Как Выжить С Множеством Патчей

или

Введение в Quilt*

Andreas Grünbacher, SuSE Labs
agruen@suse.de

26 мая 2013 г.

Аннотация

Просмотрев различные стратегии работы с пакетами, состоящими из базового исходного кода, к которому применяется целый ряд исправлений, представляю в этом документе коллекцию скриптов - *quilt*, которая была специально написана для оказания помощи в работе с множеством патчей и решения общих задач по управлению ими.

1 Введение

В старые времена программное обеспечение определенного поставщика (издателя. Здесь уместнее говорить о дистрибутиве и дистро-специфиченых патчах) в мире открытого исходного кода состояло из файла с официальной версией программного обеспечения, а также файлов с дополнительными патчами, необходимыми для адаптации пакета к конкретным потребностями. Официальный пакет программного обеспечения, как правило, содержался в файле package.tar.gz, а патч был в package.diff. Вместо того, чтобы изменять оригинальный пакет исходников, локальные изменения хранились отдельно. При формировании пакета программного обеспечения tar архив распаковывался, и к нему применялся патч.

Со временем патч-файл стал содежать несколько независимых изменений. Некоторые из этих изменений были интегрированы в более поздних версиях программного обеспечения, в то время как другие дополнения или переделки оставались внешними. Всякий раз, когда выпускалась новая официальная версия патч пересматривался: необходимо было отделить изменения, которые уже были включены в официальную версию от остальных.

*Quilt это проект под лицензией GPL размещенный на хостинге GNU Savannah. Некоторые идеи для этого документа были взяты из файла *docco.txt* в пакете [1] скриптов менеджера патчей Эндрю Мортона. Текст примеров был взят из произведения Вильяма Шекспира *Сон В Летнюю Ночь*.

Значительное улучшение наступило, когда в пакетах поставщиков были разрешены множественные патчи - так патчи обрабатываются сегодня: ряд патчей наносится поверх друг друга. Каждое исправление обычно состоит из логически связанного набора изменений. Когда некоторые патчи переносятся в апстрим, они могут быть просто удалены из пакета конкретного поставщика. Остальные патчи часто продолжают применяться так-же. Некоторые из оставшихся патчей, возможно, придется поддерживать в целом ряде последующих версий, поскольку они слишком специфичны для включения в базовую версию пакета программного обеспечения и т.д. Эти патчи часто рассинхронизируются и нуждаются в обновлении.

Для большинства пакетов, число исправлений, остается относительно низким, так что поддержание этих исправлений возможно без специальных инструментов. Однако некоторые пакеты имеют десятки патчей. Яркий пример - пакет исходных кодов ядра (kernel-source-*2.4.x*) с более чем 1 000 патчей. Сложность управления таким огромным количеством патчей, без инструментов можно легко себе представить.

В этом документе рассматриваются различные стратегии работы с большими наборами исправлений. Патчи обычно генерируются утилитой *diff*, и применяется с помощью *patch*. Различные форматы файлов патча определяется как часть спецификации утилиты *diff* в POSIX.1 [3], однако наиболее часто используемый формат сегодня *unified diff*, не охвачен POSIX.1. Хорошее описание форматов патч-файлов находится на info странице [4] утилиты *GNU diff*.

Вопрос, на который мы попытаемся ответить в данном документе - как патчи лучше хранить в условиях изменений как в оригинальной версии пакета программного обеспечения, а также предшествующих им патчей. Посмотрев на некоторые существующие подходы, рассматривая коллекцию сценариев управление патчами известную как *quilt* ([2]), начинем с основных понятий, и будем двигаться к более сложным задачам.

2 Существующие подходы

Самое простое решение для использования патча - применять все предыдущие патчи. Создается копия исходного дерева.[1] Следующий патч из последовательности (который является одним из тех, которые будут использованы) применяется только к одному из этих исходных деревьев. Это дерево исходных текстов затем изменяется, чтобы отразить желаемый результат. Новая версия патча получается сравнением двух исходных деревьев с помощью *diff*, и записи результата в файл.

