Сборочная система git.alt

Алексей Турбин <at@altlinux.org>

1 июля 2008 г.

Аннотация

Новая система сборки rpm пакетов поддерживает целостность репозитория за счёт транзакционных переходов, при которых полностью вычисляются характеристики репозитория. По умолчанию разрешены только переходы, которые не ухудшают характеристик. Система ориентирована на асинхронное продвижение транзакций, а окончательная сериализация и учет взаимного влияния между транзакциями осуществляется при помощи *слияний* (merge). Для этого система поддерживает внутреннюю структуру данных — интроспективный *метарепозиторий*.

Во введении дан обзор средств совместной разработки ALT Linux Team, созданных на основе распределенной системы контроля версий GIT.

Введение

Сборку rpm пакетов можно рассматривать как процесс, который реализует функцию B(S,C)->P, где S — src.rpm пакет с исходным кодом, C (chroot) — сборочная среда, P — собранные rpm пакеты. Надёжная сборка rpm пакетов, т. е. воспроизводимость процесса сборки B (build), осуществляется с помощью программы hasher [Левин 2004].

Позже была разработана программа gear [Левин 2007], которая позволяет хранить исходный код src.rpm пакетов в системе контроля версий GIT [Торвальдс 2005]. *Gear-репозиторием* G мы называем git-репозиторий с исходным кодом, из которого можно получить src.rpm пакет для сборки: G->S.

Несмотря на некоторые преимущества src.rpm пакетов (такие, как сохранение полностью оригинального тарболла, а также сама возможность распространения исходного кода в виде, готовом для сборки), использование gear-репозиториев значительно упрощает *совместную разработку* пакетов. Вообще, использование системы контроля версий стимулирует переход от пассивной *поддержки* пакетов к более интенсивной работе с исходным кодом, дает больше возможностей для *разработки*. Это преимущество можно считать решающим, поэтому было предложено использовать gear-репозитории как основной формат хранения исходного кода пакетов в ALT Linux Team. (При этом src.rpm пакеты продолжают существовать лишь как промежуточный «полуфабрикат» для сборки; в дальнейшем возможен полный отказ от хранения src.rpm пакетов.)

Сервер совместной разработки gear-репозиториев ALT Linux Team мы кратко обозначаем как git.alt. Одной из подсистем git.alt является girar (архив gear-репозиториев); girar осуществляет доступ разработчиков к gear-репозиториям.

Каждый разработчик имеет собственные копии gear-репозиториев, в которые он может вносить, вообще говоря, достаточно произвольные изменения (предлагая, таким образом, включить эти изменения в очередную версию пакета). Реализована система почтовых уведомлений, которая упрощает обмен изменениями. При этом фактический обмен и аккумуляция изменений осуществляется средствами распределенной системы контроля версий `GIT`.

Несмотря на то, что любой разработчик может внести изменения в пакет, окончательную версию пакета могут подготовить только один или несколько разработчиков, за которыми закреплён этот пакет (maintainers). Контроль осуществляется подсистемой `girar ACL`, и «неавторизированные» версии пакетов по умолчанию отвергаются (вместе с тем, сохранена возможность для т. н. NMU, non-maintainer upload).

Когда новая версия пакета окончательно подготовлена, разработчик делает в gear-репозитории *метку* (tag) с криптографической подписью и инициирует запрос на сборку пакета. Запрос обрабатывается сборочной системой `girar-builder`, которая является предметом настоящего доклада. Сборочная система взаимодействует с *кеширующим сервером* gear-репозиториев. Если запрос на сборку был обработан успешно, то собранные rpm пакеты помещаются в репозиторий rpm пакетов, а gear-репозиторий с новой меткой публикуется на кеширующем сервере. Названия gear-репозиториев на кеширующем сервере совпадают с именами src.rpm пакетов (это требование не является обязательным для gear-репозиториев разработчиков). Таким образом, кеширующий сервер предоставляет доступ к «официальным» gear-репозиториям, содержимое которых соответствует фактическим сборкам пакетов.

1 Задания и транзакции

Задание состоит из одного или нескольких gear-репозиториев (и их меток), отправленных на сборку. Сборочная система сначала осуществляет *первичную сборку* пакетов. Если при первичной сборке пакетов возникли грубые ошибки, то задание автоматически отменяется. В противном случае, если все пакеты в задании успешно собрались, то *открывается транзакция*: генерируется временный репозиторий, в котором выполняется ряд *проверок* (вычисляются *характеристики* нового репозитория). По умолчанию переход в новое состояние разрешен, только если характеристики репозитория не ухудшились. Если же собранные пакеты ухудшают характеристики репозитория, то составляется *список нарушений*, и задание переводится в *отложенный режим*.

