Денис Силаков

OpenSUSE Build Service

Одной из примечательных особенностей мира Linux является наличие сотен различных дистрибутивов. Для разработчиков приложений, желающих донести свои творения до как можно большего числа пользователей, такое разнообразие может стать проблемой – удостовериться, что приложение способно функционировать в большом количестве дистрибутивов – довольно непростая задача.

Типичным решением этой проблемы является делегирование всех забот разработчикам дистрибутивов – если им понравится ваше приложение, то они включат его в свои репозитории и будут сами заботиться о корректности его функционирования. Хорошо бы при этом попасть в основной репозиторий, а не в свалку всего и вся, которая в установленной системе по умолчанию может быть отключена. Однако популярности приложения надо сначала как-то добиться – а для этого неплохо бы завоевать аудиторию. К тому же не факт, что разработчики дистрибутива не добавят в приложения чего-нибудь “от себя”, что придется не совсем по нраву авторам оригинальной программы.

Конечно, можно просто раздавать всем исходный код и предлагать собирать программу самостоятельно, но далеко не каждый готов тратить время на сборку приложения лишь для того, чтобы попробовать его “вживую”. Кроме того, такой подход не является дружественным по отношению к “обычным” пользователям десктопов.

В общем, и для пользователей, и для разработчиков ПО выгодна ситуация, когда разработчик сам создает и выкладывает готовые пакеты (RPM, Deb и прочие) для различных систем.

В идеале, разработчику для этого надо иметь под рукой установленные системы, для которых планируется собирать пакеты. Поэтому организация сборки для достаточно большого числа систем является достаточно трудоемкой и ресурсоемкой задачей. К счастью, существует готовый инструментарий, способный взять на себя существенную часть забот – речь идет об OpenSUSE Build Service (OBS).

Изначально OBS предназначался для сборки дистрибутивов семейства SUSE (и используется для этой цели и поныне), однако со временем его функциональность существенно расширилась. Сейчас OBS – это система, позволяющая собирать RPM и Deb пакеты произвольных программ для наиболее популярных дистрибутивов Linux. В настоящее время поддерживается сборка пакетов для 8 дистрибутивов (причем для различных версий каждого из них) на 7 аппаратных платформах.

Давайте посмотрим – что же нам предлагает OBS и как с ним работать.

Начало работы

Первым делом, идем на http://build.opensuse.org, регистрируемся и заходим в систему.
По умолчанию, для нового пользователя создается пустой домашний проект с именем 'home:<username>'. Перейдя в проект, увидим две основные секции, из которых состоит каждый проект в OBS – пакеты (Packages) и репозитории (Repositories). В первом приближении, пакет – это то, что мы будем собирать, а репозитории соответствуют дистрибутивам, для которых будет происходить сборка. Как это окно выглядит уже в развернутом проекте – можно посмотреть на рис. 1. Справа для каждого репозитория отображается статус сборки, суммарный по всем

пакетам. Такой же статус можно посмотреть и для каждого отдельного пакета, перейдя к нему посредством пункта меню “Packages”.