Этот простой подход весьма подвержен ошибкам, и оставляет желать

[1] Два экземпляра могут быть жестко связаны друг с другом, что значительно ускоряет как копирование и окончательный "Diffing". Если используются жесткие ссылки, необходимо позаботиться о том, чтобы инструменты, используемые для обновления одной копии исходного дерева создавали новые файлы, а не перезаписывали общие файлы. Редакторы, такие как *emacs* и *vi*, а так-же утилиты типа *patch*, поддерживают это.

лучшего. Несколько человек независимо друг от друга написали скрипты для автоматизации и совершенствования этого процесса.

Системы контроля версий, такие как *CVS* или *RCS* могут быть разумной альтернативой в некоторых случаях. В систему контроля версий вносится состояние рабочего дерева с рядом примененных патчей. Затем применяется следующий патч. После обновления рабочего дерева до необходимого состояния, с помощью *cvs diff* или другой программы создается diff между копией кодов и рабочим деревом. В этом случае система контроля версий используется только для хранения и сравнения со старой версией, но переключение между различными патчами не упрощается. Стреляем из пушки по воробьям.

Один из наиболее передовых подходов - сценарии управления патчами Эндрю Мортона [1]. Автор этого документа не нашел ни одного из возможных решений которое будет масштабироваться до конкретных требований пакета исходного кода ядро SUSE, и стал улучшать сценарии Эндрю Мортона, пока они не стал тем, чем они являются сейчас [2].

3 Quilt: основные понятия и операции

Далее в документе обсуждается коллекция скриптов *quilt*.

С *quilt* вся работа происходит в одном дереве каталогов. Начиная с версии 0.30, команды могут быть вызваны из любой точки исходного дерева. Команды имеют вид "quilt cmd" и похожие на CVS командам. Они могут быть сокращены, если определенная часть команды является уникальным. Все команды выводят текст с помощью по их использованию с ключом "quilt cmd -h'.

Quilt управляет стеком патчей. Патчи применяются инкрементно на базовое дерево плюс все предыдущие патчи. Они могут быть добавлены на вершину стека (quilt push) и убраны из стека (quilt pop). Доступны команды для запросов содержания серии файл (quilt series, см. ниже), содержимое стека (quilt applied, quilt previous, quilt top), и патчи, которые не применяются в определенный момент (quilt next, quilt unapplied). По умолчанию, большинство команд применяется к верхнему патчу на стеке.

При изменении файлов в рабочей папке эти изменения становятся частью рабочего состояния верхнего патча, при условии, что эти файлы являются частью патча. Файлы, которые не являются частью исправление должны быть добавлено до изменения чтобы quilt были известны оригинальные версии файлов. Команда quilt refresh восстанавливает патч. После обновления, патч и рабочее состояние те же.

Патч-файлы находятся в подкаталоге patches дерева исходных текстов (см. Рисунок 1). Можно использовать переменную окружения QUILT_PATCHES для переопределения этого места. Каталог patches может содержать подкаталоги или быть символической ссылкой.

Файл series содержит список имен патч файлов, определяет порядок, в котором они применяются. Если есть средства, которые могут автоматиче-

```
work/ -+- ...
|- patches/ -+- series
| |- patch2.diff
| |- patch1.diff
| +- ...
+- .pc/ -+- applied-patches
|- patch1.diff/ -+- ...
|- patch2.diff/ -+- ...
+- ...
```

Рис. 1: Файлы Quilt в исходном дереве.

ски сгенерировать файл series (см. раздел 5.8), то они обычно предоставляются вместе с набором исправлений. В series каждое имя файла патчей находится на отдельной строке. Патч-файлы идентифицируются по путям, относительным к каталогу patches; патчи могут быть в подкаталогах ниже каталога patches. Строки в файле series, начинающиеся с символа решетки (#) игнорируются. Когда quilt добавляет, удаляет или переименовывает патчи, она автоматически обновляет серию файлов. Пользователи quilt могут изменить файл series когда применяются некоторые патчи, пока патчи остаются в исходном порядке.

