RPM5: новый формат и инструментарий распространения приложений для ОС Linux

RPM5: a novel format and tools to distribute Linux applications

Денис Силаков
Отдел автоматизации разработки
ЗАО «РОСА»
Москва, Россия
Denis.silakov@rosalab.ru

Abstract — **A distinctive feature of many Linux distributions is an original approach to the task of software management, which implies formation of software packages and usage of special systems to manage these packages – in particular, install, remove or update any of them. A common belief is that such systems are just archivers with minor additional features. However, modern package managers are quite complex systems that provide a lot of features besides just putting application files into archive and managing dependencies between different packages. Constantly increasing complexity of software gives rise to different improvements in this area. Nowadays package managers provide various possibilities to control the process of package installation or removal by allowing launching of arbitrary scripts embedded in the package itself, allow users to work with packages through the network, and so on. New features constantly are suggested and implemented.**

This paper describes RPM5 — a promising package management system based on the famous and widely used RPM package manager. We present an overview of already implemented improvements and discuss possible future directions. The improvements fall into two categories: the ones aimed to simplify creation of packages and the ones that speed up the user-side tools and enhance their functionality in different ways.

Improvements for developers include file triggers, a novel approach for localization of package descriptions, possibility to embed different useful utilities and libraries in the RPM itself, as well as various improvements aimed to automate the process of package creation and reduce number of manual actions.

Among changes valuable for users, we can highlight transactional package management, usage of parallelism to speed up different actions, integration with higher-level package managers and improved handling of configuration file updates. All these changes are aimed to make installation, update and removal of packages considerably faster and more flexible, at the same guaranteeing high reliability of these actions.

Keywords; Linux, package management, RPM

Аннотация — **Отличительной чертой многих представителей семейства ОС 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.**

Статья посвящена RPM5 - одной из перспективных и активно развивающихся систем управления пакетами, являющейся ответвлением широко распространенного инструментария RPM. Приводится обзор нововведений, реализованных в рамках проекта RPM5, а также улучшения, которые планируется добавить в будущем. Рассматриваются как изменения, нацеленные на облегчение создания пакетов, так и модификации, повышающие скорость работы инструментария и расширяющие его функциональность для конечных пользователей.

Ключевые слова; Linux, управление пакетами ПО; RPM

I. ВВЕДЕНИЕ

Операционная система GNU/Linux — это не просто ядро Linux и инструментарий и библиотеки GNU. Для получения полнофункциональной системы, производители конкретных вариаций Linux предоставляют пользователям согласованный и самодостаточный набор программного обеспечения, включающий не только системные компоненты и библиотеки, но и различные приложения, средства конфигурирования системы и прочее. Подобный набор ПО образует *дистрибутив* Linux.

Характерной чертой большинства дистрибутивов Linux является использование развитых систем управления пакетами — инструментария для управления процессом установки, удаления, настройки и обновления ПО, входящего в дистрибутив. Использование подобных систем подразумевает распространение приложений и прочих программных компонентов в виде *пакетов* — файлов специального формата, включающих в себя архив с файлами приложения, а также различные метаданные — описание пакета, цифровую подпись, список компонентов, от которых зависит приложение, и многое другое.

В настоящее время наибольшее распространение в корпоративном секторе получили дистрибутивы Linux, использующие менеджер пакетов RPM и одноименный формат пакетов. Изначально RPM расшифровывался как RedHat Package Manager, однако в настоящее время общепринятым является использование рекурсивного акронима RPM Package Manager.

К числу дистрибутивов, использующих RPM, относятся, в частности, RedHat Enterprise Linux и SUSE Linux Enterprise Server. RPM включен в стандарт Linux Standard Base в качестве унифицированного формата для распространения ПО в Linux. В таких системах, как Ubuntu и Debian, использующих другой популярный формат — Deb, — пакеты RPM могут быть установлены с помощью утилиты *alien*.

