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

Девятая конференция разработчиков свободных программ

Обнинск, 23–24 июля 2012 года

Тезисы докладов

Москва,
Альт Линукс,
2012

УДК 004.91
ББК 32.97

Девятая конференция разработчиков свободных программ: Тезисы докладов / Обнинск, 23–24 июля 2012 года. М.: Альт Линукс, 2012. — 64 с. : ил.

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

ISBN 978-5-905167-11-9

Программа конференции

23 июля

11.00-12.30 Регистрация участников и заселение

Дневное заседание
12.30–13.40

12.30 А. Е. Новодворский. Открытие. Информация оргкомитета

12.40–13.10 Д. А. Костюк

Организация веб-площадки конференции с помощью lvee engine 6

13.10–14.40 А.Ю. Боков

Свободное программное обеспечение в облачных сервисах Microsoft 9

14.40–15.00 Перерыв на обед

Вечернее заседание
15.00–19.00

15.00–15.30 Д. Силаков, Е. Буданов

RPM5 — основные достижения и планы на будущее 10

15.30–16.00 М. С. Пожидаев

Deepsolver — инструмент управления пакетами с программным обеспечением 13

16.00–16.30 В. А. Липатов

Korinf: трудности на пути к универсальной сборочной системе 16

24 июля

Утреннее заседание
10.00–14.00

Дневное заседание
15.00–17.00

Александр Боровский, Дмитрий Костюк, Павел Чеботарёв
Минск

Проект: lvee engine `http://lvee.org`, `http://github.com/partizan/lvee/`

Организация веб-площадки конференции с помощью lvee engine

Аннотация

Проект *lvee engine* вырос из внутренних нужд конференции Linux Vacation / Eastern Europe, и на текущий момент используется в качестве основы веб-сайтов ещё двух международных конференций. Помимо средств работы с многоязычным контентом и переводами, lvee engine предоставляет функции создания и анонсирования мероприятий (конференций, семинаров и т. п.), регистрацию и обработку заявок на участие, онлайновую подачу и рецензирование тезисов. Проект написан на Ruby on Rails и распространяется под лицензией GPL v. 2.

История проекта

Изначально *lvee engine* создавался как основа сайта конференции LVEE[1] и под ее особенности: подготовку на волонтёрской основе и многоязычный контент. В первые годы конференция использовала сайт на медиа-вики, однако в 2008 г. организаторы получили разработчика и дизайнера, предложивших создать веб-площадку, лучше отражающую её нужды, на платформе Ruby on Rails. Первая реализация была весьма ограниченной и приняла сходство с текущим вариантом только в 2009 г., когда вместе с новым разработчиком появился базовый функционал *lvee engine*: управление контентом на основе разметки textile, а также пользовательские акаунты и поэтапная регистрация, рассчитанные на проведение нескольких конференций.

По итогам LVEE 2010 стало очевидным, что разработка достаточно универсальна, чтобы её можно было использовать в виде сети сайтов дружественных конференций на одном и том же «движке» с разными оформлением, контентом и хостингом. В 2011 г. на базе *lvee engine* были подняты сайты конференций FOSS Lviv[2] и WebCamp[3],

[1] `http://lvee.org`
[2] `http://conference.linux.lviv.ua`
[3] `http:webcamp.in.ua`

начат процесс доработки функционала для обмена акаунтами пользователей между сайтами.

В 2012 г. в силу назревшей необходимости коллективного рецензирования материалов LVEE в проекте появился механизм онлайновой работы с тезисами. Однако в целом 2012 г. оказался переходным в связи с портированием на последнюю версию Ruby и сложностями в привлечении дополнительного человеческого ресурса. В результате автоматический обмен с доверенными сайтами, включающий, согласно плану разработки, репликацию учётных записей, новостей и фрагментов контента, на текущий момент не действует, а синхронизация учётных данных пользователей выполняется в полуавтоматическом режиме (например, по окончании очередной конференции).

Регистрация участников

Из-за ориентированности на несколько конференций для участников предусмотрена двух- либо трехуровневая регистрация.

Первый этап — создание учётной записи на сайте (при действующем обмене и наличии ключей доверенных сайтов должна выполняться попытка прозрачного импорта).

Активировав учётную запись, пользователь получает доступ к скрытому контенту. Для непривилегированных пользователей это оглавление вики-раздела сайта и список участников, зарегистрированных на ближайшее мероприятие.

Второй этап регистрации — подписка на участие в каком-нибудь мероприятии сайта (конференции) с открытой регистрацией.

Третий, необязательный этап — заполнение участником дополнительных полей анкеты и правка сгенерированного бэджа — становится активным после подтверждения заявки на участие, сделанной на втором этапе.

Роли пользователей

Для пользователей предусмотрены следующие роли:

- *none* (без привилегий) — права на создание и редактирование вики-страниц, своих анкетных данных и тезисов;
- *editor* (редактор) — права на редактирование контента и доступ к дополнительным управляющим страницам;

- *reviewer* (рецензент) — разновидность редактора с доступом к общему списку тезисов;

- *admin* (администратор) — может создавать новые конференции, открывать и закрывать регистрацию на них, подтверждать регистрации участников (при этом участник получает настраиваемое письмо от робота с предложением подтвердить участие, заполнив третью анкету в личном профиле), работать с логотипами спонсоров и партнёров конференции и др.

Редактирование контента

Управление контентом предполагает, что первичным языком является английский, а остальные языки являются переводами (на сайте WebCamp первичен русский язык, но соответствующий патч не включён в основное дерево проекта). Если нет перевода страницы на выбранный пользователем язык, отображается контент первичной версии, т. е. при добавлении администратором нового языка его контент полностью англоязычен.

Технически процесс редактирования контента бывает двух видов: редактирование сообщений (в виде yml-файла для каждого языка) и редактирование текстов, причём последнее делится на редактирование статей (страниц сайта), новостей и тезисов.

Редактирование статей построено следующим образом: на каждой странице редакторы видят под текстом ссылку «перевести» либо «править». В дополнение к разметке textile можно использовать html-теги и формулы LaTeX (последнее реализовано для нужд подготовки тезисов). Механизм редактирования поддерживает предварительный просмотр и показ полной истории изменений, а для переводчиков предусмотрен RSS-канал правок.

Новость отличается тем, что делится на аннотацию (которая вместе с заголовком выводится на главной странице сайта и попадает в публичный RSS-канал новостей) и основной текст (открывается щелчком по заголовку). Новость после создания не попадает на главную страницу и в RSS-канал, пока не будет опубликована. Мгновенная публикация доступна администраторам, а отложенная на сутки (чтобы дать время переводчикам) — также и редакторам.

Работа с тезисами

Создать новые тезисы участник может, перейдя по ссылке в личном профиле (после подписки на участие в конференции). Рецензентам доступен список тезисов, поданных на текущую конференцию, а остальные участники могут видеть только собственные. За основу формы создания тезисов взята форма новостей с дополнительными полями, часть из которых генерируется автоматически: название конференции, авторы, лицензия (по-умолчанию Creative Commons Attribution-ShareAlike 3.0), комментарий к изменениям (при повторных правках), флаг готовности к рецензированию. При редактировании тезисов доступен выбор пользователей для совместного редактирования и прикрепление файлов (например, изображений). Ниже текста тезисов, готовых к рецензированию, отображается область комментариев. В комментариях рецензенты могут сообщить автору о принятии тезисов, либо предложить их доработать, либо вести с автором более длительную дискуссию.

Алексей Боков
Москва, Microsoft
Проект: Windows Azure `http://windowsazure.com`/http://windowsazure.com

Свободное программное обеспечение в облачных сервисах Microsoft

Аннотация

Доклад посвящен практическому опыту использования свободного программного обеспечения для облачных проектов на базе платформы Microsoft — Windows Azure. Экосистема разработчиков свободного программного обеспечения как важный фактор развития облачных сервисов. Вклад компании Microsoft в разработку и развитие свободного программного обеспечения.

Денис Силаков, Евгений Буданов
Москва, ROSA Lab.
Проект: ROSA, RPM5 http://www.rosalab.ru

RPM5 — основные достижения и планы на будущее

Аннотация

Отличительной чертой многих дистрибутивов Linux является оригинальный подход к управлению программным обеспечением, основанный на формировании пакетов ПО и использовании специализированных систем управления такими пакетами. Распространенным является мнение, что подобные системы — это всего лишь архиваторы, однако в современном мире это далеко не так. Доклад посвящен RPM5 — одной из активно развивающихся систем управления пакетами, являющейся ответвлением широко распространенного инструментария RPM. Приводится обзор нововведений, реализованных в рамках проекта RPM5, а также улучшений, которые планируется добавить в будущем.

История проекта RPM5 насчитывает уже несколько лет. В настоящее время этот формат и инструментарий используется в качестве основного в ряде дистрибутивов (в частности, уже более года — в ROSA и Mandriva), и уже можно судить о том, какие из наработок RPM5 оказались полезными на практике, и какие из запланированных нововведений кажутся наиболее интересными для разработчиков и пользователей дистрибутивов.

Улучшения для мантейнеров

Одним из наиболее активно используемых мантейнерами новшеств являются файловые триггеры. Механизм триггеров позволяет избавиться от указания в post- и pre-скриптах повторения рутинных действий, необходимых при установке или удалении многих пакетов — например, запуска `ldconfig` для перегенерации кэша `/etc/ld.so.cache` при изменении состава библиотек или обновления кэша иконок при изменении файлов в `/usr/share/icons`. В RPM5 вместо этого предлагается использовать триггеры — скрипты, запускаемые самим RPM при появлении или удалении файлов с определенными именами или местоположением.

Еще одной активно прорабатываемой областью является локализация описаний пакетов. Существующие в настоящее время подходы

(помещение переводов в spec-файл, использование отдельного пакета specspo) на практике оказались не очень удачными. Разработчики RPM5 предложили использовать для хранения описаний пакетов специализированное хранилище (базу данных), работа с которой будет производится непосредственно инструментарием RPM.

Потенциально полезным также видится использование параллелизма при сборке пакетов — например, использование параллельного сжатие их содержимого. Это позволило бы ускорить процесс сборки на многоядерных машинах. Впрочем, технически это не всегда просто — например, параллельные алгоритмы алгоритмы сжатия существуют для gzip и для bzip2, а вот реализации для становящимся все более популярным xz пока нет.

Из других изменений, облегчающих работу мантейнеров, можно выделить следующие:

- Возможность использования встроенных интерпретаторов — RPM5 может быть собран со встроенной поддержкой Ruby, Python, Perl, Tcl и Lua, а также с базовой поддержкой ODBC и языка запросов SQL.
- Автоматизация рутинных действий при сборке пакетов, позволяющая существенно уменьшить размер spec-файлов и сделать их более читаемыми. В частности, мантейнерам предоставляется богатый набор вспомогательных макросов, автоматические генераторы зависимостей и прочие удобства.
- Улучшенная работа с сетью — практически все операции, работающие с локальными файлами, могут работать и с файлами по сети (по протоколам HTTP и FTP).

  Отметим, что достаточно много наработок в области сборки пакетов специфичны для конкретных дистрибутивов и реализуются их разработчиками как надстройки к RPM5 либо как части сборочных систем дистрибутивов.

Улучшения для пользователей

Ряд нововведений RPM5 будет заметен и конечным пользователям. К таким аспектам относится, прежде всего, транзакционное управление пакетами, позволяющее представить установку пакета как атомарную транзакцию (по аналогии с транзакциями СУБД), которую, в случае необходимости, можно откатить.

Также пользователи могут заметить прирост производительности при установке пакетов на многоядерных машинах, где RPM5 способен выполнять параллельно несколько действий на разных ядрах процессора. В настоящее время распараллеливание используется при обработке групп пакетов (для проверки их подписей, контрольных сумм и тому подобного). Возможность использования параллелизма на более низком уровне (например, для разархивирования содержимого пакета — одного из наиболее затратных по времени действий) исследуется, однако внедрение подобных механизмов сопряжено с существенными техническими трудностями и требует тщательного тестирования.