Разные файлы series могут быть использованы для сборки патчей поразному, что соответствует, например, различным ветвям разработки.

До применения патча (или "помещения в стек") копии всех файлов сохраняются в каталоге .pc/*patch*.[2] Патч будет добавлен в список примененных в данный момент патчей (.pc/applied-patches). Позже, когда восстанавливается патч (quilt refresh), резервные копии в .pc/*patch* сопоставляются с текущими версиями файлов в исходном дереве, используя *GNU diff*.

Документацию, связанную с патчем можно добавить в начале файла. Quilt тщательное сохраняет весь текст, который предшествует фактически патчу при выполнении обновления.

Файл series ищется в корневом каталоге дерева исходных текстов, в каталоге патчей, а также в каталоге .pc. Используется первый найденный файл series, также он может быть символической ссылки, или файлом с несколькими жесткими ссылками. Как правило, только один файл series используется для набора патчей, поэтому подкаталог patches - это удобное расположение.

Хотя патчи применяются для дерева исходных текстов, директория .pc играет важную роль во многих операциях, в том числе принятия патчей из стека (quilt pop), и обновлении патчей (quilt refresh). Файлы в папке .pc автоматически удаляются, когда они больше не нужны, так как правило, нет необходимости чистить их вручную. Переменные окружения QUILT_PC можно использовать для переопределения расположения .pc каталога.

[2]Имя патч файла используется в качестве имени подкаталога ниже директории .pc. *GNU patch*, который используется внутри quilt чтобы применять патчи, создает резервные копии файлов и применяет патч за один шаг.

4 Примеры

В этом разделе демонстрируется, как новые патчи создаются и обновляются, и как решаются конфликты. Давайте начнем с короткого текстового файла:

```
Yet mark'd I where the bolt of Cupid fell:
It fell upon a little western flower,
Before milk-white, now purple with love's wound,
And girls call it love-in-idleness.
```

Новые патчи создаются с помощью quilt new. Новый патч автоматически становится верхним на стеке. Файлы должны быть добавлены с помощью quilt add до их изменения. Заметим, что это несколько отличается от стиля CVS взаимодействия: с CVS файлы в репозитории, и добавить их до коммита (но после внесения изменений в них) вполне достаточно. Файлы, как правило, добавляются и тут же изменяются. Команда quilt edit добавляет файл и загружает его в редактор по умолчанию. Используется (переменная окружения EDITOR которая определяет редактор по умолчанию. Если она не установлена, используется *vi*.)

```
$ quilt new flower.diff
Patch flower.diff is now on top
$ quilt edit Oberon.txt
File Oberon.txt added to patch flower.diff
```

Давайте предположим, что следующие строки были добавлены в Oberon.txt во время редактирования:

```
The juice of it on sleeping eye-lids laid
Will make a man or woman madly dote
Upon the next live creature that it sees.
```

Сам файл патча создется (а позднее обновляется) с помощью quilt refresh. Результат выглядит следующим образом: [3]

```
$ quilt refresh
$ cat patches/flower.diff
Index: example1/Oberon.txt
===================================================================
--- example1.orig/Oberon.txt
+++ example1/Oberon.txt
@@ -2,3 +2,6 @@
It fell upon a little western flower,
Before milk-white, now purple with love's wound,
And girls call it love-in-idleness.
+The juice of it on sleeping eye-lids laid
+Will make a man or woman madly dote
+Upon the next live creature that it sees.
```

[3] Временные метки в патчах исключены из вывода в примерах.

Теперь давайте предположим, что в тексте была пропущена строка, и она должна быть вставлена. Файл Oberon.txt уже является частью патча flower.diff, поэтому он может быть немедленно изменен в редакторе без использования команды quilt add. Кроме того, можно использовать команду quilt edit, она просто открывает редактор по умолчанию, если файл уже является частью патча.