Выделим следующие проверки, выполняемые сборочной системой.

- **Неудовлетворенные зависимости** (unmet dependencies). Если при помещении пакетов в репозиторий возникают новые неудовлетворенные зависимости, то по умолчанию такой переход не может быть разрешен.

- **Тестовая пересборка пакетов** в наибольшей степени обнаруживает взаимное влияние между пакетами. Эта проверка может быть довольно ресурсоемкой: должны быть пересобраны все пакеты, у которых в сборочной среде `C` оказывается хотя бы один новый пакет из транзакции. Если же хотя бы один

новый пакет их транзакции входит в базовую сборочную среду, то потребуется полная тестовая пересборка всего репозитория rpm пакетов. Сборочная система фиксирует не только саму возможность пересборки, но и изменение зависимостей у пересобранных пакетов.

- **Идентичность noarch пакетов** при сборке на всех архитектурах.
- **Нарушение ACL** у какого-либо пакета в задании не относится, строго говоря, к характеристикам репозитория; однако включение его в общий список нарушений транзакции упрощает NMU.

Воздействовать на задания в отложенном режиме можно, в основном, двумя способами.

- **Добавление новых пакетов**, которые исправляют нарушения. Например, если у библиотеки изменилось имя (soname), то в репозитории могут образоваться неудовлетворенные зависимости на старое имя библиотеки. Тогда в задание можно добавить все пакеты, которые зависят от старой библиотеки, чтобы пересобрать их с новой библиотекой.
- **Разрешение нарушений.** Например, администратор `git.alt` может разрешить нарушение ACL, если «неавторизированная» версия пакета (NMU) содержит исправления критических ошибок.

Таким образом, задания могут переводиться в отложенный режим и позднее возобновляться. Если все нарушения были сняты, то выполняется транзакционный переход репозитория в новое состояние.

2 Метарепозиторий

Необходимость учета взаимных влияний между транзакциями приводит к идее *метарепозитория* — структуры данных, используемой сборочной системой, которая также представляет самостоятельный интерес для разработчиков. Метарепозиторий организован как обычный git-репозиторий, который содержит каталоги по имени src.rpm пакетов `S`. В этих каталогах хранится вся существенная информация о пакетах, в частности:

- **Сборочные зависимости** (BuildRequires) src.rpm пакетов. Хранение сборочных зависимостей упрощает запуск тестовой пересборки пакетов.
- **Зависимости** собранных пакетов. Изменение зависимостей при тестовой пересборке позволяет изучать взаимное влияние между пакетами.
- **Логи сборки** пакетов. Изменения в логах сборки пакетов позволяет анализировать свойства пакетов, которые трудно учесть формально (например, новые предупреждения при компиляции).

История изменений в метарепозитории соответствует истории продвижения транзакций. В каждый момент времени основное состояние (master) метарепозитория соответствует текущему (стабильному) состоянию репозитория rpm пакетов. Когда открывается новая транзакция, в метарепозитории создается новый *бранч* (в терминах `GIT`), имя которого соответствует целочисленному идентификатору задания; дальнейшие изменения в бранче метарепозитория соответствуют этапам продвижения транзакции.

Когда отложенное задание вновь активизируется, может потребоваться *слияние* (merge) master в бранч, которое учитывает, в частности, возможные изменения в сборочной среде `C`. Наконец, при транзакционном переходе осуществляется другой тип слияния: окончательное слияние бранча в master (в простейшем случае оно может быть реализовано при помощи т. н. fast forward). Оба типа слияний являются не чисто «текстовыми» (как в системе контроля версий), а «логическими», т. е. результат слияния определяется дополнительной логикой работы сборочной системы.

Список литературы

[Левин 2004] *Дмитрий Левин*, Hasher: технология безопасной сборки пакетов // Первая международная конференция разработчиков свободных программ на Протве. Тезисы докладов. М., 2004. С. 28–30.

[Левин 2007] *Дмитрий Левин*, От SRPMS к GEAR // Четвёртая международная конференция разработчиков свободных программ на Протве. Тезисы докладов. М., 2007. С. 14–18.

[Торвальдс 2005] *Линус Торвальдс*, GIT — the information manager from hell. `http://git.or.cz` и др.