История RPM насчитывает уже более 15 лет, причем последние десять лет в большинстве дистрибутивов используется четвертая версия формата и инструментария. Безусловно, развитие RPM4 не стоит на месте; как в сам формат, так и в инструментарий время от времени вносятся различные дополнения и улучшения. Однако все эти изменения носят эволюционный характер. В то же время за последние годы в мире ИТ произошли существенные перемены, которые непозволительно игнорировать в такой ключевой технологии, как RPM.

В 2007 году Джефф Джонсон, на протяжении десятилетия являвшийся ведущим разработчиком RPM, инициировал разработку RPM5 [1] — ответвления оригинального проекта, нацеленного на активное добавление новых функциональных возможностей и избавление от устаревших (даже если последнее ведет к частичной потере обратной совместимости со старыми версиями инструментария и собранными с их помощью пакетами). Сегодня RPM5 уже готов к промышленному использованию и применяется в дистрибутивах Unity, Ark Linux, POCA, Mandriva и ряде других проектов, оценивших перспективы обновленного механизма управления пакетами.

Статья состоит из двух частей. В первой части статьи представлены нововведения RPM5 (как уже реализованные, так и планируемые), призванные упростить жизнь конечным пользователям. Вторая часть посвящена изменениям, которые будут оценены разработчиками, создающими пакеты в формате RPM.

II. RPM5 для пользователей

Какие бы изменения не вносились в различные составляющие дистрибутива, все они так или иначе нацелены на получение дополнительных преимуществ с точки зрения пользователей. Не является исключением и RPM5. Часть изменений, предлагаемых новым поколением инструментария, нацелена на упрощение процесса разработки дистрибутива и формирования пакетов. Подобные улучшения видны пользователям косвенно — например, за счет более быстрого внедрения новых технологий. Однако RPM5 содержит и наработки, которые будут видны пользователям непосредственно.

A. Транзакционное управление пакетами

Одним из главных нововведений RPM5 стало транзакционное управление пакетами, в рамках которого установка, обновление или удаление каждого пакета представляется как атомарная транзакцию (по аналогии с транзакциями СУБД), которую, в случае необходимости, можно откатить.

Для реализации такой возможности, RPM5 отслеживает все системные вызовы, осуществляемые в процессе обработки пакета. При этом отслеживаются действия, выполняемые не только самим инструментарием (размещение файлов в системе, обновление базы пакетов и прочее), но и скриптами, входящими в пакеты. Для действий, выполняемых инструментарием, возможность отката гарантирована; для скриптов в общем случае это сделать не представляется возможным — единственным универсальным решением было бы создание снимка всей системы, но такой подход не представляется разумным (во всяком случае, в рамках инструментария RPM).

B. Использование параллелизма

Одним из наиболее ресурсоемких действий при установке многих дистрибутивов, распространяющихся в виде набора прекомпилированных пакетов ПО в формате RPM, является установка и обновление этих пакетов. Развертывание современного дистрибутива, состоящего из нескольких тысяч пакетов, может занять порядка часа даже на современных машинах. Обновление дистрибутива до новой версии посредством обновления всех пакетов занимает схожее время.

Средний размер типичного дистрибутива за последние годы существенно вырос — увеличилось как количество пакетов, так и их размер. Поэтому, несмотря на стремительный рост производительности компьютеров, время установки и обновления дистрибутивов уменьшилось не сильно. В то же время современные машины предлагают способ увеличения быстродействия приложений, пока еще никак не задействованный в инструментарии RPM — речь идет о возможности параллельного выполнения подзадач одного процесса на разных физических ядрах процессора.

В случае RPM, распараллеливание задач возможно на достаточно высоком уровне — например, инструментарий способен принимать на вход сразу несколько пакетов, но обрабатываются они по очереди. В то же время некоторые действия, осуществляемые при такой обработке, абсолютно независимы и для разных пакетов могут выполняться параллельно. К таким действиям относятся:

- Проверка подписей пакетов.
- Вычисление и проверка контрольных сумм заголовков пакетов и содержащихся в них файлов.
- Проверка наличия в системе необходимых пакету зависимостей (при условии, что уже определено, какие из этих зависимостей удовлетворяются пакетами, переданными на вход инструментарию вместе с обрабатываемым пакетом).