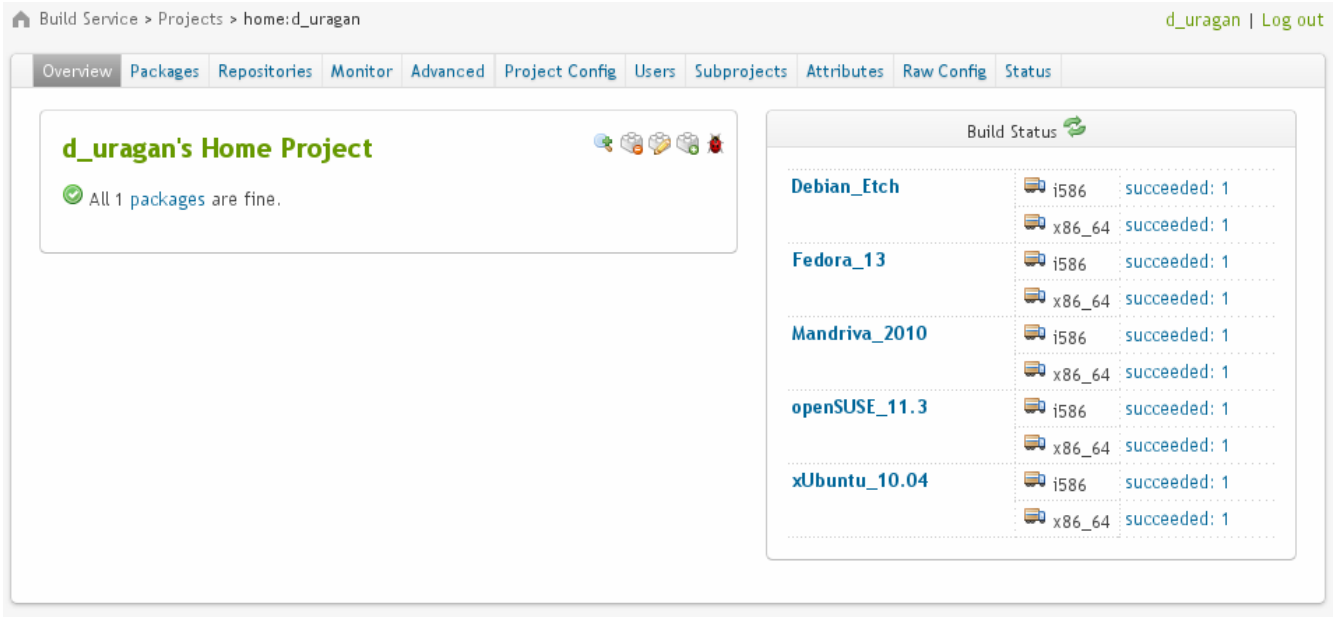

Рисунок 1. Страничка домашнего проекта

В рамках одного проекта можно собирать несколько различных пакетов; все они будут собираться для одного и того же множества целевых систем. Если хочется собирать разные пакеты для разных наборов дистрибутивов, то можно завести подпроекты – для этого есть соответствующая кнопка на вкладке Overview проекта. Имя подпроекта отделяется от имени родительского проекта двоеточием ('home:<username>:<subproject>'). Подпроект, в свою очередь, может иметь собственные подпроекты (соответственно, с именами вида 'home:<username>:<subproject>:<subsubproject>').

Для каждого пакета в OBS надо загрузить файлы, необходимые для его сборки (пункт меню “Source Files”) – как правило, это архив с исходным кодом и файлы с инструкциями для соответствующих сборочных инструментов (rpmbuild или скрипты dpkg-*). Как только для пакета загружены файлы с инструкциями, OBS попробует начать сборку. При этом система “знает”, какие дистрибутивы используют dpkg, а какие – rpmbuild; если в проекте есть только spec-файл с инструкциями для rpmbuild, то и сборка будет запущена только для RPM-based дистрибутивов.

Процесс сборки

Процесс сборки пакетов устроен достаточно просто – для каждой целевой системы, OBS развертывает виртуальную машину (ВМ), куда устанавливает дистрибутив с минимальным сборочнымм окружением. Также устанавливаются пакеты, разрешающие зависимости времени сборки (указанные в секциях BuildRequires spec-файла при сборке RPM и Build-Depends/Build-Depends-Indep при сборке Deb).

Файлы проекта также копируются внурть ВМ в директории, где их ожидает увидеть инструментарий сборки. После этого запускается собственно сборка. В случае успеха, собранные пакеты извлекаются из ВМ и становятся доступными пользователю – к ним можно перейти, кликнув по нужному имени дистрибутива. Также пользователю предоставляется полный журнал событий, происходивших внутри ВМ – для доступа к нему надо кликнуть на статус сборки пакета в дистрибутиве. Обратите внимание, что размер Web-формы для

отображения журнала ограничен, и зачастую весь журнал в нее не помещается; в этом случае помогает пункт “Download raw logfile”, выдающий весь журнал в текстовом виде.

“Фишкой” OBS является автоматический перезапуск сборки в случае, если изменился один из файлов проекта либо один из пакетов, от которых проект зависит. Особого интеллекта OBS при этом не демонстрирует – например, даже если был изменен только spec-файл, используемый при сборке RPM-пакетов, то пересборка будет запущена как для RPM, так и для Deb репозиториев. Отключить автоматическую пересборку (либо для конкретных репозиториев, либо сразу для всех) можно посредством соответствующих флагов в меню Repositories. Там же можно указать, следует ли делать ваши пакеты доступными для всех желающих, и нужно ли собирать пакеты с отладочной информацией.

