего.)

А почему не старые смартфоны?

Слишком много сложностей в плане разблокировки устройства (загрузчик, исходники драйверов и прочего), и доведения до ума без применения спец. средств обратной разработки.

Некоторые смартфоны действительно имеют возможноть загрузки дистрибутивов Linux, например ту же PostMarketOS. Но поддержка устройств осуществляется преимущественно силами отдельных энтузиастов и небольших групп, и многие устройства так и остались с поддержкой на уровне «Экран с tty-консолью».

Куда?

Дальнейшее развитие направления поддержки портативных игровых приставок — создание варианта прошивки с заточкой под игры (как эмуляторы, так и нативные игры), а также расширение списка поддерживаемых устройств по примеру проектов `Batocera Linux`, `EmuELET`, `AmberELEC`, `Rocknix` и пр.

Литература

[1] Страница на Альт Вики, посвящённая ALT Mobile на игровых приставках Anbernic. `https://www.altlinux.org/ALT_Mobile/Портативные_приставки_Anbernic`

[2] База мобильных устройств, на которые можно поставить Linux. `https://many.tuxphones.com/`

[3] ТГ-канал о новостях портативных эмуляторных приставок. `https://t.me.RetroGamingNaBalkone`

[4] Игровой дистрибутив для различных устройств. `https://github.com/batocera-linux/batocera.linux`

А. А. Волчек, Д. А. Костюк, А. А. Маркина, Д. О. Петров, Н. Н. Шешко
Брест, Брестский государственный технический университет

Разработка мультикомпонентной масштабируемой ГИС на базе виртуализации

Аннотация

Рассмотрена распределённая архитектура для динамического запуска элементов ГИС на основе платформы контейнерной виртуализации Kubernetes, изначально апробированная авторами в рамках проекта системы мониторинга паводков. Ядро системы включает кластер СУБД PostgreSQL с обеспечением высокой доступности на основе шаблона Patroni и расширением PostGIS. Рассмотрены взаимодействие компонентов системы, а также интеграция модулей импорта потоков данных от внешних источников и вычислительных модулей для оперативного обновления наложенных слоёв геоданных. Обсуждаются вопросы горизонтального масштабирования и автоматизации плановых апгрейдов системы, а также возможность интеграции специализированных средств хранения временных рядов.

Ключевые слова: *ГИС, PostgreSQL, Kubernetes, масштабирование*

Современные системы виртуализации и оркестрации виртуальных машин позволяют добиться качественно нового уровня программных комплексов в плане распределённой работы их компонентов, автоматического обновления и масштабирования. В настоящей работе рассмотрены особенности применения оркестратора виртуальных машин для создания сетевой геоинформационной системы, обладающей свойствами высокой доступности и возможностью автоматического восстановления от сбоев. Подход был апробирован в Брестском государственном техническом университете при проектировании системы визуализации паводков [1], [2], однако является универсальным, подходит для широкого класса сетевых геоинформационных систем и основывается на использовании свободных программных компонентов.

Разработанная архитектура программного комплекса построена на платформе контейнерной виртуализации Kubernetes, обеспечивающей параллельную работу программных модулей. Архитектура включает хранение в кластере СУБД PostgreSQL с расширением PostGIS, а также временных рядов, получаемых от средств мониторинга, при их наличии; в той же СУБД предусматривается хранение результатов расчётов полигонов для визуализации распределения климатических и др. параметров и селитебной информации. Применение оркестратора контейнеров Kubernetes позволяет осуществить автоматическое распараллеливание вычислений для одновременной работы с заданным количеством сегментов территории, а также обеспечивает горизонтальное масштабирование.

Архитектура системы, показанная рисунке 1, охватывает подсистемы сбора данных, расчёта контуров визуализации, СУБД и хранения данных. Подсистема сбора данных включает контейнеризованные модули импорта по расписанию данных из внешних источников (точек климатического мониторинга). Подсистема расчёта контуров визуализации включает контролёр развёртывания и репликации для динамического запуска модулей расчёта полигонов для всех участков моделируемой территории. Подсистема базы построена на оснвое образов пост-реляционной СУБД с расширением PostGIS для геоданных, а подсистема хранения предоставляет дисковые образы с необходимыми файлами данных контейнеризованным компонентам системы, обеспечивая возможность их динамического запуска и остановки.

Контейнеры СУБД PostgreSQL используются совместно с утилитой резервного копирования и восстановления данных pgBackRest, пулом сетевых соединений PostgreSQL pgBouncer и подсистемой высокой доступности PostgreSQL на основе шаблона Patroni. Согласованное взаимодействие компонентов обеспечивается специализированным контроллером Kubernetes — оператором, в качестве которого использован Percona Operator for PostgreSQL.