Наконец, встраивание в RPM минимальных возможностей по работе с системами контроля версий позволяет использовать такие системы для отслеживания модификаций конфигурационных файлов при обновлении пакетов. В настоящее время RPM умеет сохранять предыдущую версию файла с суффиксов .rpmsave (либо оставлять старую версию на месте, устанавливая новую с суффиксом .rpmnew). Использование систем контроля версий позволит отслеживать все изменения в файлах, не замусоривая файловую систему. При этом пользователь получает возможность сравнивать различные версии и даже производить их слияние (merge) помощью RPM.

Итоги

Сегодня уже можно сказать, что RPM5 доказал свою состоятельность и пригодность к использованию в промышленных масштабах. Проект активно развивается, при этом наряду с собственными улучшениями, производится перенос полезных наработок из RPM4, что позволяет сохранять совместимость с этим форматом. За развитием RPM5 можно следить на сайтах `http://rpm5.org/` и `http://launchpad.net/rpm`. Процесс разработки полностью открыт и принять в нем участие могут все желающие.

Михаил Пожидаев
Томск, ALT Linux
Проект: Deepsolver `http://deepsolver.org`

Deepsolver — инструмент управления пакетами с программным обеспечением

Аннотация

Доклад посвящён новому инструменту управления пакетами с программным обеспечением, разработка которого ведётся сотрудниками компании ALT Linux. Особое внимание уделяется алгоритмам вычисления списков пакетов для внесения изменений в состояние операционной системы.

Deepsolver — новый инструмент для управления пакетами с программным обеспечением (ПО). Разработка ведётся сотрудниками компании ALT Linux с целью замены менеджера APT новым продуктом, удовлетворяющим современным требованиям. Основные функции Deepsolver:

- доставка с удалённых узлов, установка, обновление и удаление пакетов;
- наполнение автоматизированной среды для компиляции исходных текстов;
- поиск доступных пакетов в удалённых репозиториях и предоставление информации о них;
- поддержание целостности операционной системы (ОС).

Разработка ведётся с выделением трёх основных компонент:

1. Обработка запросов пользователя на внесение изменений в ОС.
2. Поиск информации о доступных пакетах.
3. Построение индексов репозиториев.

Схема зависимостей и конфликтов между пакетами в современных дистрибутивах GNU/Linux может быть сведена к булевому уравнению. Это доказывает, что задача обработки запросов пользователя является задачей из класса NP-полных алгоритмов. Нахождение точного решения за полиномиальное время невозможно, и необходимо использовать методы приближённого или ограниченного поиска.

Приближённый метод подразумевает сохранение задачи NP-полной с привлечением одного из известных приближённых алгоритмов решения булевых уравнений (SAT). Время вычислений сокращается, но нахождение решения не гарантируется даже в случае его существования. Ограниченный подход заключается в использовании точного полиномиального алгоритма с предварительным наложением дополнительных условий в формулировке задачи.

Архитектура Deepsolver допускает существование нескольких реализаций алгоритма одновременно. Тем не менее, реализация по умолчанию основана на ограниченном подходе. Это достигается путём наложения дополнительного ограничения «одна зависимость — одно возможное разрешение». Другими словами, сформулирован точный алгоритм, однозначно выбирающий пакет, подходящий под требования некоторой зависимости. Хотя в действительности за счёт существования записей provides подобных вариантов может быть несколько. Помимо возможности применять полиномиальный алгоритм, новое ограничение повышает предсказуемость работы в условиях формирования автоматизированного сборочного окружения.

Во время вычисления изменений состояния ОС в ответ на запрос пользователя обрабатываются три множества пакетов:

1. Пакеты, непосредственно указанные пользователем.
2. Пакеты, установка, обновление или удаление которых однозначно необходима вследствие прямых зависимостей или конфликтов пакетов, указанных пользователем.
3. Пакеты, установка, обновление или удаление которых необходима для сохранения целостности ОС.

Первые два множества пакетов в большинстве случаев могут быть вычислены однозначно, но при поиске путей сохранения целостности ОС возможно существование нескольких решений. Особое внимание вызывают пакеты, для которых принято решение об удалении или обновлении. Для них существует три варианта действий:

1. При нарушении зависимости на пакет в случае его обновления произвести обновление зависимых пакетов.
2. Произвести замену пакета, используя правила обработки зависимостей, если результат вычислений не будет совпадать с удаляемым или обновляемым пакетом.

3. Полностью удалить зависимое поддерево установленных пакетов.

Обновление зависимых пакетов имеет смысл выполнять только в случае обновления исходного пакета и должно рассматриваться отдельно. Для выбора между второй и третьей альтернативой можно использовать ограниченное решение булевого уравнения, в котором каждый элемент нормальной формы имеет только две переменных, т. е. так называемый 2-SAT.

Пусть пакет p зависит от r_1 и r_2, а конфликтует с c_1 и c_2. В таком случае соответствующий фрагмент уравнения должен быть следующим: $(!p \vee r_1) \wedge (!p \vee r_2) \wedge (!p \vee !c_1) \wedge (!p \vee !c_2)$. В роли p выступают все пакеты, для которых необходимо принять решение, а также связанные с ними путём зависимостей или конфликтов. Для каждой альтернативы удаления поддерева или установки замещения в уравнение должен добавляться новый элемент вида $(p_1 \vee !p_2)$, где p_1 — пакет замещения, а p_2 — пакет удаления.

На первый взгляд кажется, что уравнение принимает огромные размеры, но его решение может быть найдено за линейное время. Каждый элемент вида $a \vee b$ эквивалентен двум импликациям: $!a \to b$ и $!b \to a$. Это позволяет совершить переход к ориентированному графу импликаций, в котором каждая переменная заменяется двумя вершинами: для положительного и для отрицательного значения. Справедливо утверждение, что решение существует, если не существует одновременно пути на графе из a в $!a$ и наоборот для любых a. Фактически, необходимым и достаточным условием для этого является требование, чтобы a и $!a$ принадлежали бы разным компонентам сильной связности. Их поиск может выполняться за линейное время и позволяет определить конкретное решение уравнения.

Виталий Липатов
Санкт-Петербург, Etersoft

Проект: Korinf `http://wiki.etersoft.ru/korinf`

Korinf: трудности на пути к универсальной сборочной системе

Аннотация

Доклад посвящён системе Korinf, позволяющей осуществлять сборку пакетов под множество операционных систем из единого исходника. Будет рассмотрен круг решаемых системой задач, а также сложности, возникшие в ходе реализации проекта и планы по его развитию.

«Идеальные спеки и полная пересобираемость нужны для упрощения разработки решений, которые принято называть дистрибутивами.»

Алексей Новодворский в рассылке devel@altlinux.org, 17.02.2008

Сборочная система нужна для сборки бинарных пакетов с программами из исходных кодов. Сборочная система необходима как производителям дистрибутивов, так и поставщикам (разработчикам) программ, причём они предъявляют разные требования. В дистрибутиве стоит задача собрать множество пакетов в одной сборочной среде (определённой версии дистрибутива). Поставщику проприетарной (или свободной, но не попавшей во все дистрибутивы программы) требуется собирать свою программу под множество различных дистрибутивов, или по крайней мере сделать её работающей на разных дистрибутивах.

Варианты тут следующие:

- собирать программу статически со всеми библиотеками (или со многими, потому что со всеми — нереально). Например, собрать статически с Qt, и на довольно широком круге систем это будет работать;
- носить все библиотеки с собой (так поступал раньше Oracle и Dr Web, причём с собой они носили даже glibc);
- можно надеяться, опираясь на стандарты, что в системе будет нужный набор библиотек нужных версий;

- можно собирать программу под каждую конкретную систему в каждой конкретной системе.

Стоит заметить, что по последнему варианту собираются все пакеты в штатных репозиториях дистрибутива. Сборка с конкретными версиями библиотек и последующая работа с этими же библиотеками — самый предусмотренный путь (по сути — знаменитый configure && make && make install).

Дополнительно задача осложняется тем, что мало скомпилировать программу, далее её нужно упаковать в пакет, который потом желательно устанавливать средствами самой системы.

Тут тоже есть несколько решений:

- поставлять собранную программу просто в архиве (tar.gz) в виде набора файлов;
- поставлять программу в виде саморазархивирующегося архива (по сути — инсталлятора);
- поставлять программу в неком отдельном формате (например, проект OpenPKG), не совместимым с пакетным менеджером системы ;
- создавать пакет под конкретный пакетный менеджер целевой системы.

Из чего и как собирать. Опять же, существует несколько подходов, применяемых в существующих сборочных системах:

- программа собирается из одного исходного кода, но по спецификации (спеку), написанной для каждого дистрибутива (его пакетного менеджера) и для каждой версии;
- иногда в попытках упростить пытаются писать универсальный спек для rpm, который полон условиями и пытается вести себя правильно в зависимости от дистрибутива и версии дистрибутива, под которой он собирается;
- используется единая спецификация, на основании которой готовится задание для конкретного пакетного менеджера;
- используется спецификация для одного из пакетных менеджеров: например, собирается всегда rpm-пакет, который далее конвертируется в формат пакетного менеджера целевой системы.

В Korinf используется следующий подход: программа (пакет) собирается под каждую конкретную систему в соответствующей ей сборочной среде (то есть в самой целевой системе). Сборка происходит с помощью rpm по спеку, полученному путём конвертирования из идеального спека, созданного для ALT Linux Sisyphus. Полученный rpm конвертируется в формат пакетного менеджера целевой системы.

При этом решаются следующие задачи:

- получение списка бинарных и сборочных зависимостей пакета (названий пакетов в системе, от которых зависит данный пакет);
- конвертирование названий пакетов и названий групп пакетов в названия, принятые в целевой системе;
- создание списка зависимостей для целевой системы из списка бинарных зависимостей пакета для ALT Linux;
- конвертирование спека в формат, принятый в целевом rpm-менеджере;
- портирование rpm-макросов, созданных для ALT Linux, на целевую систему.

Что не решено хорошо и подлежит улучшению:

- сборка для большинства систем происходит в обычном chroot, только для ALT Linux используется hasher. Нужно использовать изолированное сборочное окружение: портировать hasher, использовать mock, pbuilder и пр., доступное для соответствующих платформ, чтобы обеспечить повторяемость и изоляцию сборки;
- разворачивание минимальной сборочной системы. Задача похожа на создание шаблона системы для OpenVZ. Общего решения нет, обычно это делается вручную;
- трансляция названий пакетов сейчас производится по вручную заполняемой базе соответствий названий, а не из базы, полученной на основании автоматического анализа дистрибутивов (есть несколько плохо развивающихся проектов, которые это делают: distromatch, packagemap);
- создание репозиториев. Собранные пакеты неплохо бы объединять в репозиторий (добавлять файлы с метаинформацией о пакетах), но как правило в системе с одним пакетным менеджером сложно генерировать репозитории для других пакетных менеджеров;

- конвертирование исходного src.rpm в src.rpm для целевой системы происходит в хост-системе, что накладывает ограничения (например, для python-пакетов может неверно сохранится требуемая версия python);
- веб-интерфейса для взаимодействия нет, только командная строка;
- нет полного взаимодействия с girar (сборочной системой в ALT Linux), поэтому автоматическая сборка делается скриптом, который следит за изменениями в заданных репозиториях ALT Linux и пересобирает пакеты при появлении новой версии.

Описанный подход позволяет «из одного исходника» собирать бинарные пакеты для систем на базе Linux, FreeBSD, Solaris. Для сборки под Windows требуется написание отдельного сценария для сборки и для инсталлятора.

Korinf, не являясь сборочной системой в общепринятом смысле (когда подразумевается разделение доступа, отправка пакетов на сборку, создание репозиториев, веб-интерфейс управления, контроль зависимостей и пр.), предполагает использование совместно с системами сборки, обеспечивающими взаимодействие с пользователями (разработчиками).