После того как строка добавлена, мы используем quilt diff -z, чтобы увидеть изменение правок, которые мы сделали:

```
$ quilt diff -z
Index: example1/Oberon.txt
===================================================================
--- example1.orig/Oberon.txt
+++ example1/Oberon.txt
@@ -2,6 +2,7 @@
It fell upon a little western flower,
Before milk-white, now purple with love's wound,
And girls call it love-in-idleness.
+Fetch me that flower; the herb I shew'd thee once:
The juice of it on sleeping eye-lids laid
Will make a man or woman madly dote
Upon the next live creature that it sees.
```

Изменения самого верхнего патча могут быть просмотрены командой quilt diff без аргументов. Это не изменит сам файл патча. Эти изменения добавятся к патчу путем его обновления командой quilt refresh. Затем мы удаляем пакет из стека командой quilt pop:

```
$ quilt refresh
Refreshed patch flower.diff
$ quilt pop
Removing flower.diff
Restoring Oberon.txt

No patches applied
```

Далее, давайте предположим, что Oberon.txt был изменен в "апстриме": слово *girl* не очень хорошо подходит, и поэтому оно было заменено на *maiden.* Oberon.txt теперь содержит:

```
Yet mark'd I where the bolt of Cupid fell:
It fell upon a little western flower,
Before milk-white, now purple with love's wound,
And maidens call it love-in-idleness.
```

Это приводит к тому, что flower.diff больше не применяется корректно. При попытке добавить flower.diff в стек с помощью quilt push, мы получим следующий результат:

```
$ quilt push
Applying flower.diff
patching file Oberon.txt
```

```
Hunk #1 FAILED at 2.
1 out of 1 hunk FAILED -- rejects in file Oberon.txt
Patch flower.diff does not apply (enforce with -f)
```

Quilt не может автоматически применить патчи, в которых есть отвергнутые изменения. Патчи, которые не применяются без ошибок можно "применить насильно" командой quilt push -f, в результате чего будут созданы файлы отвергнутых изменений, для каждого файла, который имеет конфликты. Эти конфликты должны разрешаться вручную, после чего данный патч может быть обновлен (quilt refresh). Quilt помнит, когда патч был применен насильно. Он отказывается добавлять дальнейшие патчи на верх стека, и не удаляет их из стека. Примененный насильно патч может быть "насильно" удален из стека командой quilt pop -f, однако вот что происходит, когда flower.diff применяется "насильно" :

```
$ quilt push -f
Applying flower.diff
patching file Oberon.txt
Hunk #1 FAILED at 2.
1 out of 1 hunk FAILED -- saving rejects to file Oberon.txt.rej
Applied flower.diff (forced; needs refresh)
```

После повторного добавления строк из flower.diff в Oberon.txt, мы обновим патч командой quilt refresh.

```
$ quilt edit Oberon.txt
$ quilt refresh
Refreshed patch flower.diff
```

Наша последняя версия Oberon.txt содержит:

```
Yet mark'd I where the bolt of Cupid fell:
It fell upon a little western flower,
Before milk-white, now purple with love's wound,
And maidens call it love-in-idleness.
Fetch me that flower; the herb I shew'd thee once:
The juice of it on sleeping eye-lids laid
Will make a man or woman madly dote
Upon the next live creature that it sees.
```

5 Дальнейшие команды и понятия

В этом разделе представлено несколько основных команд, а затем описаны дополнительные понятия, которые могут быть не очевидны. Мы не описываем все особенности quilt здесь, так много команд quilt совершенно интуитивны, кроме того, справка, которая описывает доступные параметры для каждой команды, доступна через quilt *cmd* -h.

Команда quilt top показывает название верхнего патча. Команда quilt files - какие файлы патч содержит. Команда quilt patches - какие патчи изменяют указанный файл. В нашем предыдущем примере, мы получим следующие результаты:

```
$ quilt top
flower.diff
$ quilt files
Oberon.txt
$ quilt patches Oberon.txt
flower.diff
```

Команды quilt push и quilt pop дополнительно принимают число или имя исправления в качестве аргумента. Если указано число, определенное количество исправлений добавляется (quilt push) или удаляется (quilt pop). Если задано имя патча, патчи добавляются (quilt push) или удаляются (quilt pop), пока указанный патч не будет находится на вершине стека. С опцией -a все патчи в серии файлов добавляются (quilt push), или все примененные патчи удаляются из стека (quilt pop).