В настоящее время в RPM5 уже реализован механизм параллельной проверки подписей и контрольных сумм,

ведутся работы над параллельной проверкой зависимостей. Текущая реализация основана на использовании OpenMP, хотя для других действий более удобным может оказаться использование многопоточности (в частности, функций из набора POSIX Threads). Эксперименты на репозиториях дистрибутива РОСА показали, что использование разработанных механизмов на машине с двумя ядрами ускоряет перечисленные выше проверки в полтора раза. Использование параллельного вычисления контрольных сумм посредством утилиты rpmdigest с опцией --alldigests (для каждого пакета вычисляющей до сотни различных контрольных сумм) на четырехпроцессорной машине позволило достичь семикратного роста производительности.

Теоретически, возможно одновременное выполнение над несколькими пакетами и ряда других действий — например, непосредственной установки пакетов в систему, включающей разархивирование содержимого пакетов и их размещение в файловой системе. Однако одновременное выполнение подобных действий может потребовать соответствующей доработки инструментов, работающих поверх RPM и специфичных для конкретных дистрибутивов. В частности, во многих системах используются триггеры, срабатывающие при появлении в системе определенных файлов. Например, при появлении нового файла ядра можно автоматически прописывать это ядро в меню загрузчика системы. Перед реализацией параллельного выполнения некоторых действий в инструментарии RPM, необходимо изучить готовность существующих триггеров к работе в таких условиях.

Помимо одновременного выполнения действия над разными пакетами, возможно распараллеливание действий и на более низком уровне. Так, одним из самых затратных действий при установке и обновлении является разархивирование содержимого пакета. Для сжатия содержимого пакета, в RPM применяется один из инструментов gzip, bzip2 или xz (реализующий сжатие по алгоритму LZMA). Для gzip и bzip2 существуют реализации, способные распаковывать архив в несколько параллельных потоков — pgzip и pbzip2 соответственно. Однако подобные возможности в RPM в настоящее время не используются, и распаковка остается сугубо однопоточным процессом.

Возможность использования параллельных алгоритмов распаковки ведутся, однако их внедрение сопряжено с существенными техническими сложностями. В частности, многие программы, использующие API RPM, получают информацию о том, какая часть архива уже распакована, посредством соответствующих обратных вызовов. Дизайн существующего механизма обратных вызовов *rpmtsCallback()* не рассчитан на параллельную распаковку, но при этом достаточно сложен и используется многими программными компонентами. К тому же возможность параллельной распаковки архива отсутствует в случае xz, а этот способ становится все более популярным благодаря более высокой степени сжатия и скорости разархивирования в однопоточном режиме.

Наряду с алгоритмами распаковки, исследуется возможность использования алгоритмов параллельного сжатия, что позволит ускорить процесс формирования пакетов в сборочных средах дистрибутивов и на машинах разработчиков. Такая возможность существует как для gzip/bzip2, так и для xz. Также для разработчиков представляется полезной возможность параллельного создания нескольких подпакетов из разных подмножеств одного большого приложения.

C. Интеграция с высокоуровневыми менеджерами пакетов

В наши дни пользователи Linux редко обращаются непосредственно к инструментарию RPM, работающему в командной строке на уровне отдельных пакетов и их групп. Традиционной схемой для большинства дистрибутивов является использование дополнительных программ управления пакетами, работающих поверх RPM и вызывающих этот инструментарий для обработки конкретных пакетов.

Высокоуровневые менеджеры пакетов — такие как Yum, Apt, Zypper или Urpmi — работают с *репозиториями* — наборами пакетов, сгруппированных согласно некоторому принципу. Например, создатели дистрибутива могут предусмотреть отдельные репозитории для игр, для офисных программ, либо для всех пакетов, не поддерживаемых официально, но собираемых и обновляемых членами сообщества. способны подключать конкретные репозитории (в том числе находящиеся на удаленных машинах), получать информацию о пакетах в них, производить поиск среди пакетов по имени, описанию либо предоставляемым файлам и многое другое.