Роман Вялов, Владимир Рубанов, Евгений Соколов, Владимир Шаршов
Москва, ЗАО «РОСА»

Проект: Automatic Build Farm(ABF) http://abf.rosalinux.ru

Automatic Build Farm (ABF): система для совместной разработки свободного ПО

Аннотация

Доклад посвящен системе для совместной работы над свободным программным обеспечением — ABF (Automatic Build Farm). Описываются основные идеи, положенные в ее основу при проектировании и разработке, рассматриваются наиболее важные возможности для различных групп пользователей. Дается обзор текущей архитектуры, описываются недостатки, выявленные в процессе первичной эксплуатации, рассматриваются дальнейшие направления развития.

Введение

Разработку сложного многокомпонентного программного обеспечения (ПО) практически невозможно проводить без использования специальных средств автоматизации. Одним из самых ярких примеров такого ПО служат многообразные дистрибутивы операционной системы Linux, каждый из которых включает в себя тысячи различных компонентов (пакетов). Соответствующие системы сборки пакетов являются сердцем разработки любого серьезного дистрибутива. Выигрыш от их использования по сравнению с ручной организацией колоссален. У каждой такой системы есть свои особенности — развитая система автоматического тестирования, поддержка частных репозиториев, возможность сборки под различные дистрибутивы и т.п.. Однако большинство систем не предоставляют разработчикам ПО интегрированной площадки для работы, импортируя необходимый код из других сервисов.

В то же время развитие таких веб-сервисов как Github, который предоставляет git-хостинг для открытых проектов, показало, что разработчикам важно использовать удобный веб-интерфейс для работы с проектами. Подход Github к пользовательскому интерфейсу и возможностям, которые он предоставляет, сделал его одной из самых популярных площадок для хостинга кода в мире открытого ПО.

Логичным видится объединение подобных систем сборки и сервисов разработки в единую среду, которая бы позволила разрабатывать, тестировать и собирать ПО, а также общаться и обмениваться кодом, используя единую площадку. Самая известная на текущий момент система, которая попыталась объединить эти направления — Launchpad от компании Canonical.

ABF — система для совместной работы над свободным ПО

Изначально для разработки продуктов в компании РОСА использовались наработки Mandriva — одного из старейших международных Linux дистрибутивов, разработка которого велась с использованием собственной системы сборкиnull — Kenobi. К сожалению, к 2010 году система морально и технологически устарела, не осталось специалистов, которые могли бы её активно поддерживать и развивать. Исходя из этого, в РОСЕ было принято решение начать разработку новой сбо-

рочной системы на основе опыта, современных знаний и технологий, накопленных мировым сообществом.

Основные идеи, которые были взяты за основу:

- хостинг, разработка и сборка кода должны быть доступны на единой площадке;
- пользовательский интерфейс — важная часть системы;
- поддержка разработки множества различных дистрибутивов;
- собрать свою образ дистрибутива должно быть не просто, а очень просто;
- персональный репозиторий должен быть доступен каждому пользователю;
- всесторонние автоматические проверки пакетов — залог здоровья репозитория.

Самые интересные возможности, уже доступные для пользователей:

- *Запросы на изменения* (PullRequest) — запросы на изменения кода проекта в веб-интерфейсе позволили значительно упростить и ускорить процесс обсуждения и приема патчей.
- *Интегрированная проектная wiki* - сохранять знания о проекте рядом с ним, как показал опыт Github, удобно и полезно как для разработчиков, так и для сообщества.
- *Приватные проекты* — ПО бывает не только открытым. Возможно использовать одну систему для разного ПО, а в будущем использовать его как площадку для распространения закрытого ПО для различных Linux-дистрибутивов.
- *Частные репозитории* — возможность для самореализации пользователей, распространение собственного ПО, сборка общего ПО со своими специфическими настройками и патчами, формирование разносторонних сообществ на одной площадке.

Архитектура ABF

Главная идея архитектуры: изолированные подсистемы (модули), общающиеся друг с другом строго по API и оперирующие обобщенными (абстрактными) элементами. Изоляция позволяет точно определить зоны ответственности каждой подсистемы, уменьшить взаимное влияние и возникновение “наведенных” ошибок, а также разрабатывать подсистемы параллельно разными командами. Важным

следствием такого разделения стала возможность сделать сборочные клиенты для множества дистрибутивов, так как вся специфическая для конкретного дистрибутива логика сосредоточена именно в самом клиенте, в то время как остальные элементы системы по-прежнему работают с общими абстрактными элементами.

Система состоит из следующих компонентов:

- система хранения git-репозитория;
- веб-подсистема;
- ядро сборочной системы;
- сборочный клиент;
- клиент для сборки ISO-образов.

Конечно, высокий уровень абстракции накладывает ряд ограничений. В ряде случаев в процессе разработки приходится в целях оптимизации вводить частные эвристики. Данные исключения с одной стороны ускоряют разработку и работу системы, с другой снижают "уровень универсальности" системы. Выбор оптимального соотношения является предметом постоянного исследования и сопоставления теории с практикой разработки различных дистрибутивов.

Текущие проблемы и способы их устранения

В процессе первичной эксплуатации при разработке релиза ROSA Marathon 2012 был выявлен ряд недостатков и узких мест в текущей реализации:

- хранение архивов с исходным кодом (tar, zip) в git — в будущей версии будет реализован отдельный сервис хранения таких архивов, что позволит значительно сократить объемы трафика при работе с системой, позволяя в том числе работать мейнтейнерам со spec файлом через веб — разработка Linux с не-Linux машин станет реальной;
- возможность потери задания или его результатов при нештатном отключении сборочного узла — переход на очередь сообщений позволит не терять ни одного сообщения между подсистемами;
- общая файловая система NFS для взаимодействия между клиентами и ядром сборочной системы — сборочный клиент будет

изолирован от основных подсистемы как наиболее уязвимое звено, все команды будут передаваться через очередь сообщений, а передача данных осуществляться через протокол HTTP.

Заключение

ABF находится в самом начале своего развития, хотя уже обладает минимально необходимым набором функций и качеством. В частности, с помощью ABF были проведены работы по созданию и выпуску промышленной версии дистрибутива ROSA 2012 Marathon, а сейчас осуществляется поддержка весеннего и подготовка осеннего выпусков, а также разработка серверной версии ROSA. Кроме того, ABF используется для пересборки целого ряда дистрибутивов (НауЛинукс, Fedora, OpenSUSE, Альт Линукс и др.) в рамках проекта Минобразования.

Компания РОСА нацелена на долговременные инвестиции в разработку ABF — регулярно будут выходить новые версии системы. В ближайших планах, помимо рефакторинга и переработки архитектуры и решения упомянутых выше проблем, мы планируем реализацию следующих функций:

- REST API для управления сборочными заданиями;
- интеграцию автоматических тестов от rpmlint до LSB;
- повышение безопасности сборочной системы путем изоляции сборочных клиентов;
- консольный клиент работы с ABF.

ABF является свободной и открытой системой и доступна для разворачивания частных сборочных для разработки различных дистрибутивов. В частности, ведутся обсуждения об использовании ABF для разработки коммьюнити версий Mandriva, MIB и Unity Linux. Мы предлагаем всем желающим участвовать в проекте.

Игорь Власенко
Киев, ALT Linux

Проект: Автоматизация сопровождения программных пакетов
http://www.altlinux.org/Категория:Packaging_Automation

Автоматизация сборки и сопровождения пакетов для дистрибутива. Технологии и инфраструктура.

Свободное программное обеспечение за последнее десятилетие сделало потрясающие успехи в своем развитии. Однако этого нельзя сказать о собственно дистрибутивах Linux. Linux все еще не стал основой рабочего стола пользователя; дистрибутивы Linux не успевают за потребностями пользователей; среди недовольных пользователей дистрибутивов Linux такие люди, как Линус Торвальдс и Инго Молнар.

В своей статье «Что отравляет Linux desktop?» Инго Молнар пишет: *«Удивительно, но главным недостатком дистрибутивов Linux для настольных ПК является их недостаточная свобода. . . .*

Дистрибутивы Linux создали собственные замкнутые (и даже закрытые) экосистемы и пытаются контролировать по 20 тысяч программных пакетов, которые суммарно содержат миллиарды строк кода. Обычные задержки при обновлении приложений составляют недели (вплоть до месяца) для исправлений безопасности и месяцы (вплоть до года) для серьёзных нововведений. Все Linux-дистрибутивы — централизованные организации с иерархической структурой, а не распределённые в пространстве свободные демократические сообщества.»

Большой проблемой при создании и сопровождении дистрибутивов является проблема масштабирования: рост числа пакетов требует роста числа майнтайнеров, но пакеты зависят друг от друга и от политик дистрибутива, поэтому рост майнтайнеров ограничивается в том числе проблемами координации и взаимодействия, такими, как закон Брукса.

Отвечая Инго Молнару, можно сказать, что централизованные Linux-дистрибутивы с иерархической структурой являются вынужденным ответом на проблему взаимодействия. Распределённые в пространстве свободные сообщества являются действительно огромной силой; но для того, чтобы ей воспользоваться, необходимы технологии и инфраструктура. Днепр, протекающий через Украину, облада-

ет огромной энергией; но для того, чтобы взять из нее хотя бы пару мегаватт, необходимо перегородить реку плотиной и установить гидроэлектростанцию. То же и для использования возможностей сообщества.

В докладе «Автоматизация сопровождения программных пакетов» будет рассказано о различных разработанных в рамках ALT Linux Team технологиях автоматизации цикла сопровождения программных пакетов, позволяющих автоматизировать рутинные задачи сопровождения от мониторинга выхода новых версий до автоматического обновления пакета.

В докладе «Облачно-дружественная распределенная инфраструктура для сервисов автоматизации ALT Linux Team» будет рассказано о развертываемой и находящейся в процессе развертывания инфраструктуре сервисов автоматизации, предназначенной для майнтайнеров и пользователей репозиториев ALT Linux.

Технологии и сервисы автоматизации позволяют

– значительно повысить экономическую эффективность сопровождения, т.е. с их помощью майнтайнер может на порядки снизить затраты своего времени либо на порядки повысить число сопровождаемых пакетов;

– активно использовать пакетную базу других дистрибутивов, позволяя достичь подлинно распределенной разработки Linux desktop платформы и избежать неоправданного дублирования усилий;

– активно привлечь пользователей к тестированию пакетной базы.

Технологии и сервисы автоматизации уже прошли стадию прототипов и экспериментов, следующей целью является их широкое внедрение в работу ALT Linux Team.

Литература

[1] `http://www.altlinux.org/Croncopy`

[2] `http://www.altlinux.org/Cronport`

[3] `http://www.altlinux.org/Gear/cronbuild`

[4] `http://www.altlinux.org/Git.alt/girar-nmu`

[5] `http://www.altlinux.org/Packaging_Automation/DistroMap`

[6] http://www.altlinux.org/Repocop

Виталий Кузнецов
Москва, ALT Linux
Проект: Sisyphus http://sisyphus.ru

ALT Linux: Седьмая платформа

Аннотация

ALT Linux Sisyphus — постоянно развивающийся нестабильный репозиторий свободного ПО. На его базе периодически выпускаются различные решения. Доклад посвящён готовящемуся выпуску очередного стабильного бранча ALT Linux, описываются основные задачи выпуска, трудности на пути их решения, текущий статус процессов разработки.

Что такое «платформа» ALT Linux?

Платформа ALT Linux — это всё необходимое для выпуска стабильных поддерживаемых дистрибутивных решений. В неё входит:

- Стабильный поддерживаемый срез Sisyphus с определёнными свойствами;
- Инструменты создания дистрибутивов (mkimage), типовые профили (mkimage-profiles, mkimage-profiles-desktop,...);
- Инфраструктура поддержки (сборочница, bugzilla,...);
- Сопутствующие инструменты.