5.1 Уровни Вложенности Патчей

Quilt предполагает, что патчи применяются с уровнем вложенности 1 (опция -p1 программы *GNU patch*) по умолчанию: верхняя директория в именах файлов в патчах игнорируется. Quilt помнит уровнь каждого патча в папке series. При создании (quilt diff) или обновлении патча (quilt refresh), может быть определен уровень вложенности и в ряд файлов будут внесены соответствующие изменения. Quilt можно применять патчи с произвольным уровнем вложенности, а также генерировать патчи с уровнем равным нулю или единице. Когда уровень вложенности равен единице название каталога, который содержит рабочие дерево используется в качестве дополнительного компонента пути. (Так, в нашем примере Oberon.txt содержится в каталоге example1.)

5.2 Импорт Патчей

Команда quilt import автоматизирует импорт исправлений в quilt. Команда копирует патч в каталог patches и добавляет его в series. Для патча уровень вложенности которого отличается от единицы он добавляется после имени файла патча. (Запись для файла a.diff с нулевым уровнем будет выглядеть так "`a.diff -p0`".)

Другая распространенная операция это добавить новые патчи на самый верх стека. Это можно сделать вручную сначала добавить все файлы, содержащиеся в дополнительных патчах для исправления коммандой quilt add, [4], а затем применять патч на рабочее дерево. Команда quilt fold объединяет эти шаги.

[4] Утилита *lsdiff*, которая является частью пакета *patchutils* генерирует список измененных в патче файлов.

5.3 Делимся патчами с другими

Для обмена набором патчей с кем-то еще все, что необходимо, это файл series, который содержит список патчей и каким образом они применяются, и сами патчи. Каталог *.pc* содержит только рабочее состояние quilt, и не должен распространяться. Убедитесь, что все патчи актуальны, и обновите их по мере необходимости. Опция –combine команды quilt diff может быть использована для получения одного большого патча из всех исправлений в серии файлов.

5.4 Объединение с "апстримом"

Концепция объединения ваших патчей с апстримом идентична применению ваших патчей на более новой версии программы.

До слияния, убедитесь, что все ваши патчи удалены с помощью quilt pop -a. Затем обновите оригинальных исходный код. И наконец, удалить устаревшие патчи из файла series и запустите quilt push для применения остальных, урегулируйте конфликты и обновите патчи по мере необходимости.

5.5 Ветвление

Есть ситуации, в которой обновления патча на месте не является идеальным: один и тот же патч может быть использован нескольких сериях, Он может также служить удобным местом хранения документации старых версий патча, а также создавать новые с различными именами. Это можно сделать вручную, создав копию патча (который не должен быть применен), и обновить его патча в файле series.

Команда quilt fork упрощает это: она создает копию верхнего патча в серии, и обновляет файл series. Если патч указывается явно, quilt fork сгенерирует следующую последовательность имен патчей: patch.diff, patch-2.diff, patch-3.diff,...

5.6 Зависимости

Когда число исправлений в проект растет, становится все труднее найти правильное место для добавления новых исправлений в серию патчей. В определенный момент, вы добавляете патч в конец, поскольку нахождение нужного места стало слишком сложным. В долгосрочной перспективе, беспорядок накапливается.

Чтобы избежать этого, сохраняя общую картину, команда quilt graph генерирует *dot* графики, показывающие зависимость между патчами.[5] Вывод этой команды можно изобразить с помощью инструментов, таких как AT&T

[5] Команда quilt graph вычисляет зависимости на основе того какие патчи меняют какие файлы и дополнительно проверяет на перекрывающиеся изменения в файлы. Хотя первый подход часто приводит к ложных срабатываниям, последний подход может привести к ложным негативам (то есть, quilt graph может не учитывать зависимостей).