В силу исторических причин, высокоуровневые менеджеры пакетов частично дублируют функциональность самого RPM. Например, при установке пакета из удаленного репозитория Urpmi, Yum и другие проверяют, от каких пакетов он зависит и какие из этих пакетов отсутствуют в системе. Все недостающие зависимости также извлекаются из репозитория. После этого производится установка всех загруженных компонентов средствами RPM, который снова производит проверку зависимостей для каждого устанавливаемого пакета.

Для избавления от подобного дублирования, необходима более тесная интеграция RPM с вышестоящими программами. Одним из возможных путей решения вопроса является вызов инструментария RPM из высокоуровневых программ с явным отключением проверки зависимостей. Однако при таком подходе необходимо гарантировать, что вызывающая программа проверила *все* возможные зависимости; поскольку код программ обычно развивается параллельно с RPM, то в реальной жизни дать подобные гарантии не всегда возможно. Более продуктивным представляется выделение некоторых возможностей RPM в отдельную библиотеку, которая могла бы использоваться как самим RPM, так и другими утилитами. Создание таких библиотек в настоящее время находится на стадии идей и проектирования.

D. *Управление конфигурационными файлами приложений*

При обновлении пакета до более новой версии, существующие в системе файлы перезаписываются новыми. Однако для некоторых файлов это может оказаться нежелательно — например, для файлов конфигурации, который пользователь мог изменить для своих целей. Использовать старые файлы при этом не всегда возможно, поскольку в новой версии программы их формат мог измениться. В существующих системах управления пакетами (не только в RPM) этот вопрос решается сохранением старых версий файлов, явно помеченных создателями пакета как конфигурационные. При этом сохраняется только одна версия, более старые удаляются.

Такой подход позволяет избежать многих проблем, связанных с поддержкой конфигурационных файлов, однако при большом количестве таких файлов приводит к замусориванию файловой системы. Кроме того, пользователям приходится вручную (либо с помощью дополнительных программ) изучать разницу между файлами и приводить новую конфигурацию в работоспособное состояние. Наконец, в ряде случаев хранение только предыдущей версии файла может оказаться недостаточным.

Разработчики RPM5 предложили более универсальный способ отслеживания изменений фалов, фактически заключающийся в помещении их в систему контроля версий (СКВ), поддерживаемую непосредственно инструментарием RPM. Благодаря своей гибкости, инструментарий может быть собран с поддержкой одной из библиотек, позволяющих реализовать весь необходимую функциональность СКВ — например, libgit2 или libsvn.

Использование СКВ для отслеживания истории файлов снимает проблемы, перечисленные выше — количество сохраняемых версий файла практически не ограничено, файловая система при этом не замусоривается, а функциональность СКВ позволяет не просто получать разницу между определенными версиями, но и осуществлять их слияние. В настоящее время уже проводятся эксперименты по использованию в RPM5 библиотеки libgit2 для хранения истории изменений конфигурационных файлов.

E. *Работа с сетью*

Изначально в RPM не было заложено никаких возможностей работы по сети — например, инструментарий позволял получать детальную информацию по заданному пакету (версию, описание, список файлов и другое), только если этот пакет либо установлен в системе, либо если файл с пакетом находится на той же машине, на которой работает инструментарий. Обрабатывать файлы с удаленных машин инструментарий RPM долгое время был неспособен. Вся подобная работа традиционно возлагалась на более высокоуровневые менеджеры пакетов. Однако менеджеры пакетов работают только с репозиториями дистрибутивов и обычно не предусматривают возможности обращения к произвольным файлам на удаленных машинах.