Почему настало время выпускать новый бранч?

В ALT Linux нет календарного плана выпуска новых бранчей, подготовка нового стабильного среза начинается тогда, когда на старом либо невозможно обеспечить требуемый новый функционал с сохранением обратной совместимости, либо это требует слишком больших трудозатрат.

Что нового будет в p7?

Основные задачи выпуска следующие:

- Новый xorg-1.12 с поддержкой современных методов ввода (multitouch). Будет обеспечена поддержка выпуска дистрибутивов для ноутбуков с тачскринами и планшетных устройств.
- Поддержка Systemd. Решаемые в рамках выпуска задачи:
  - Обеспечение возможности выпуска дистрибутивов на systemd;
  - Поддержка systemd в основных сервисах;
  - Unit-файлы для распространённых сервисов.
- Поддержка работы в «облачных» окружениях Amazon EC2 и Microsoft Azure. Задачи:
  - Поддержка утилит для работы с окружениями (ec2-api-tools, azure-sdk-for-node);
  - Поддержка создания образов VM;
  - Публичные образы VM (как минимум на EC2).
- IPV6 для основных приложений
  - Обеспечение функционирования основных компонентов дистрибутива в IPv6 сети (тестирование);
  - Обеспечение настройки IPv6 в установщике / конфигураторах (Alterator).
- Поддержка установки на EFI (с помощью grub2-efi) в установщике.
- Новая, «беспроблемная» для пользователя реализация apepo:
  - Разработка способа перепаковки RPM ‘на лету’ в момент сборки задания в репозиторий;
  - Интеграция в сборочницу.
- «Карманы»:
  - Обновление инфраструктуры;
  - Распределённая сборочница, способная обрабатывать большое (сотни) количество репозиториев;
  - Оптимизация обработки задания (параллелизация проверок, сборки,...);

  - Межрепозиторные зависимости.
- Современные toolchain, autotools.
- GNOME3.
- Синхронизация ARM-репозитория.
- Поддержка deepsolver в режиме ‘technology preview’.

Что хотелось бы видеть, но не стоит ожидать в p7?

В Sisyphus существует достаточно много актуальных задач, которые в настоящий момент не решаются и, по всей видимости, будут отложены до следующих выпусков стабильных срезов. К таковым относятся:

- Новый alterator-vm:
  - Отказ от EVMS в установщике (либо его активная разработка);
  - Установка на шифрованные разделы;
  - Установка на iSCSI.
  - Простой интерфейс для простых десктопных установок.
- «Человеческая» программа установки ПО:
  - Установка прикладного ПО, каталог на основе desktop-файлов;
  - Поддержка установки проприетарных RPM (Google, Skype и т.п.).
- Поддержка ввода в домен AD.
- Централизованное управление серверами и десктопами.

Когда ожидать релиза?

Бранч будет выпущен тогда, когда будут достигнуты цели выпуска либо станет ясно, что эти цели могут быть достигнуты при разумных трудозатратах при работе в бранче. Выпуск альфа-версий дистрибутивов должен начаться уже летом, ориентировочным сроком фиксации бранча сейчас является начало осени.

Дмитрий Левин
Москва, ALT Linux

Проект: strace, `http://sourceforge.net/projects/strace/`

strace глазами апстрима

Аннотация

Напоминается история проекта, рассматриваются основные возможности *strace* и нововведения в недавно выпущенных версиях, приводятся примеры использования *strace*, дается обзор *ptrace API* и реализации *strace*, рассматриваются перспективы дальнейшего развития.

История проекта

- 1991-1992: Paul Kranenburg;
- 1993: Branko Lankester;
- 1993-1996: Richard Sladkey;
- 1996-1998: смутное время;
- 1998-2002: Wichert Akkerman;
- 2002-2009: Roland McGrath;
- с 2009.

Основные возможности и примеры использования

- Отслеживание порождаемых процессов: -f, -ff.
- Мониторинг процессов, существующих на момент запуска: -p.
- Сбор статистики: -c, -C, -S.
- Дополняющие описатели: -i, -r, -t, -tt, -ttt, -T.
- Формат вывода строк: -s, -x, -xx.
- Выбор множества отслеживаемых системных вызовов: -e trace=set.
- Глубина детализации множества системных вызовов: -v, -e abbrev=set, -e verbose=set, -e raw=set.
- Выбор множества отслеживаемых сигналов: -e signal=set.
- Мониторинг операций чтения/записи: -e read=set, -e write=set.
- Конвейеризация: -o.
- buildreq.

Нововведения с 2009 года

- 4.5.19: прозрачность, новые архитектуры, новые системные вызовы, улучшенные парсеры.
- 4.5.20: новый ключ -C, новые архитектуры, новые системные вызовы, улучшенные парсеры.
- 4.6: переход на новый Linux ptrace API для реализации отслеживания процессов, новые архитектуры, новые системные вызовы, улучшенные парсеры.
- 4.7: новые ключи (-y, -P, -I), новые архитектуры, новые системные вызовы, улучшенные парсеры.

Обзор реализации

- Linux PTRACE API.
- Отслеживание процессов.
- Парсеры системных вызовов.
- Реализация multiarch.

Перспективы дальнейшего развития

- Интерфейс PTRACE_SEIZE.
- Все более и более детализованные парсеры.
- Надежный multiarch.

Воронин Игорь Вадимович
Шатура, ИПЛИТ РАН

Проект: Образовательный проект — УМКИ `http://umki.vinforika.ru/`

Использование КУМИР-а для управления сенсорными сетями роботов

Аннотация

Работа посещена вопросам управления и связи множества разнообразных устройств в единую сенсорную сеть для выполнения различных миссий, при помощи которых в игровой и увлекательной форме происходит обучение базовым навыкам программирования и управления дистанционным способом различными роботизированными передвижными комплексами. Проект основан на идеологии свободного программного обеспечения и в нём используется программный пакет КУМИР для удобства и наглядности программирования передвижения роботов.

В настоящее время — мало кто может удивиться радиоуправляемой машинке. Но как вы поступите, если не имея опыта, вам интересно самому научиться управлять такими устройствами — передвижными колёсными платформами — которые по вашему приказу передвигаются в заданном направлении и там выполняют заранее предписанные действия? Например измеряют температуру, или влажность, или освещённость, или уровень радиации. А может быть одна машинка объезжает территорию, находит очаг дыма, другая приезжает в указанное место и заливает его водой. Хорошо, если такая экспериментальная площадка полностью вам видна, и вы с пульта можете управлять машинкой. Если машин — две, то вам скорее всего понадобиться второй пульт и может придётся позвать товарища в помощь. А если таких машин — 5, 10, 100? И действуют они не на открытой взору поверхности, а за углом или за каким-то препятствием?

Как договориться с устройством, чтобы радиосигнал от одного пульта не заставлял сразу все машинки двигаться в одну сторону — вправо, например? Как добиться, чтобы затраты энергии на управление были бы минимально возможными, а сеть таких машинок просуществовала максимально долго? Как организовать взаимодействие между операторами машинок — если их много? Кто решает, кому и в какой момент времени подавать нужную команду, чтобы не было конфликтов?

Очевидно, что в ситуации где много управляемых элементов, люди могут очень легко запутаться. Поэтому нужно, чтобы команды управления по радиоканалу адресовались на конкретное устройство, точно в нужный момент времени. Даже если машинка находится вне зоны прямой видимости базовой станции, нужно уметь позиционировать, с достаточно высокой точностью местонахождение каждой, с тем, чтобы зная в каком месте обнаружен очаг задымления, другая платформа — с брандспойтом, могла подъехать точно в это место и потушить пожар.

Так вот, чтобы легко и просто — как бы играя, научиться управлять такими устройствами, был разработан образовательный конструктор — который называется: УМКИ. Это Управляемый по радио каналу Машинный Конструктор Инновационный. Достоинство конструктора в том, что он комплектуется роботизированными платформами, которые связываются между собой в единую сенсорную сеть, на основе протокола ZigBee. Изучив этот курс, ученики получают базовые знания по управлению сначала одним устройством, потом группой устройств объединеных в распределённую беспроводную сенсорную сеть. Каждая роботизированная платформа оснащается либо набором сенсоров — датчиков для различных физических величин, либо исполнительными механизмами. Вы сможете заставить машинку двигаться по программе, ориентироваться на местности и выполнять разнообразные задания. Курс составлен так, чтобы на каждом его этапе было максимально интересно получать знания. Выполняя шаг за шагом задания к занятиям — вы будете проходить разнообразные миссии. Занимаясь с УМКИ, в игровой форме вы овладеете серьёзными инженерными знаниями которые сможете реализовать в дальнейшем, например в научных исследованиях.

Текущая версия УМКИ, базируется на элементах и различных инженерных схемах конструктора «Знаток». С её помощью можно с интересом использовать конструктор. Она укомплектована методическими материалами для преподавателей, которые включают в себя поурочное планирование. Аудитория курса — это начальные классы дополнительного школьного образования. Документация по работе с роботизированными машинками УМКИ наполнена научно-популярной информацией для детей и представлена в игровой форме — как процесс изучения окружающего нас мира. Вдобавок, для большей наглядности в ней используются интерактивные технологии. Все

миссии курса представлены в разных вариантах и связаны между собой логической составляющей.

Первым вариантом является «Миссия на Марс».

В ней ученикам необходимо реализовать миссию проводя как бы освоение Красной планеты. Первоначально собирается конструктор, при этом ученик получает базовые навыки по управлению роботом и попутно решает другие задачи (связанные со сборкой схем из технической документации). Цель миссии — создать колонию на Марсе.

Поскольку наборы конструкторов УМКИ, предполагается использовать в дополнительном образовании детей (чаще всего в форме кружков технического творчества), то вместе с изучением робототехники, предполагается коллективная работа нескольких детей и совместное решение задач, в форме стратегии. Таким образом, выполняя каждый этап миссии, дети собирают схему из конструктора, разбираются в принципе его работы, отвечают на вопросы по естествознанию (астрономии, физике, химии, биологии) и тем самым как бы увеличивают себе виртуальное жизненное пространство на Марсе.

К достоинству курса на базе УМКИ, можно отнести то, что это не отупляющая «стрелялка» или «бродилка», а интеллектуально развивающее пособие, которое позволяет через игровую форму, каждому ребёнку, занимающемуся с конструктором УМКИ, под руководством преподавателя, или самостоятельно получить в увлекательной форме базовые знания и умения их использовать по основным естественным дисциплинам: физике, математике, механике, электротехнике. Кроме того, курс разработан таким образом, что преподаватель может общаться с учениками не только в очной форме, но и давать задания, и проверять результаты выполнения в форме дистанционного обучения, используя платформу MOODLE

В базовой комплектации, роботизированный комплекс — конструктор УМКИ состоит из (см. рис.1):

1. 4-х колёсного вездехода, или иначе говоря — передвижной платформы с модулем zigbee, который позволяет связываться множеству платформ по стандарту IEEE 802.15.4 в единую, распределённую самоорганизующуюся сенсорную сеть.

2. Радио шлюза — который по USB соединяется с ПК и служит для отправки команд по радиоканалу и приёма ответов о выполненных процедурах.

Рис. 1: Состав конструктора

3. Программного обеспечения (ПО) на ПК, для управления передвижной роботизированной платформой — одной конкретной — выбираемой по MAK адресу, или множеством сообщества.

Вместе с тем, конструктор УМКИ может быть до укомплектован набором различных датчиков — которые унифицированы таким образом, что они легко и просто подключаются к базовой платформе и становятся доступными для выполнения разнообразных миссий.

Программное обеспечение для управления вездеходом состоит из следующих модулей:

1. Серверной части, написанного на C++, и скомпилированной под GCC, которая загружается при запуске системы в оперативную память и находится там в постоянно активном состоянии — для формирования команд управления передвижной платформой, и приёма ответов о выполненных командах.