Когда создаётся новый пользовательский ресурс или существующий подвергается каким-либо изменениям или удаляется, оператор автоматически создаёт/изменяет/удаляет все необходимые объекты Kubernetes с предустановкой их настроек, требуемых для корректной работы кластера PostgreSQL. Функциональность оператора расширяет API Kubernetes за счёт определений пользовательских ресурсов (Custom Resource Definition, CRD), через которые и происходит декларативное конфигурирование кластера пользователем — админи-

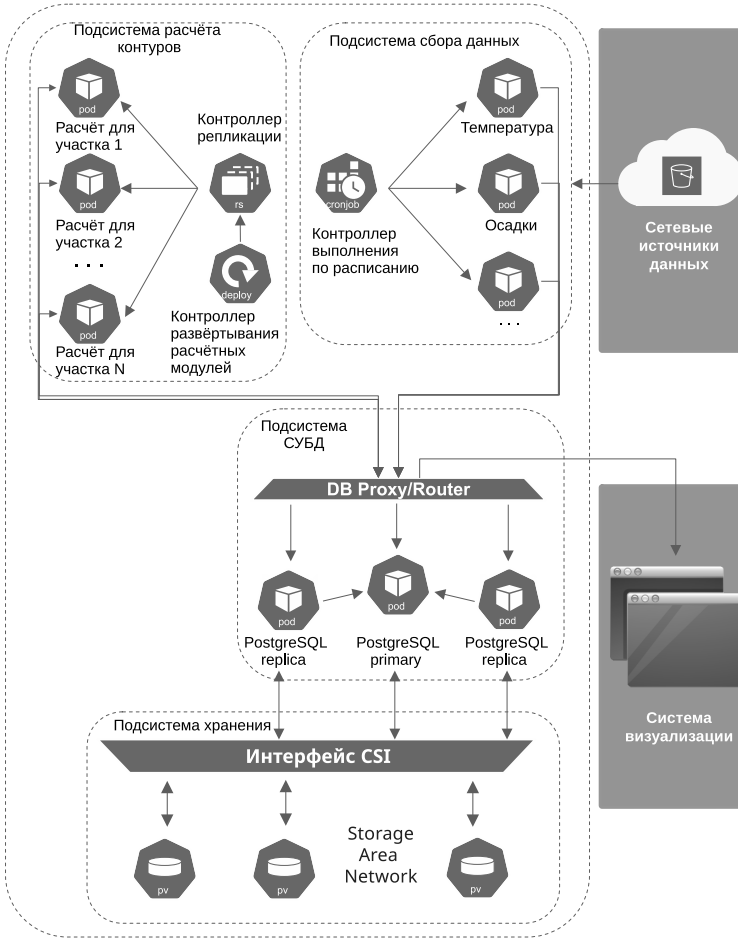


Рис. 1: Архитектура системы.

стратором системы. Оператор по мере необходимости развёртывает экземпляры СУБД PostgreSQL, пул соединений pgBouncer и т. д., и управляет ими. Также его задачей является автоматический проброс дисковых томов в запускаемые контейнеры с программными компонентами, что необходимо для реализации функционала базы данных

в среде, изначально ориентированной преимущественно на распределённые системы без сохранения состояния.

Чтобы обеспечить высокую доступность, оператор включает аффинитет — привязку узлов для запуска экземпляров кластера PostgreSQL, заданных в его пространстве имён, на отдельных рабочих узлах, если это возможно. При этом узлы являются взаимозаменяемыми, и если какой-то узел выходит из строя или теряет сетевое соединение, соответствующие контейнеры автоматически воссоздаются заново на другом узле.

Одним из больших преимуществ Kubernetes является простота масштабирования кластера, т. е. добавления ресурсов или модулей и их распределения на доступных узлах. Вертикальное масштабирование изменяет существующим узлам PostgreSQL доступное количество вычислительных ресурсов или доступное пространство для хранения данных. Горизонтальное масштабирование сводится в изменению числа узлов в кластере (аналогичный механизм обеспечивает высокую доступность, включая дополнительные узлы с целью поддержания работоспособности системы в случае сбоя).

Добавление или уменьшение доступных контейнерам ресурсов (вычислительной мощности процессора или памяти) выполняется редактированием соответствующих значений в пользовательском ресурсе. Горизонтальное масштабирование более проблематично для реализации в случае типичных СУБД, изначально предназначенных для использования на одном узле, поскольку в их архитектуре нет секционирования подсистемы хранения и вычислительной нагрузки. Поэтому кластеры строятся за счёт создания полных избыточных копий одних и тех же данных. Один из экземпляров берет на себя задачу обслуживания запросов на запись и репликации данных на другие узлы кластера, задачей которых является обслуживание запросов на чтение. При горизонтальном масштабировании запросов на чтение, когда увеличивается количество компонентов кластера в Kubernetes, один из компонентов выполняет создание резервной копии данных и отправляет её новому экземпляру для восстановления.