RPM5 и современный RPM4 содержат встроенные средства обращения к удаленным машинам по протоколам FTP и HTTP и поддерживают установку/обновление пакетов, а также получение информации о них, непосредственно с HTTP или FTP сервера. В рамках проекта RPM5 ведутся работы над расширением списка действий, в которых можно использовать информацию с удаленных машин. В перспективе планируется полностью убрать различие между локальными и удаленными операциями — если какое-то действие можно осуществить с локальным файлом, то это же действие можно будет произвести и с файлом на другой машине из сети.

III. RPM5 для разработчиков

Многие новации в RPM5 на упрощение создания пакетов, экономя время разработчиков и позволяя собрать большее количество пакетов при тех же ресурсах. Рассмотрим основные улучшения для разработчиков.

A. *Файловые триггеры*

При установке или удалении многих пакетов возникает необходимость обновлять системное окружение. Например, при добавлении в систему новых библиотек (равно как и при удалении библиотек) нужно обновить кэш динамического загрузчика; при добавлении или удалении иконок для приложений, необходимо обновить кэш иконок и так далее.

Традиционно такие действия выполняются в скриптах, входящих в состав пакетов. Поддержкой этих скриптов занимаются разработчики, формирующие пакеты. Однако ряд действий одинаков для многих пакетов, и их дублирование во всех скриптах нецелесообразно.

Для избавления от подобного дублирования, разработчики RPM5 предложили концепцию файловых триггеров. Триггеры позволяют перенести на инструментарий RPM выполнение действий, инициируемых появлением или удалением из системы определенных файлов.

Каждый триггер состоит из двух частей:

- Регулярное *выражение для имен файлов, при появлении/удалении которых триггер должен срабатывать.*
- Перечень действий, которые необходимо выполнять с целевыми файлами.

Опыт использования файловых триггеров в дистрибутивах ROCA и Mandriva показал, что они не только сокращают размер скриптов, входящих в пакеты, но и позволяют избавиться от человеческого фактора, приводящего к появлению ошибок в таких скриптах.

B. *Локализация*

Одной из важнейших для пользователей характеристик пакета в формате RPM является его описание. Обычно в пакете присутствуют краткое однострочное описание (Summary) и более длинное и развернутое (Description). Эти описания демонстрируются пользователю как при работе в командной строке, так и при использовании более высокоуровневого инструментария с графическим

интерфейсом. Для удобства пользователя, описания желательно демонстрировать на его родном языке — то есть необходима возможность *локализации* описаний пакетов.

Как ни удивительно, но за все время развития RPM4 удобного способа предоставлять пользователям описание пакетов на различных языках предложено не было. Сам формат RPM предусматривает подобных описаний непосредственно в пакете. Для этого все описания должны храниться вместе с остальными данными, используемыми при сборке — исходным кодом программы, различными патчами, файле с инструкциями для инструмента сборки rpmbuild и прочим. Все существующие переводы помещаются в метаданные пакета при сборке. Инструментарий RPM не предоставляет возможности изменять описание уже собранного пакета, поэтому при обновлении перевода пакет необходимо пересобрать. Кроме того, чтобы измененное описание было доступно пользователю, ему необходимо поставить новую версию всего пакета.

Вследствие подобных недостатков, помещение переведенных описаний непосредственно в RPM-пакет во многих дистрибутивах считается плохой практикой (см., например, политики сборки дистрибутивов Mandriva [2] или POCA [3]).

Альтернативным подходом является помещение переводов описаний пакетов в отдельные файлы. Как правило, используются файлы в формате, используемом утилитами gettext (стандартным механизмом для локализации приложений в Linux). С этими файлами переводчики работают, как при переводе любой программы, использующей gettext. Для каждого языка создается свой файл, в который помещаются переводы описаний всех пакетов на этом языке. Файлы с переводами собираются в отдельный пакет (традиционно называемый *specspo*), который помещается в систему при ее установке. Инструментарий RPM при работе в системе использует файл с описаниями, соответствующей текущим языковым настройкам.

Как показала практика, такой подход достаточно удобен для переводчиков (которые работают с единственным файлом), но имеет ряд недостатков с точки зрения разработчиков дистрибутива.