Ниже приведён код обработки USB порта.

```
while (1){
    if (flag\_com == 0){
        fs = fscanf(fl\_com,{\textquotedbl}\%s{\textquotedbl},ss);
        if (fs{\textless}0) break;
    }
}
```

2. Модуля внешнего интерфейса, написанного на QT и предназначенного для управления мышкой или с клавиатуры для управления передвижением поворотной платформы.

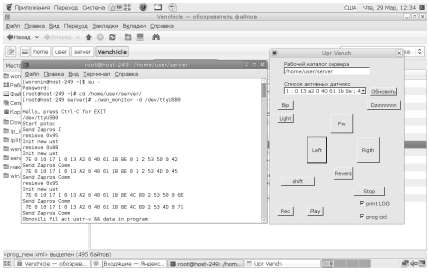

Рис. 2: Внешний вид монитора и программы управления

Ниже приведён код отрабатывающий при нажатии клавиши вправо.

```
void Vench::BPressRight()
{
    if(fl_play)return;
    if(fl_rec){
    b_tim_comm=GetTime();
    }
    if (checkOld->isChecked() == TRUE) {
    if (shiftPress==0) SendCommRight();
    else SendCommExtrRight();
}
    if (checkOld->isChecked() == FALSE) {
    if (shiftPress==0) SendCommRight1();
    else SendCommExtrRight1();
}
    PressRh=1;
}
```

Внешний вид монитора и программы управления приведен на рис. 2.

3. А так же свободно распространяемого пакета КУМИР — для программирования и задания пути движения передвижной платформы (см. рис. 3).

Код кумира

```
использовать Робот
алг
нач
. нц 3 раз
. . вниз
. кц
. вправо
. вправо
. нц пока сверху стена
```

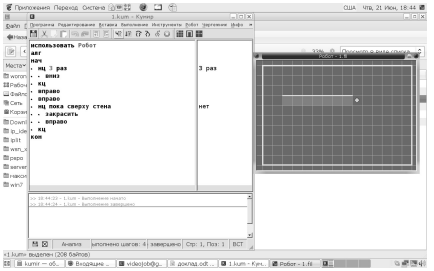

Рис. 3: Программа КУМИР

```
. . закрасить
. . вправо
. кц
кон
```

Следует также обратить внимание на то, что программный пакет КУМИР — рекомендован для использования в школах при подготовке к сдаче ЕГЭ.

В дальнейших планах развития этого проекта — переход от обучения программированию перемещениями в плоскости — 2D, посредством использование алгоритмов, заложенных при разработке данного УМКИ, на управление объектами, перемещающимися в 3D пространстве, например беспилотными летательными аппаратами.

Литература

[1] `http://wiki.laser.ru/index.php/`Беспроводные_распределённые_сенсорные_сети

Михеев Андрей Геннадьевич
Москва, Консалтинговая группа РУНА
Проект: RunaWFE http://wf.runa.ru/rus

Концепция межпроцессного взаимодействия в свободной системе управления бизнес-процессами и административными регламентами RunaWFE

Аннотация

В докладе рассказывается про реализованную в системе RunaWFE в начале 2012 года концепцию взаимодействия выполняющихся экземпляров бизнес-процессов при помощи сообщений

Системы управления бизнес-процессами и административными регламентами

Процессную организацию управления организацией используют для повышения эффективности работы организации. Эффективность повышается за счет оптимизации бизнес-процессов коммерческой организации или административных регламентов ведомства, появления возможности быстрого изменения бизнес-процессов в ответ на изменение условий деятельности предприятия, а также за счет повышения производительности труда работников..

Некоторое время назад выполнение бизнес-процессов в организациях производилось в основном косвенным образом — через изменение должностных инструкций, организационной структуры предприятия, прямые указания руководителей. Однако степень автоматизации современных предприятий позволяет реализовывать прямое выполнение бизнес-процессов в компьютерной среде. В этом случае в организации появляется аналог производственного конвейера, от которого можно получить увеличение производительности труда, сравнимое с тем, которое было получено от внедрения конвейера на производстве. Повышение производительности труда достигается вследствие того, что данный механизм позволяет работникам выполнять поступившие задачи, не отвлекаясь на:

- Получение от других работников необходимой для выполнения задания информации

- Передачу результатов своего труда другим работникам
- Изучение должностных инструкций

Все необходимое возникает перед работником на экране компьютера.

Для решения данной задачи разработан специальный класс компьютерных систем — системы управления бизнес-процессами и административными регламентами. Такие системы рассматривают бизнес-процессы и административные регламенты как потоки элементов работ. Выполнение процесса или регламента можно представить в виде перемещений точек управления потоков работ по определенным маршрутам между исполнителями в соответствии с заданными правилами. Последовательность выполнения элементов работ определяется схемой процесса, которую можно разработать в специальном редакторе. В узлах схемы система управления бизнес-процессами и административными регламентами раздает задания исполнителям и контролирует их выполнение.

Система RunaWFE

RunaWFE — открытая, масштабируемая, ориентированной на конечного пользователя система управления бизнес-процессами и административными регламентами. Система платформонезависима (написана на Java), распространяется под LGPL-лицензией.

Реализация межпроцессного взаимодействия в системе RunaWFE

Межпроцессное взаимодействие реализовано в RunaWFE при помощи сообщений. В соответствии с этой концепцией экземпляр бизнес-процесса может послать сообщение из одного узла схемы бизнес-процесса узлу другого экземпляра бизнес-процесса, или другому узлу того же самого экземпляра бизнес-процесса.

Для посылки и получения сообщения в палитру графических элементов были добавлены новые элементы «Отправить сообщение» и «Получить сообщение» (см. Рис. 1)

Для отправки и получения сообщений используется технология JMS. Узел-отправитель посылает сообщение сразу после прихода в него точки управления, после этого точка управления перемещается

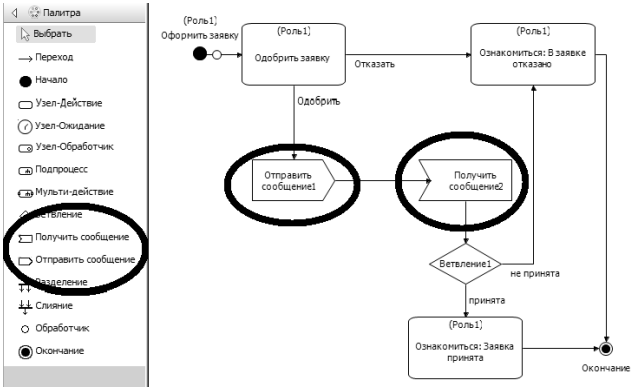

Рис. 1: Элементы отправки и получения сообщений в системе RunaWFE (вариант UML нотации)

в следующий узел, а узел-получатель ожидает сообщения, и точка управления может находиться в этом узле неопределенно долго.

В сообщении содержится следующая информация:

- кому предназначено сообщение
- передаваемые значения переменных бизнес-процесса
- время жизни сообщения (опционально)

В некоторых случаях одно сообщение может быть обработано несколькими получателями.

Адресат сообщения может быть определен следующим образом:

- по названию бизнес-процесса
- по названию узла бизнес-процесса
- по номеру экземпляра бизнес-процесса

Для задания этих параметров можно использовать переменные экземпляра бизнес-процесса

Структура данных сообщения позволяет задавать соответствие между переменными отправителя и получателя на любой из сторон (или с обеих сторон).

Время жизни сообщения предусмотрено для очистки сообщений из системы в случае, если для данных сообщений нет получателя.

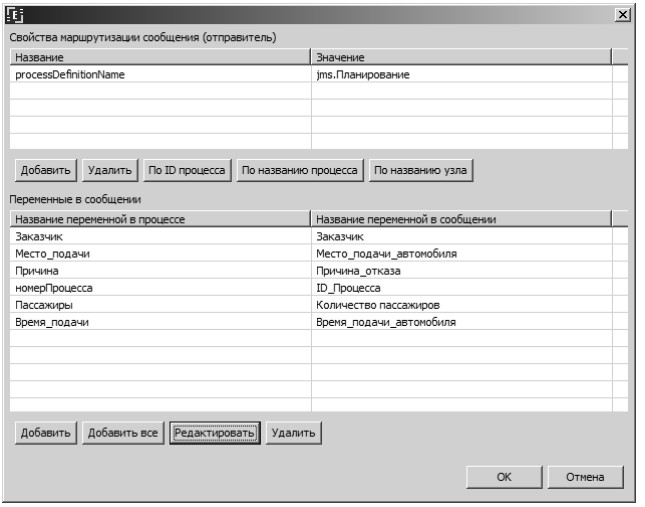

Рис. 2: Форма, связывающая переменные бизнес-процесса и параметры сообщения

Если сообщение послано по названию бизнес-процесса, а выполняющихся экземпляров этого бизнес-процесса не существует, то сообщение будет ждать появления первого экземпляра этого бизнес-процесса и передаст значения своих параметров первому появившемуся экземпляру в узле «Получить сообщение». Если, наоборот, существует сразу несколько выполняющихся экземпляров этого бизнес-процесса, то сообщение будет передано сразу всем этим экземплярам.

Литература

[1] Сайт проекта: `http://wf.runa.ru`

Радик Юсупов, Ленар Шакиров
Казань, ГК «Центр»

Проект: Zarafa http://www.zarafa.com/, http://openware.pro

Zarafa. Создание решений уровня Enterprise для российского бизнеса.

Аннотация

Бизнес избалован вниманием крупных и мелких IT-компаний, которые предлогают разные решения, которые различаются как по качеству, так и стоимости владения. Компания «Центр» в своих решениях использует СПО-решения, достигая оптимального соотношения цена/качество. Но решения СПО зачастую требует большой доработки и адаптации

Позвольте представиться, меня зовут Юсупов Радик Анасович. Я руководитель отдела Инфраструктуры Информационных Систем. Мой отдел занимается разработкой и внедрением инфраструктурных решений в Татарстане и регионах России.

Многие инфраструктурные проекты создавались нами с использованием СПО. Наши решения на основе ALTLinux работают например в ТатНефть, МЧС, министерствах и банках.

К нам часто обращаются с вопросом замены MS Exchange на линуксовый функциональный аналог. Мы протестировали несколько систем групповой работы, такие как ClearOS, Zimbra, Open-Xchange и др. и остановились на Zarafa.

Zarafa Collaboration Platform — это сервер групповой работы с открытым исходным кодом, предназначенный для обмена сообщениями и организации совместной работы сотрудников компании. Благодаря совместимости с MAPI, Zarafa является лучшей масштабируемой заменой Microsoft Exchange.

Zarafa предоставляет все необходимые возможности связки Exchange/Outlook:

- почта;
- адресная книга;
- задачи, записки, календари;
- персональные/публичные папки;
- планирование мероприятий.

Благодаря своей открытой архитектуре Zarafa легко масштабируема и включает в себя лучшие возможности в своем классе продуктов с открытым исходным кодом, такие как MTA (например, Postfix), базы данных SQL и сервера Apache. Среди возможностей Zarafa следует упомянуть следующие:

- поддержка Outlook 2000-2010;
- поддержка POP3/IMAP-клиентов;
- веб-интерфейс, привычный для пользователей Outlook Web Access;
- синхронизация с любым устройством, поддерживающим ActiveSync (iPhone, Android, Nokia);
- поддержка MTA;
- интеграция с LDAP-каталогами (в том числе Active Directory);
- хранение данных в MySQL;
- интеграция с антиспам- и антивирусными решениями;
- интеграция с открытыми решениями SugarCRM, O3Spaces, OpenERP и Alfresco, благодаря собственному фреймворку.

Так как ГК «Центр» в создании решений на Linux использует платформу ALTLinux, то мы сразу обсудили с разработчиком вариант включения в список поддерживаемых дистрибутивов — ALTLinux. Причем сразу оговорились, что адаптацию Zarafa для ALTLinux будем делать мы сами. В том числе и локализацию на русский язык.

Почти все патчи и локализацию разработчики приняли и добавили в свой код.