Research's Graph Visualization Project (GraphViz, `http://www.graphviz.org/`). Команда quilt graph поддерживает различные виды графиков.

5.7 Расширенный Diffing

Quilt позволяет нам сравнивать и обновлять патчи не только на вершине стека (quilt diff -P *patch*) и quilt refresh *patch*). Это полезно в ряде случаев, например, когда патчи выше по стеку затрагивают те-же файлы, что и этот патч. Мы можем изобразить это как тень, которую бросают патчи выше по стеку на эти файлы. При обновлении патча, изменения в файлах, которые не являются теневыми (а значит, в последний раз были изменены патчем, который в настоящее время обновляется), принимаются во внимание, теневые изменения не будут обновлены.

Команда quilt diff позволяет объединить несколько патчей в один по желанию с указанием диапазона включаемых патчей (см. quilt diff -h). Комбинированный патч будет изменять каждый файл, содержащиеся в этих патчах только один раз. Результат применения комбинированного патча такой же, как применения всех патчей в указанном диапазоне последовательно.

Иногда бывает удобно использовать другой инструмент вместо *GNU diff* для сравнения файлов (например, графическая утилита Prog tkdiff ()). Quilt не будет использовать другие инструменты, кроме *GNU diff* при обновлении патчей (quilt refresh), но quilt diff принимает аргумент --diff=*utility*. С этим аргументом, указанный утилита вызывается для каждого файла, в который вносятся изменения с передачей в качестве аргументов исходного и измененного файлов. Для новых файлов, первый аргумент будет /dev/null, а для удаленных - второй.

Когда команде quilt diff передается список имен файлов, просмотр будет ограничен только этими файлами. С параметром -R меняются местами оригинальные и новые файлы, в результате чего получается обратный diff.

Иногда бывает полезно создать diff-файл между произвольным состоянием рабочего дерева и текущей версией. Это может быть использовано для создания различий между разными версиями патчей (см. раздел 5.5), и т.д. Для этой цели quilt позволяет сделать снимок рабочего каталога (quilt snapshot). Позднее файл различий с этим снимком рабочего деревф может быть создан с помощью команды quilt diff --snapshot.

В настоящее время поддерживается только один снимок. Он хранится в каталоге .pc/.snap. Для экономия свободного места на диске, он может быть удален командой quilt snapshot -d, или путем удаления каталога .pc/.snap вручную.

5.8 Работа с RPM пакетами

Несколько дистрибутивов Linux основаны на пакетном менеджере RPM [5]. RPM пакет состоит из спецификации, определяющий, как строится пакет, а также ряда дополнительных файлов, таких как TAR архивы, патчи, и

т.д. Большинство RPM пакетов содержат официальный пакет программного обеспечения а также ряд патчей. Прежде чем этими патчами можно манипулировать в quilt, должен быть создан файл series, который содержит список исправлений вместе с их уровнями вложенности.

Команда quilt setup автоматизирует сборку большинства пакетов RPM. При передаче spec-файла в качестве аргумента, она выполняет раздел %prep spec-файла, который должен извлечь официальный пакет программного обеспечения, а также применить патчи. В этом случае, quilt запоминет TAR архивов и патчи которые нужно применить, и создает файл series. На основании этого файла, quilt setup распаковывает архивы, а также копирует патчи в подкаталог patches. Некоторая мета-информация, такая как имена файлов хранится в виде комментариев в файле series. quilt setup также принимает файл series в качестве аргумента (который должен содержать некоторую мета-информацию), и предусматривает создание рабочего дерева из этого файла.

6 Настройка Quilt

После запуска, quilt выполняет файл .quiltrc в домашнем каталоге пользователя, или указанный в опции –quiltrc файл. Этот файл представляет собой обычный сценарий Bash. Параметры по умолчанию могут быть переданы в любой команде, определив переменную QUILT_*COMMAND*_ARGS (например, QUILT_DIFF_ARGS=-color=auto" подсвечивает вывод quilt diff при выводе на терминал).