Ввиду особенностей specspo в совокупности с gettext, в результирующих файлах нет указаний, какое описание к какому пакету относится. Как следствие, при поиске перевода инструментарий вынужден просматривать весь файл в поисках нужных строк, что не лучшим образом сказывается на скорости работы. А при внесении изменений в описание любого пакета, файл с описаниями и переводами необходимо перегенировать полностью — возможности указать, какие строки изменились, не предусмотрено.

При использовании specspo снимается потребность обновить пакет для получения его обновленного описания. Тем не менее, остается необходимость обновлять сам пакет specspo.

Дискуссии по поводу более удобного способа локализации описаний пакетов длятся уже много лет (см, например, обсуждение в списке рассылки Fedora [4]), однако в рамках RPM4 никаких сдвигов не наблюдается.

Разработчиками RPM5 несколько месяцев назад был предложен альтернативный подход, заключающийся в помещении локализованных описаний пакетов в отдельное хранилище (базу данных), которое по ключу <*имя пакета; язык*> будет выдавать необходимый перевод. В качестве прототипа реализовано хранение описаний в базе данных SQLite3. В настоящее время производится тестирование этой реализации. В случае успеха (и, возможно, после некоторых доработок) новый метод планируется внедрить в дистрибутивы POCA и Mandriva.

C. Автоматизация подготовки пакета к сборке

Основной сущностью, с которой имеет дело разработчик, собирающий пакет RPM — это *spec-файл*, содержащий инструкции по сборке пакета. Spec-файлы создаются преимущественно вручную и для программ со сложной схемой сборки могут иметь достаточно большой размер. Кроме того, при составлении таких файлов необходимо учитывать специфику конкретных дистрибутивов — местоположение тех или иных файлов, версии сборочных инструментов и компонентов окружения и так далее.

Одним из основных преимуществ систем управления пакетами в целом и RPM в частности является использование механизма *зависимостей*. В общем случае зависимость — это сущность, предоставляемая одним или несколькими пакетами, которую другие пакеты могут потребовать для своей работы. Каждая зависимость представляется в пакете в виде обычной текстовой строки. При установке пакета в систему, инструментарий RPM проверяет, что в ней есть все необходимые пакету зависимости. При удалении пакета проверяется, что этот пакет не требуется для работы других компонентов системы. Такой подход позволяет поддерживать целостность и самодостаточность набора ПО, установленного в системе.

Изначально перечисление зависимостей каждого пакета практически целиком отдавалась на откуп разработчикам, формирующим этот пакет. Естественно, у разных разработчиков имелись разные представления о том, как именовать зависимости, фактически означающие одно и то же, что в итоге привело к существенным расхождениям в именовании зависимостей среди различных дистрибутивов. Подобное расхождение является существенным препятствием для создания пакетов, которые могут быть установлены во всех системах. Даже если содержимое пакетов одинаково для разных дистрибутивов, имена зависимостей могут оказаться различными, и для каждой системы потребуется создавать свой собственный пакет.

Частично решить эту проблему позволяет использование автоматических генераторов зависимостей, используемых и в RPM4, но наиболее активно развивающихся в RPM5. Помимо унификации имен зависимостей и экономии времени разработчиков, использование генераторов снимает еще ряд проблем, например:

- При необходимости переименовать зависимость, это достаточно сделать в одном месте (инструментарии сборки) и пересобрать все пакеты, на которых это переименование может сказаться.
- Исключается возможность опечаток и прочих ошибок, возникающих из-за невнимательности разработчика.

В настоящее время в RPM5 реализованы генераторы зависимостей приложений от имен времени исполнения разделяемых библиотек, а также зависимостей от модулей интерпретаторов Perl, Python и Ruby. Отметим, что многие разработчики дистрибутивов добавляют собственные генераторы зависимостей.