Патчи создавались при поддержке ALTLinux Team, за что им огромное спасибо!

В качестве LDAP-сервера был выбран 389 Directory Server. Это весьма мощный инструмент, предоставляющий возможности по управлению пользователями, группами и серверами средствами графического пользовательского интерфейса. Для удобства пользования этим удобным продуктом, мы произвели локализацию 389DS на русский язык и через ментейнера 389DS в сизифе — Виталия Кузнецова, передали переводы разработчикам. Надеемся, что в ближайшее время локализация на русский язык появится в новых версиях 389DS.

Итак. На данный момент мы имеем локализованный на русский язык сервер групповой работы Enterprise уровня, который легко интегрируется с 389DS, который в скором времени будет получит локализацию на русский язык. И все это в дружелюбном Российском дистрибутиве.

Имеющиеся внедрения показывают, что мы сделали востребование решение для Российского бизнеса.

Михаил Быков
Москва, http://diglossa.ru
http://diglossa.ru

Морфеус, морфологический анализатор латинского языка

Аннотация

Простой морфологический анализатор Морфеус построен на основе стандартных хорошо знакомых выб-технологий. В качестве основной структуры данных всегда используется JSON, в качестве базы данных — CouchDB. При разработке для генерации тестов (более 3000) использовалась хорошо себя зарекомендовавшая программа «words» У.Уитеккера.

Морфе́ус [1], морфологический анализатор латинского языка, построен с использованием стандартных методов веб-разработки на языке Ruby и CouchDb. Имеет систему тестов, «гарантирующих» корректность полученных результатов.

Большинство известных мне морфологических анализаторов построены на основе openSFT [2] — на конечных преобразователях. То есть на преобразовании строк с помощью регулярных выражений. Получающееся в результате чудовищное (много экранов) регулярное выражение компилируется в бинарник, способный очень быстро обработать большой объем текста. Происходит преобразование строка (анализируемое слово) → строка (морфологический анализ этого слова). В результате получается немодифицируемый, очень сложный бинарник. Очевидно, что он может быть только коммерческим продуктом. Это очень трудоемкий подход.

В Морфеусе я предлагаю, напротив, подход более человечный. Все данные о парадигмах языка хранятся в JSON и легко читаются и

модифицируются человеком. Обрабатываются не строки, а ruby (на сервере) или Javascript (на клиенте) объекты.

Алгоритм работы простого морфологического анализатора таков:

Каждая словоформа проходит через сито парадигм и связанных с ним правил, и на первом шаге 1) выявляются парадигмы, способные породить данную словоформу и возможные словарные формы, затем на втором 2) выбираются только те парадигмы, которые порождают действительно существующее в словаре слово и наконец 3) результат кешируется в БД. Вдобавок есть механизм заполнения базы словоформ исключениями, (терминами, не требующими никакого анализа, вне системы парадигм.). Все происходит в NoSQL-БД (поскольку json), а именно в CouchDB [3].

По сравнению со временем работы скомпилированного openSFT выражения, это медленный подход. Впрочем, нужно сравнивать не со временем выполнения скомпилированного SFT-приложения, а со временем его компиляции. А это также долго. Зато мгновенной результат — время обращения к базе.

Преимущества: 1) знакомые и обкатанные веб-технологии, все на http-rest запросах, все на стандартных библиотеках обработки ruby-хеша или js-объекта. И 2) крайняя простота и человекоразмерность данного подхода, 3) возможность постепенного улучшения анализатора, 4) возможность коллективной работы. И, очень важно, прозрачная возможность предварительной и заключительной (синтаксической, контекстной) обработки поступающего потока текста.

А именно, собственно морфологический анализатор является лишь частью процесса. Полностью процесс выглядит так:

Вначале клиент (браузер) посылает анализируемую строку на сервер. Сервер выполняет предварительный анализ строки, выделяя предложения, части предложения (главные, подчиненные), и обращается к Морфеусу за морфологическим анализом каждой словоформы. В результате получается упорядоченный массив морфологических данных. На следующем этапе выполняется частичный синтаксический анализ, а именно, находятся жестко связанные между собой слова, например предлог, сочетающийся только с определенным падежом, или причастие и определенная форма глагола esse, прилагательное плюс его существительное, и т.д. На этом же этапе мы можем отбросить «лишние» морфологические результаты, избавившись от неопределенности морфологического анализа (в том распространенном случае, когда морфологический анализ вырванной из контекста

словоформы дает несколько взаимоисключающих результатов). Наконец, приходит время terra incognita — полноценного синтаксического анализа. В результате же должен получиться граф синтаксических связей предложения. Но это — дело будущего, крайне заманчивое.

Отдельно я хочу отметить, что данный подход опирается на тесты, точнее, спецификации. И является примером стандартного BDD-style [4] способа разработки. Я использовал несколько источников «верного ответа» для создания спецификации. В основном, программу words Уильяма Уитеккера [5]. Каждая парадигма имеет исчерпывающий набор примеров, и разработка ведется до тех ровно пор, пока все полученные морфологические формы не совпадут с «верными» ответами. Кавычки я употребляю потому, что и в программу Уитеккера (лучшей из всех), и на сайте Персея [6], и на сайте Wiktionary [7] существует масса расхождений и между собой, и с известными мне учебниками. Улучшение работы Морфеуса сверх этих черновых BDD-спецификаций, т.е. полировка программы, потребует работы уже не программистов, но лингвистов и филологов и, видимо, годы труда.

В качестве вывода, я хотел бы предложить, чтобы на основе этого подхода, или схожих с ним,

1. объединились бы люди, разрабатывающие подобные морфологические анализаторы для всех близких нам языков, всех главных европейских языков, и всех языков будущего EAC, древних языков и т.д.
2. возникла бы общедоступная веб-служба с простым стандартным API для запроса анализа любой словоформы любого языка.

Некоторые подробности.

Морфеус написан на руби и имеет несколько удобных для работы классов с самообъясняющими названиями: Paradigm, Dict, Term (для исключений), Spec и некоторые вспомогательные классы, например Couch для работы с Кауч-DB.

```
> load 'morpheus.rb'
```

пример парадигмы:

```
> pp Morpheus::Paradigm.get "verb_act_fut-perf_ind"
[{:_id=>"8417feaa0669fda52d6d4d8d2b409302",
  :_rev=>"1-b2cd18b4cc2d616ed767fb7b47401940",
  :type=>"latin-paradigm",
```

```
  :pos=>"verb",
  :descr=>"act.fut-perf.ind",
  :dict=>"i",
  :flexes=>
   {:"sg.1"=>"ero",
    :"sg.2"=>"eris",
    :"sg.3"=>"erit",
    :"pl.1"=>"erimus",
    :"pl.2"=>"eritis",
    :"pl.3"=>"erint"}... часть вывода опущена для ясности]
```

пример вычисления возможных словарных форм:

```
>  pp Morpheus::Morph.guess "aquae"
{:nouns=>
  [{:pos=>"noun",
    :dict=>"aqua",
    :descr=>"prima",
    :gend=>"fem",
    :stem=>"aqu",
    :full=>"aquae",
    :sggen=>"ae",
    :morphs=>["pl.nom", "pl.voc", "sg.dat", "sg.gen"],
    :var=>"a"},
   {:pos=>"noun",
    :dict=>"aqua",
    :descr=>"prima",
    :gend=>"masc",
    :stem=>"aqu",
    :full=>"aquae",
    :sggen=>"ae",
....
 :adjs=>
  [{:pos=>"adj",
    :dict=>"aquus",
    :descr=>"primsec",
    :gend=>"fem",
    :stem=>"aqu",
    :full=>"aquae",
```

```
    :sggen=>"ae",
    :morphs=>["pl.nom", "pl.voc", "sg.dat", "sg.gen"],
    :grad=>"pos",
    :var=>"us-f"},
  ...
:parts=>[],
 :verbs=>
  [{:pos=>"verb",
    :dict=>"aquaeo",
    :descr=>"act.pres.imp",
    :stem=>"aqua",
    :morphs=>["sg.2"],
    :var=>2},
....
:infs=>[]}
```

прмер получения окончательного результата:

```
 >  pp Morpheus::Morph.parse "aquae" ;1
[{:pos=>"noun",
  :dict=>"aqua",
  :form=>"aquae",
  :morphs=>
   [{:gend=>"fem", :number=>"pl", :kase=>"nom"},
    {:gend=>"fem", :number=>"pl", :kase=>"voc"},
    {:gend=>"fem", :number=>"sg", :kase=>"dat"},
    {:gend=>"fem", :number=>"sg", :kase=>"gen"}],
  :type=>"latin-form"}]
```

Литература

[1] https://github.com/mbykov/morph-couch

[2] http://wiki.apache.org/couchdb/FrontPage

[3] http://www.openfst.org/twiki/bin/view/FST/WebHome

[4] http://rspec.info/

[5] http://ablemedia.com/ctcweb/showcase/whitakerwords.html

[6] http://www.perseus.tufts.edu/hopper/morph?la=la&l=rationi

[7] http://en.wiktionary.org/wiki/aquae#Latin

Шигорин Михаил
Киев, Massive Solutions Ltd

Проект: mkimage-profiles http://www.altlinux.org/Mkimage/Profiles/m-p

Макраме из дистрибутивов: mkimage-profiles

Аннотация

Когда дистрибутивов было ещё меньше сотни, а создавались они вручную — вопрос управления конфигурацией особенно не стоял: «большие» универсальные дистрибутивы создавались ровно в одном варианте. Потом начались работы по локализации и поддержке различных архитектур, которые породили немало форков сами по себе. Затем пошли производные «под задачу». А теперь широко доступна ещё и виртуализация, которая сильно подняла интерес к небольшим специализированным образам, сконструированным под конкретную задачу — поскольку появилась возможность под каждую частность выделить отдельный контейнер или VM.

Как мы уже обсуждали [1], раздвоение чего-либо (форк) может являться мощным средством как развития, так и уничтожения проектов — в зависимости от того, насколько приветствуется и удобно сведение результатов опять воедино (мерж).

mkimage-profiles на данный момент является экспериментальной разработкой по части уменьшения излишнего дублирования общей части конфигурации и вспомогательного кода, необходимого для формирования образов дистрибутивов и виртуальных окружений. Он создан на основе опыта расширения и рефакторинга альтовского mkimage-profiles-desktop и семейства схожих профилей плюс создания набора installer-feature-*, а также более ранних наработок (spt-profiles-*).

Проект стартовал в августе 2010 года по мотивам очередного рефакторинга m-p-d; после первоначальных экспериментов по определению траектории тогда же осенью был опубликован черновик, а ещё через год состояние оформилось в достаточной степени для «официального» представления. Коллектив разработчиков потихоньку увеличивается [2], а патчи либо кусочки кода приходят от ещё нескольких человек.

Использован для создания tech preview дистрибутива Simply Linux 7.

Поддерживается:

- наследование конфигурации на всех уровнях — от перечня пакетов до образа
- гибридные ISO с LiveCD, RescueCD, инсталятором или их комбинацией
- шаблоны виртуальных окружений OpenVZ
- образы VM
- архитектуры i586, x86_64, ARM
- пакетная база ALT Linux 6+ (возможно бэкпортирование для более ранних)

Возможны:

- при востребованности — PowerPC
- более ранняя пакетная база ALT Linux, как минимум до 5+
- иные пакетные базы (проведены эксперименты с openSUSE 11.4 и CentOS 6)

Литература

[1] Восьмая конференция разработчиков свободных программ: Тезисы докладов. Обнинск 25–26 июля 2011 года. М.: Альт Линукс, 2011. — 43 с.: ил.

[2] `https://www.ohloh.net/p/mkimage-profiles`

Денис Баранов
Санкт-Петербург, Etersoft

Проект: Gitum http://freesource.info/wiki/GitUM

Git Upstream Manager — средство для организации разработки и взаимодействия с upstream проекта.