В дополнение к этому, в quilt признаются следующие переменные:

QUILT_DIFF_OPTS Дополнительные параметры, которые quilt передает программе *GNU diff* при генерации патчей. Полезная опция для исходного кода на языке C "-p", которая заставляет программу *GNU diff* показать в результирующем патче функции которые были изменены.

QUILT_PATCH_OPTS Дополнительные опции, которые quilt передает программе *GNU patch* при применении исправлений. (Например, в некоторых версиях *GNU patch* поддерживает опцию "–unified-reject-files" для создания файлов откланенных изменений в стиле unified-diff.

QUILT_PATCHES Расположение файлов исправлений (см. раздел 3). По умолчанию этот параметр равен "patches".

7 Ловушки и известные проблемы

Как упоминалось ранее, файлы должны быть добавлены, прежде чем они могут быть изменены. Если игнорировать этот шаг, будет происходить одна из следующих проблем: если файл входит в патч ниже в стеке, то изменения будут опубликованы в этот патч при обновлении, и для этого патча команда

quilt pop завершится неудачно, до тех пор пока он не будет обновлен. Если файл не входит в какой-то патч, то будет изменен оригинальный файл в рабочем дереве.

Файлы патчей могут изменить тот же файл несколько раз. *GNU patch* содержит ошибку, которая повреждает резервные копиии файлов в этом случае. Исправление ошибки доступно , и будет интегрировано в более поздней версии *GNU patch*. Исправление уже включено в версию *GNU patch* из SUSE.

Есть некоторые пакеты, которые предполагают, что это хорошая идея, чтобы удалить все пустые файлы по рабочим деревьев, в том числе в каталоге .pc. Цель *make clean* в исходных кодах ядра Linux является примером. Quilt использует файлы нулевой длины в .pc для отметки файлов, добавленных патчами, поэтому такие пакеты могут повреждать каталог .pc. Чтобы обойти проблему, создайте символическую ссылку .pc в рабочем дерева, которая указывает на каталог снаружи.

Это может случиться, что файлы в каталоге patches рассинхронизируются с рабочим деревом (например, они могут быть случайно обновлены из CVS или других систем контроля версий). Файлы в каталоге .pc также могут быть рассинхронизированы, особенно если файлы не добавили до изменения их (quilt add / quilt edit). Если это произойдет, это возможно восстановить исходное дерево, но очень часто лучшим решением будет начинать все с исходной рабочей директории и подкаталога patches. Нет необходимости сохранять какие-либо файлы из каталога .pc в этом случае.

8 Резюме

Мы показали, как коллекция сценариев *quilt* решает различные проблемы, которые возникают при работе с патчами для программного обеспечения. Quilt гораздо лучше чем традиционные средства работы с патчами (*GNU patch*, *GNU diff*, и т.д.), и предлагает множество функций, которые недоступны в конкурирующих решениях. Вступайте в клуб!

Домашняя страница проекта - `http://savannah.nongnu.org/projects/quilt/`. Список рассылки разработчиков - `http://mail.nongnu.org/mailman/listinfo/quilt-dev`. Конечно, мы всегда рады дополнительным функциям, которые вы предложите добавить в quilt.

Переведено на русский: Ivan Borzenkov <ivan1986@list.ru>.

Список литературы

[1] Andrew Morton: Patch Management Scripts, `http://lwn.net/Articles/13518/` and `http://userweb.kernel.org/~akpm/stuff/patch-scripts.tar.gz`.

[2] Andreas Grünbacher et al.: Patchwork Quilt, `http://savannah.nongnu.org/projects/quilt`.

[3] IEEE Std. 1003.1-2001: Standard for Information Technology, Portable Operating System Interface (POSIX), Shell and Utilities, diff command, pp. 317. Online version available from the The Austin Common Standards Revision Group, `http://www.opengroup.org/austin/`.

[4] *GNU diff* info pages (info Diff), section *Output Formats.*

[5] Edward C. Bailey: Maximum RPM: Taking the Red Hat Package Manager to the Limit, `http://www.rpm.org/max-rpm/`.