В тот момент, когда кластер масштабируется, производительность чтения существенно снижается, пока не завершится процесс сохранения резервной копии. Поэтому, в отличие от вертикального масштабирования, которое может выполняться адаптивно по мере необходимости, горизонтальное масштабирование целесообразно производить

вручную либо по расписанию, учитывающему ожидаемые пики нагрузки.

Отдельными компонентами на рис. 1, расширяющими функционал ГИС, выступают подсистема расчёта контуров, содержащая вычислительные модули, и подсистема сбора данных с модулями мониторинга. Вычислительные модули обеспечивают расчёт полигонов для обновления наложенных слоёв геоданных и определяются конкретным назначением ГИС (в нашем случае это расчёт контуров зон затопления территории в момент паводка). Модули мониторинга занимаются опросом сетевых сервисов либо устройств сбора данных (например, климатических). Поскольку и сервисы и устройства данной категории достаточно гетерогенны, модули сбора данных слабо поддаются унификации и представляют из себя штучный, но относительно простой программный код. Их задача — выполнение сетевого запроса в рамках API производителя устройства или сервиса, извлечение данных из ответа и загрузка новых значений конкретного временного ряда в базу.

В общем случае реляционные СУБД не позволяют эффективно работать с упорядоченным множеством элементов временного ряда. Наиболее производительным решением в рамках данной архитектуры является применение расширения PostgreSQL TimescaleDB. Однако в ряде случаев собственная реализация механизма хранения временных рядов в реляционной СУБД оказывается предпочтительнее, чем использование специализированного решения, несмотря на его функциональные возможности и оптимизацию. В таком случае применяется централизованная таблица, хранящая данные о конкретных измерениях (значения временных рядов) с уникальными составными ключами, объединяющими первичные ключи «таблиц измерений», которые содержат атрибуты измеряемых параметров. При соединении таблиц используется либо схема «звезды» (с уменьшением числа таблиц и ускорением запросов за счёт денормализации таблиц измерений), либо схема «снежинки» (с большим числом таблиц и более полной нормализациеи) [3]. В нашем случае вместо расширения TimescaleDB применялась схема «звезды». В частности, это позволило использовать контейнеры PostgreSQL с PostGIS из реестра Docker, на работу с которыми ориентирован применяемый оператор и которые он может обновлять в автоматическом режиме по мере выхода новых версий, оперативно устраняя возможные уязвимости.

Литература

[1] *Volchek A. A., Kostiuk D. A., Petrov D. O., Sheshko N. N.*, Estimating the socio-economic damage caused by river flooding // Modern technologies for solving actual society's problems / ed.: O. Nestorenk, I. Ostopolets. — Publishing House of University of Technology, Katowice, 2022. — Chapter 2.9. — P. 235–241.,

[2] *Volchek A., Kostiuk D., Petrov O., Sheshko N.*, A system for visualization and prediction of floods on lowland rivers // Pattern Recognition and Information Processing (PRIP'2023). Artificial Universe: New Horisont : Proceedings of the 16 th International Conference, Belarus, Minsk, October 17—19, 2023 / Belarusian State University : eds. A. Nedzved, A. Belotserkovsky. — Minsk : BSU, 2023. — Pp. 324–327. — P. 324–327, `http://www.lib.ru/CTOTOR/BRUKS/mithsoftware.txt`

[3] *Дубицкий А. В., Маркина А. А.*, Свободные СУБД для хранения временных рядов // Новые горизонты–2018. Сборник материалов белорусско-китайского молодёжного инновационного форума в 2 т. Т.2. — Минск, 15–16 ноября 2018. — С. 136–137.

Научное издание
ДВАДЦАТАЯ КОНФЕРЕНЦИЯ
РАЗРАБОТЧИКОВ СВОБОДНЫХ ПРОГРАММ

Сборник материалов конференции

Переславль-Залесский
04–06 октября 2024 года

Оформление обложки: А. А. Бусоргин.
Вёрстка: В. Л. Чёрный.
Редактура: В. Л. Чёрный.

Отпечатано с готового оригинал-макета
Подписано в печать 24.09.2024 Формат
60x90 1/16 Усл. пч. л. 7
Тираж 200 экз. Изд. 145

Издательство ООО «МАКС Пресс»
Лицензия ИД N 00510 от 01.12.99 г.
119992, ГСП-2, Москва, Ленинские горы, МГУ им. М. В. Ломоносова,
2-й учебный корпус, 527 к
Тел. 8(495)939-3890/91. Тел./Факс 8(495)939-3893

Отпечатано в полном соответствии с качеством
предоставленных материалов в ООО «Фотоэксперт»
109316, г. Москва, Волгоградский прооспект, д. 42,
корп. 5, эт. 1, пом. I, ком. 6.3-23Н

elibrary.ru

basealt.ru

basealt.ru/
20dev-conf