Аннотация

В докладе рассматривается инструмент для работы с системой контроля версий Git — Git Upstream Manager (Gitum), позволяющий эффективно создавать и сопровождать ответвления от upstream-репозиториев; области его применения и принципы работы.

В большинстве случаев проект на основе свободного ПО строится на базе стабильного релиза, добавляется новый функционал. При попытке смержиться с upstream-веткой происходят конфликты, после исправления которых все наработки сползают в середину истории коммитов. В такой ситуации поиск «своих» патчей становится задачей не из лёгких. При ведении быстроразвивающихся проектов, таких как WINE (`http://winehq.org`), частые мержи и невозможность отделить «свои» и upstream-коммиты делают актуальным вопрос о корректном управлении и разборе кода. Проект Gitum (Git Upstream Manager), разработка которого была начата в 2011 году, призван помочь более легко и быстро ориентироваться в коде.

Git Upstream Manager — дополнительный режим работы Git, позволяющий вести ветки разработки со своими патчами и обновлять их с upstream-веток, при этом ведя общую историю изменений и поддерживая патчи в актуальном состоянии согласно состоянию upstream.

Краткая характеристика рабочего процесса с gitum.

Gitum имеет 5 рабочих веток:

1. Ветка апстрим репозитория — upstream.
2. Ветка с патчами наверху — rebased.
3. Ветка с непрерывной историей изменений — рабочая ветка — mainline.
4. Ветка с патчами в виде отдельных файлов — каждый коммит относится к состоянию репозитория — patches.
5. Ветка с конфигурационным файлом, где содержатся имена четырёх предыдущих веток — gitum-config.

Таким образом, разработчик всегда имеет актуальную версию upstream-ветки, ветку со всеми «своими» патчами и всей историей изменений в процессе разработке «ответвления».

Разработанная система Git Upstream Manager позволяет разработчикам с меньшем количеством усилий производить обновление своих продуктов до последних версий upstream и отсылать пачти в основную ветку разработки.

Алексей Чеусов
Минск, IHS

Проект: mk-configure https://github.com/cheusov/mk-configure

mk-configure — легковесная система сборки ПО

Аннотация

mk-configure is a lightweight replacement for GNU autotools written in bmake (portable version of NetBSD make) and traditional UNIX tools such as awk, grep etc. The main goal of the project is to make development process easier, for example mk-c provides only one top-level tool mkcmake instead of aclocal+automake+autoconf+autoheader. Other goals are portability, clean design, no heavy dependencies, simplicity, and of course "NO CODE GENERATION!".

В феврале 2009 года в результате очередного «обсуждения» в одной из эхоконференций сети FIDO различных вопросов, связанных со сборкой программных проектов, как то «чей make лучше», «безальтернативен ли autotools», «GNU make vs. BSD make» и прочих веселых вопросов я решил, что пришло время реализовать свои идеи и представления об идеальной с моей точки зрения (или близкой к идеалу) системе сборки ПО. Так, в ответ на вопрос «Зачем слова? Вы мне покажите...», родился еще один проект с открытым исходным кодом, один из десятков тысяч других, претендующих на место под солнцем. Называется он mk-configure (иногда я называю его mk-c или mkc). Его целью и задачей является упрощение написания приложений под UNIX-подобные ОС, и, ни много ни мало, составить конкуренцию таким известным и популярным проектам, как autotools, scons, cmake, а также другим, менее популярным, но решающим все ту же задачу — сборку программного проекта и установку его на систему пользователя для дальнейшего использования.

Наиболее часто задаваемый мне вопрос — «Зачем?». И действительно, зачем? Проектов аналогичного назначения — масса. К тому же «GNU autotools прекрасно у всех работает. Зачем что-то еще?»©. Я не стану отвлекаться на тему, насколько хорошо работает GNU autotools, и как много этих «всех», хотя у меня есть большие сомнения в том, что autoconf и automake вообще пригодны для сколь-нибудь разумного использования. Но в качестве ответа приведу небольшой пример кода из NetBSD, который в определенном смысле и явился первопричиной... Думаю, большинству пользователей операционных систем с открытым исходным кодом и даже пользователям закрытых коммерческих ОС известна программа GNU grep. Она входит в состав практически каждого Linux дистрибутива, является частью базовых систем NetBSD, OpenBSD, FreeBSD и многих других. Программистам, связанным с UNIX-подобными системами, хорошо известно, что в качестве системы сборки GNU grep использует GNU autotools. Менее известный факт, но о котором хорошо знают пользователи BSD систем: процессом сборки в *BSD системах управляет, по большому счету, одна программа, make(1), и центральным элементом дерева исходного кода этих систем являются Makefile-ы. Вот как выглядит src/usr.bin/grep/Makefile в NetBSD 5.1 (назовем его «BSD grep»).

```
PROG=   grep
SRCS=   binary.c file.c grep.c mmfile.c queue.c util.c

LINKS=  ${BINDIR}/grep ${BINDIR}/egrep  \
        ${BINDIR}/grep ${BINDIR}/fgrep  \
        ${BINDIR}/grep ${BINDIR}/zgrep  \
        ${BINDIR}/grep ${BINDIR}/zegrep \
        ${BINDIR}/grep ${BINDIR}/zfgrep

MLINKS= grep.1 egrep.1  \
        grep.1 fgrep.1  \
        grep.1 zgrep.1  \
        grep.1 zegrep.1 \
        grep.1 zfgrep.1

LDADD=  -lz -lbz2
```

```
WARNS=  4

.include <bsd.prog.mk>
```

Я убежден в том, что даже начинающий UNIX-программист легко поймет без документации, что написано в этом Makefile-е и для чего. На мой взгляд приведенный Makefile является идеальным сценарием сборки программы grep, поскольку не содержит ничего лишнего, а назначение того, что в нем записано, предельно понятно. И каким образом это «собирается»? Где правила компиляции и связывания, установки и чистки мусора? Всё необходимое для разработки, сборки и установки программы находится в библиотечном файле bsd.prog.mk. И единственной программой, необходимой для сборки, является команда make(1) NetBSD. На мой взгляд именно так и должны выглядеть инструкции по сборке программного проекта: кратко, понятно, описательно настолько, насколько это возможно. Можно сравнить самостоятельно это решение с тем, что мы имеем в GNU grep.

bsd.prog.mk вместе с другими bsd.*.mk библиотечными файлами часто называют mk файлами. К сожалению, набор Mk скриптов NetBSD не лишен недостатков. Он изначально не был расчитан на разработку программного обеспечения вне системы NetBSD. То же относится и к Mk файлам из FreeBSD и OpenBSD. Основная область их применения — базовые системы соответствующих ОС. Кроме того, они не предоставляют никакой функциональности для анализа программного окружения (функциональности GNU autoconf), а это крайней важно для разработки переносимого ПО. Перспектив по развитию NetBSD Mk файлов в данном направлении тоже незаметно, все-таки их основное назначение совсем иное.

Тем не менее, насмотревшись на «успехи» кодогенерации для сборки ПО на примере GNU autotools и cmake и получив представление о том, как может быть по-другому, я решил воспользоваться идеями, положенными в основу Mk файлов из *BSD систем, исправить недостатки, отмеченные выше, и разработать новую систему сборки. Собственно, основных идей всего три:

- декларативный подход описания сценария сборки, максимально удаленный от деталей реализации;
- «библиотечный» подход к реализации (по сути, включаемые в основной Makefile проекта фрагменты являются библиотеками make-а), т.е. полный отказ от кодогенерации;

- использование bmake (переносимый вариант NetBSD make(1)) и POSIX утилит в качестве языков реализации.

По сути, mk-configure является ответвлением (fork-ом) NetBSD mk файлов, именно их я взял за основу на начальном этапе разработки, хотя в настоящий момент mk-configure уже довольно далеко отошел от своего прародителя.

Цели проекта.

- Переносимость. К большому сожалению, в наше время многие разработчики для ОС Linux не придают значения вопросам переносимости ПО, считая другие UNIX-подобные системы или мертвыми или тупиковыми ветвями развития, считая Linux единственной «правильной» ОС. Я, тем не менее, придерживаюсь другой точки зрения, и потому одним из основных приоритетов для mk-c является переносимость на другие операционные системы, поддержка различных toolchain-ов. Так, например, поддерживаются такие ОС как Solaris, Linux, Haiku, Syllable, HP-UX, все варианты BSD систем и др. Среди поддерживаемых компиляторов можно отметить sunpro, clang, pcc, gcc, OSF/1 cc, HP-UX cc и др. Обеспечена поддержка «родной» программы связывания (ld) на всех поддерживаемых ОС, например Solaris и Darwin.

- Простота использования для разработчика. Давно прошло время, когда пользователи UNIX-подобных систем самостоятельно компилировали необходимое им ПО. В наше время инсталляция ПО происходит практически всегда посредством пакетного менеджера. Это касается *BSD, Linux, Solaris и других систем. Сегодня основным, если не единственным, пользователем ПО, компилирующим его, является куратор пакета в той или иной системе или дистрибутиве. Поэтому одной из главных задач mk-c я считаю предоставление удобного способа использования прежде всего для разработчика, а не для пользователя. Так, например, для сборки ПО используется всего одна команда — mkcmake (фактически обертка над bmake). Ничего подобного на http://en.wikipedia.org/wiki/File:Autoconf-automake-process.svg нет и никогда не будет. Между удобством для пользователей и разработчика, вообще говоря, нет противоречий, если бы не «сверхъестественный интеллект» сценариев сборки и конфигурации многих открытых проектов, который просто мешает

при пакетировании ПО, особенно в кросс-платформенных пакетных системах, поддерживающих различные ОС и toolchain-ы, таких, например, как pkgsrc.

- Минималистичность. В настоящий момент объем исходного кода mk-c, за исключением регрессионных тестов и документации, составляет порядка 4000 строк кода, и, я уверен, никогда не превзойдет 10000. При этом уже давно реализована, на мой взгляд, вся необходимая для разработки переносимого ПО функциональность.
- Гибкость. Несмотря на то, что сценарии сборки ПО при использовании mk-c как правило очень коротки и просты, поведение системы сборки можно легко изменить заданием определенных значений переменным окружения или make-а. Так, например, при сборке исполняемого файла можно установить переменные MKPIE или USE_RELRO в yes для передачи компилятору и линкеру соответствующих опций, а корневой каталог для установки задается переменной окружения PREFIX. Разработчик не должен заботиться о таких мелочах, как опции оптимизации компилятора или способ сборки разделяемых библиотек на незнакомых и далеких для него операционных системах. Это задача системы сборки.
- Настраиваимость на среду сборки. При разработке переносимого ПО часто возникает необходимость проверки окружения на наличие определенной функциональности или каких-либо особенностей. mk-configure предоставляет возможность, например, проверить наличие в libc реализаций функций strlcat или strlcpy, определить имя библиотеки, в которой реализованы функции семейства dlopen, определить размеры типов данных, проверить наличие необходимых заголовочных файлов, полей структур и переменных и т.п.
- Модульность. mk-configure является модульной системой и его функциональность легко расширяется. Так, например, в виде отдельных модулей реализованы поддержка языка программирования Lua, простейшая система Unit тестирования и поддержка формирования .deb пакета, пригодного для установки на ОС, основанную на deb пакетах.
- Поддержка кросс-компиляции. Думаю, здесь все ясно без лишних комментариев. Актуальность кросс-сборки связана с наблю-

даемым в течение последних лет 5–7 бумом на переносимые устройства, оснащенные полноценными ОС.

Более подробно с функциональностью mk-configure можно ознакомиться в документации проекта. Также стоит отметить, что в состав ПО входит большое количество примеров использования.

