Комплементарное хеширование подмножеств

Алексей Турбин
<at@altlinux.org>

9 апреля 2010 г.

1. Зависимости на разделяемые библиотеки

- Пакет с разделяемой библиотекой предоставляет зависимость Provides: libfoo.so.1.
- Пакет, который использует библиотеку, требует зависимость Requires: libfoo.so.1.
- Зависимость на имя библиотеки может быть недостаточной — при частичном обновлении могут появиться undefined symbols.
- Один из подходов — версионирование символов. Но его нельзя автоматизировать.

2. Модель зависимостей с учетом символов

- Символы ассоциируются с библиотеками.
- Пакет с разделяемой библиотекой предоставляет зависимость Provides: libfoo.so.1 и множество символов foo1, foo2, foo3.
- Пакет, который использует библиотеку, требует зависимость Requires: libfoo.so.1 и множество символов foo1, foo2, ...
- Зависимость считается удовлетворенной если множество требуемых символов является *подмножеством* предоставляемых символов (т.е. все требуемые символы предоставлены).

3. Комплементарное хеширование

- Модель громоздка. Но вообще-то нам и не нужно хранить полный список символов. Нужно лишь уметь проверять, является ли требуемые символы подмножество предоставляемых.
- Нельзя ли придумать такую процедуру хеширования двух множеств, при которой существует возможность проверить вложение?
- Пакет с разделяемой библиотекой предоставляет зависимость Provides: libfoo.so.1 = set:7f0252c3...
- Пакет, который использует библиотеку, требует зависимость Requires: libfoo.so.1 >= set:3f5b289c...

4. В чистом виде комплементарное хеширование невозможно

- Нельзя придумать комплементарный хеш фиксированной длины.
- Потому что всевозможных подмножеств слишком много — 2^N, и все подмножества должны быть различимы (то есть иметь различный хеш).
- Другое объяснение — с точки зрения количества информации любой хеш можно заменить битовой шкалой, на которой отмечены элементы подмножества. Тогда ясно, что число битов растет как O(N).
- Значит, речь идёт лишь о более компактном кодировании.

5. Компактное кодирование множества строк

- Каждый элемент можно кодировать отдельно, используя 20-32 битный хеш.
- Используя специальную процедуру упаковки и распаковки элементов (дельта-кодирование, «Кодирование сочетаний» Чистикова), последовательность элементов можно сжать.
- Односторонняя ошибка с вероятностью не хуже 10^{-3}.
- Если использовать хеш по маске (а не по модулю), то Requires и Provides хеши могут быть разной битности.

6. Недостатки

- Это более компактное представление, но оно всё же недостаточно компактно - «версия на полэкрана».
- Нельзя диагностировать отсутствующий символ, непонятное сообщение о неудовлетворенной зависимости.
- Привносит элемент вероятности, то есть защита может не сработать.
- Будет долго выполняться проверка неудовлетворенных зависимостей.