Важным аспектом работы системы является то, что один и тот же spec-файл используется во всех RPM-based дистрибутивах, а один и тот же набор файлов для скриптов dpkg – во всех Debian-based системах, на всех аппаратных платформах. Соответственно, составлять эти файлы надо так, чтобы они успешно работали во всех целевых системах, которые имеют тенденцию отличаться друг от друга, и иногда довольно сильно. Для многих программ это достаточно просто, но в общем случае составление переносимых инструкций для сборки – нетривиальная задача. Увы, здесь OBS никакой помощи не оказывает – все ошибки приходится выявлять и править в ходе экспериментов.

Зависимости от других проектов

Как было сказано выше, OBS автоматически устанавливает в BM пакеты, разрешающие зависимости времени сборки. Зависимости времени исполнения (секции Requires в RPM и Depends/Suggests/Recommends в Deb) никак не учитываются.

По умолчанию, зависимости разрешаются за счет пакетов в репозитории целевого дистрибутива. Однако можно добавлять зависимости и от любых других пакетов из OBS – в этом случае в проект надо добавить ветку (branch) необходимого пакета из соответствующего проекта. Делается это посредством пункта “Branch Package” в меню Packages. В появившемся меню надо задать имя проекта, откуда следует импортировать пакет, имя пакета в том проекте, и имя, под которым пакет будет виден в вашем проекте. Также можно указать, следует ли изменения в пакете, осуществляемые его разработчиком, автоматически переносить в вашу ветку (по умолчанию, так и происходит).

Пакеты, добавленные таким образом, копируются в проект и пересобираются внутри него (то есть даже если в проекте, от которого “ответвился” пакет, есть собранные RPM/Deb для нужных дистрибутивов, то они не используются). Такой подход позволяет вам не просто использовать сторонний пакет, но и добавлять к нему свои патчи. Естественно, все сторонние пакеты должны “уметь собираться” в целевых дистрибутивах проекта.

Обратите внимание, что еще не так давно вместо “Branch package” использовался термин “Link package” (добавить ссылку на пакет) – следы этого еще можно увидеть в различных местах OBS.

Особенности сборки RPM и Deb

Полезной (а для кого-то – раздражительной) особенностью сборки RPM в OBS является автоматический запуск на собранных пакетах утилиты rpmlint, проверяющей соответствие пакетов общепринятым стандартам. В случае, если пакет не проходит тесты rpmlint, сборка считается завершившейся неудачно, и пакеты из BM не извлекаются. Вообще, следование стандартам – это хорошо, но если бороться с rpmlint нет сил, то можно добавить в секцию

BuildRequires spec-файла опцию '-post-build-checks'.

Аналога rpmlint для Deb-пакетов в OBS нет. Однако у сборки Deb-пакетов в системе – свои особенности.

Отличием процесса сборки Deb от RPM является то, что имя архива с исходным кодом не прописывается в инструкциях для скриптов dpkg; вместо этого, инструмент просто ищет файл с именем, имеющим вид <имя_проекта>-<версия>-src.tar.gz (tgz, tar.bz2, …). Впрочем, если ваш архив называется по-другому, беспокоится не стоит – OBS самостоятельно переименует архив при помещении его в виртуальную машину. Правда, такой интеллект предполагает, что среди файлов проекта есть только один архив. В потивном случае, система не сможет понять, который из архивов надо переименовать.

Наконец, при сборке пакета в некоторых Debian-based системах можно получить ошибку

```
dpkg-source: error: unrecognized file for a v1.0 source package
```

Эта ошибка означает, что инструментарию сборки не понравился формат вашего архива, а точнее – метод сжатия. Самый надежный способ избежать таких ошибок – это использование архивов tar, сжатых gzip (tar.gz).

Что еще?

Таким образом, принцип работы OBS досточно прост. Web-интерфейс системы также прост и интуитивно понятен, хотя и имеет тенденцию время от времени изменяться (а документация, как водится, за изменениями не всегда поспевает).

Но Web-интерфейс не всегда удобен – все-таки, он предполагает участие человека, а в производственном процессе удобнее было бы автоматизировать “общение” с системой, написав скрипты для загрузки в нее исходного кода и извлечения результатов сборки. Разработчики OBS такую возможность предусмотрели – пользователям предоставляется REST API (так что можно общаться с OBS посредством, например, curl) и утилиты командной строки.

Наконец, OBS является свободным ПО, так что каждый желающий может развернуть свою собственную копию системы, настроив ее по своему вкусу и потребностям.

Но обо всем этом – как-нибудь в другой раз.