На мой взгляд, за три года, прошедших с начала не очень активной разработки, мне удалось создать проект, по своим характеристикам не уступающий многим широко используемым системам. Он может быть использован в серьезных проектах, при этом существенно экономя время разработки по сравнению с аналогами и в некоторых случаях предоставляет уникальную, отсутствующую в конкурирующих проектах функциональность. Нельзя сказать, что проект полностью завершен, но активная фаза его разработки позади, уже имеющейся функциональности достаточно для решения очень широкого класса задач. Не могу не отметить, что проект получился весьма небольшим по размерам и обладает богатым набором регрессионных тестов, что, с одной стороны, позволяет мне не тратить на его разработку слишком много времени, а с другой, дает возможность пользователям легко убедиться в его работоспособности и, при необходимости, легко исправить ошибки в случае их обнаружения. Среди проблем, стоящих на пути более широкого применения проекта основной, мне кажется, является слабая распространенность bmake за пределами NetBSD несмотря на то, что в целом он ничем не уступает GNU make-у, стандарту де факто в мире Linux, а во многом значительно его превосходит. На момент написания настоящей статьи пакет mk-configure имеется в следующих системах: ALT Linux, Arch Linux, RHEL (repoforge) DragonFlyBSD, FreeBSD, NetBSD и MirBSD. Авторские примеры использования в реальных проектах доступны на странице `http://github.com/cheusov`

Игорь Власенко
Киев, ALT Linux

Проект: Облачно-дружественная распределенная инфраструктура для сервисов автоматизации ALT Linux Team
http://www.altlinux.org/Категория:Packaging_Automation

Автоматизация сборки и сопровождения пакетов для дистрибутива. Технологии и инфраструктура.

В настоящее время в OpenVZ контейнерах autoports.altlinux.org, autoimports.altlinux.org, repocop.altlinux.org, watch.altlinux.org развернуты

- 3 сервиса repocop — для репозиториев Sisyphus, t6 и autoimports;
- сервисы Cronbuild, Cronport, Croncopy;
- сервис Watch;
- репозитории и сервис Autoports для p5 и t6;
- репозиторий Autoimports для Sisyphus;

Часть этих сервисов еще находится в разработке. Другие сервисы автоматизации еще только планируются к развертыванию.

В докладе «Облачно-дружественная распределенная инфраструктура для сервисов автоматизации ALT Linux Team» будет рассказано о развертываемой и находящейся в процессе развертывания инфраструктуре сервисов автоматизации, предназначенной для майнтайнеров и пользователей репозиториев ALT Linux.

Будет представлена новая распределенная архитектура repocop, теперь позволяющая отслеживать файловые конфликты пакетов при обновлении с одного дистрибутива на другой; планы по разворачиванию сети автономных нод импорта и генерации пакетов; альтернативный подход к организации разделения Sisyphus на компоненты и поддержка множественных компонент репозитория.

Даниил Михайлов, Виталий Липатов
Санкт-Петербург, Etersoft
http://wiki.etersoft.ru/Epm

Единая команда управления пакетами EPM

Аннотация

В докладе сделан обзор возможностей применения и особенностей реализации единой команды управления пакетами, которая при интерфейсе, похожем на rpm и apt одновременно, способна выполнять необходимые операции на любой платформе.

Для выполнения операций над программами и компонентами системы часто используются так называемые пакеты. Пакеты — это в первую очередь файловые архивы, в которых хранятся все необходимые для работы программы файлы вместе с путями к ним. Поскольку программы используют различные библиотеки, хранящиеся в других пакетах, появляется понятие «зависимость пакета».

Хранением списка доступных и установленных пакетов, контролем их зависимостей и выполнением операций по установке и удалению занимается менеджер пакетов.

Основные функции менеджеров пакетов:

- установка пакетов
- удаление пакетов
- получение списка пакетов
- поиск пакета по названию
- обновление системы

Рассмотрим команды для установки пакетов на разных дистрибутивах (см. табл. 1):

Таблица 1: Команды для установки пакетов

Система	Низкий уровень	Верхний уровень
ALTLinux	rpm -Uvh пакет	apt-get install пакет
Mandriva	rpm -Uvh пакет	urpmi пакет
SUSE	rpm -Uvh пакет	zypper install пакет
Fedora/RH	rpm -Uvh пакет	yum install пакет
Debian/Ubuntu	dpkg -i пакет	apt-get install пакет

Таблица 1 — продолжение

Система	Низкий уровень	Верхний уровень
MOPSLinux	—	mpkg install пакет
Slackware	installpkg пакет	—
FreeBSD	pkg_add пакет	pkg_add -r пакет
Gentoo	—	emerge -uD пакет
Solaris	pkgadd -d пакет	pkginstall пакет
ArchLinux	pacman -U пакет	pacman -S пакет

Такое разнообразие команд, выполняющих, с точки зрения пользователя, одну и ту же функцию, не может не огорчать. Особенно такое положение дел расстраивает тех, кому приходится пользоваться большим количеством дистрибутивов — разработчиков программ, используемых на различных системах или сотрудников фирм, предоставляющих хостинг.

Чтобы избавить администратора от необходимости вспоминать нужную команду в зависимости от используемой ОС, была создана команда epm, унифицирующая синтаксис менеджеров пакетов. Далее приведён пример выполнения некоторых операций над пакетами с помощью epm. Параллельно приведён пример выполнения тех же операций в ALT Linux, откуда можно видеть, что команды epm во многом схожи с командами apt и rpm (см. таблицу 2).

Таблица 2: Команды epm

Описание операции	Команда epm	Команда в ALT Linux
Установка пакета	epm install пакет	apt-get install пакет
Удаление пакета	epm remove пакет	apt-get remove пакет
Поиск пакета в базе	epm search текст	apt-cache search текст
Проверка наличия пакета	epm -q пакет	rpm -qa (pipe) grep пакет
Список установленных пакетов	epm -qa	rpm -qa
Принадлежность файла к пакету	epm -qf (file)	rpm -qf (file)
Список файлов в пакете	epm -ql пакет	rpm -ql пакет

После установки epm можно забыть о различиях в пакетных менеджерах. Простота и схожесть синтаксиса epm с синтаксисом apt и rpm помогут сделать процесс перехода на epm максимально простым и быстрым.

Геннадий Кушнир
Москва

Проект: Электронный классный журнал РУЖЭЛЬ www.rujel.net

Web-сервисы обмена данными в РУЖЭЛЬ

Аннотация

Проприетарные разработчики систем электронных журналов (ЭЖ) ведут борьбу за региональных чиновников, которые склонны внедрять единообразные решения на весь регион. Поэтому разработка открытых протоколов обмена данными для коммерсантов не приоритетна. Но в некоторых регионах право школ на собственный выбор варианта ЭЖ не оспаривается. При этом возникает задача единообразного сбора данных на уровне органов управления. СПО-разработчик РУЖЭЛЬ предлагает решение в виде открытого протокола обмена данными посредством web-сервисов.

Работа по открытым протоколам позволяет снять массу проблем, связанных с конкретными программными и архитектурными решениями разных вариантов ЭЖ. Поскольку сегодня работа ЭЖ в сети Интернет является нормой, web-сервисы являются наиболее логичным инструментом организации процедуры обмена данными.

В противовес применяемым рядом разработчиков техникам передачи данных средствами СУБД, сервисный механизм стимулирует делать обмен данными интеллектуальным: ЭЖ передает не «сырые данные», а подготовленные к формированию нужного отчета агрегированные данные, что существенно снижает нагрузку на центральную систему, сетевой траффик, и помогает избежать проблем с законом о ПДн.

Одним из узких мест такого подхода является отсутствие в нормативной базе стандартных требований к составу и формату данных, формам их представления. Это требует готовности оперативно подстраиваться под новые запросы.

Для решения этой задачи был разработан протокол обмена данными через web-сервис, реализующий идеологию REST. Ключевой особенностью данного решения является возможность полностью описать требуемый отчет в запросе.

Запрос напоминает логику SQL и опирается на обобщенную модель данных ЭЖ. На наш взгляд, каждый ЭЖ можно привести к

этой обобщенной модели и реализовать предложенный нами сервис. Это позволит на центральной точке сбора данных обеспечить необходимую сводную отчетность с разных ЭЖ от разных производителей. Издержки на создание такого web-сервиса не являются значительными для любого разработчика.

Александр Загацкий
Москва, Независимый разработчик

Проект: WriteAt http://writeat.com

WriteAt: доступные книги для читателей и писателей

Аннотация

Открытый проект «WriteAt» ставит перед собой цель — сделать книгу доступнее как для читателей, так и для писателей. Доступность и простота создания книг позволит раскрыть творческие способности большего количества людей, а так же сделает книгу, особенно учебную, доступнее. Проект включает в себя бесплатный и открытый инструментарий для написания и подготовки к публикации книг, руководств, инструкций. Проект «WriteAt» построен на открытых технологиях. В основе лежит современный формат разметки pod6. «WriteAt» является opensource стартапом.

Современная экономика направлена на воспитание в нас потребителей. Поэтому с понятием «доступный» связывают в первую очередь низкую стоимость какого-либо товара или услуги. Например: доступное жилье, доступные продукты питания, доступные книги.

Однако насколько легко построить дом или написать книгу? Ответ на данный вопрос зависит от доступности орудий производства и технологий. Если построить дом просто для строителя, то и квартиры в этом доме будут доступны большему количеству жильцов.

Проект «WriteAt» является бесплатным и простым инструментом для создания книг в электронном и печатном виде.

Пишутся книги в формате pod6. Pod6null — простой и лаконичный язык разметки.

Для создания файлов в этом формате подойдет любой текстовый редактор.

Шаблон книги выглядит следующим образом:

```
=ЗАГОЛОВОК Моя очередная книга
=ПОДЗАГОЛОВОК или как просто делиться знаниями
=АВТОР  Вася Пупкин
=ОПИСАНИЕ
В этой книге описаны основные правила, которые позволят
наиболее просто поделиться своими знаниями и опытом.
=CHAPTER Вступление
Начало книги !
```

Специальные директивы начинаются со знака = и следующим за ним названия блока. Используются следующие блоки:

```
=ЗАГОЛОВОК, =TITLE   заголовок книги
=ПОДЗАГОЛОВОК, =SUBTITLE подзаголовок
=АВТОР, =AUTHOR автор
=ОПИСАНИЕ, =DESCRIPTION краткое описание книги
=ГЛАВА, =CHAPTER название главы
```

Основной чертой формата pod6 является его расширяемость. Благодаря этому стала возможной вставка изображений и разбиение книг на части.

Иногда бывает удобно, чтобы главы располагались в отдельных файлах. Для этих целей используется блок =Include :

```
=Include src/preface.pod6
=Include src/basics.pod6
=Include src/operators.pod6
=Include src/subs-n-sigs.pod6
```

Такой прием облегчает совместную работу над книгой нескольких авторов.

Для вставки изображений используется следующий блок:

```
    =Image img/bold1.jpg
```

«WriteAt» позволяет облегчить поддержку технической документации, особенно публичную. Дополнительный блок ИЗМЕНЕНИЯ (CHANGES) позволяет вести журнал изменений документа. Он располагается в самом начале документа и необходим для описания основных изменений. Например, изменения в API сервиса.

```
=begin ИЗМЕНЕНИЯ
 Jun 6th 2012(v0.2)[zag]   Предисловие
 May 27th 2012(v0.1)[zag]   Начальная версия
=end ИЗМЕНЕНИЯ
```

Для установки последней версии пакета writeat для Ubuntu необходимо выполнить команды:

```
sudo add-apt-repository ppa:zahatski/ppa
sudo apt-get install writeat
man writeat
```

«Writeat» на данный момент находится в начале развития. Он бесплатен и открыт `https://github.com/zag/writeat`. В его основе лежат открытые технологии и форматы.

Наличие бесплатного и простого способа создать книгу позволяет решить задачу доступности книг как для читателей, так и для писателей. Это значит, что будущее, в котором учебники бесплатны, вполне реально.

Литература

[1] Брукс, Ф., Мифический человеко-месяц, или как создаются программные системы., `http://www.lib.ru/CTOTOR/BRUKS/mithsoftware.txt`