Некоторые генераторы используют в своей работе эвристики, что не позволяет гарантировать полное отсутствие ошибок. Однако практика показывает, что доля ошибок на реальных пакетах дистрибутивов существенно меньше одного процента, и ручная обработка случаев некорректной работы генераторов существенно менее ресурсоемка, чем ручное формирование всех зависимостей.

D. Встроенные интерпретаторы

Одной из востребованных возможностей RPM является выполнение определенных действий в процессе установки, удаления или обновления пакета. Например, если для корректной работы приложения необходимо наличие в системе пользователя или группы с определенным именем, то такой пользователь или группа могут быть созданы скриптом, автоматически вызываемым на заключительной стадии установки пакета.

Подобные скрипты помещаются в метаданные пакета и традиционно создаются на языке интерпретатора Shell. Выбор этого языка был обусловлен тем, что при установке пакета интерпретатор Shell в системе гарантированно присутствует, чего нельзя сказать об интерпретаторах других языков. Использование же скомпилированных программ для подобных действий редко оправдано, поскольку сильно усложняет процесс сборки и поддержки пакета.

В то же время язык Shell достаточно специфический, и многие сборщики пакетов с ним знакомы недостаточно хорошо. Для многих из них удобнее было бы использовать более современный и выразительный язык — например, Ruby, Python или Perl. Кроме того, иногда возникают проблемы, которые сложно решить штатными средствами Shell, но которые можно обойти с использованием других языков. Например, если при обновлении пакета создается символьная ссылка и выясняется, что уже существует директория с таким названием, то RPM обновлять пакет откажется (что обусловлено особенностями реализации функции *cp,* используемой для копирования новых файлов на место предыдущих). В то же время такая проблема может быть эффективно решена посредством небольших скриптов на Perl или Lua, выполняющихся перед копированием файлов.

Создатели RPM5 предоставляют возможность использования для выполнения скриптов интерпретаторов, отличных от Shell, посредством встраивания этих интерпретаторов непосредственно в RPM. Поскольку многие интерпретаторы предоставляют разделяемые библиотеки, реализующие всю нужную функциональность, то такая интеграция обычно не составляет труда. В настоящее время RPM5 может быть собран со встроенной поддержкой Ruby, Python, Perl, Tcl и Lua.

Также стоит отметить поддержку в RPM5 базовых возможностей по работе с реляционными базами данных. Инструментарий RPM5 может быть собран с поддержкой ODBC и базовой поддержкой языка запросов SQL, что дает возможность обращаться к различным базам данных при установке/удалении пакетов, не требуя наличия каких-то дополнительных утилит и библиотек.

IV. Заключение

В данной статье приведены только основные нововведения RPM5. Помимо рассмотренных изменений, в ходе работ над RPM5 был произведен рефакторинг кода инструментария, переработаны привязки к Python и Ruby, внесено множество других относительно небольших модификаций, в совокупности позволивших улучшить качество кода и удобство использования API RPM.

При этом RPM5 остается обратно совместимым с RPM4 и способен устанавливать пакеты, собранные посредством инструментария четвертой версии. Разработчики RPM5 отслеживают все изменения, происходящие в RPM4, и при необходимости вносят соответствующие модификации в свой продукт.

Опыт использования RPM5 в дистрибутивах РОСА и Mandriva показывает, что проект пригоден к использованию в промышленных масштабах, в качестве основного инструментария управления пакетами. При этом инструментарий продолжает активно развиваться, и в будущем следует ожидать еще большего количества улучшений. За развитием RPM5 можно следить на официальном сайте, а также на портале Launchpad [5]. Процесс разработки полностью открыт, и принять в нем участие могут все желающие.

Примечания

[1] Официальный сайт RPM5. http://rpm5.org.

[2] Mandriva Packaging Policies – Charsets. http://wiki.mandriva.com/en/Policies/Charset.

[3] [link removed for blind review]

[4] Критика specspo в списке рассылки Fedora. http://lists.fedoraproject.org/pipermail/devel/2005-November/076390.html

[5] Раздел RPM5 на портале Launchpad. http://launchpad.net/rpm