

# Pkg - система управления пакетами FreeBSD

Всеволод Стахов  
vsevolod@FreeBSD.org



FreeBSD

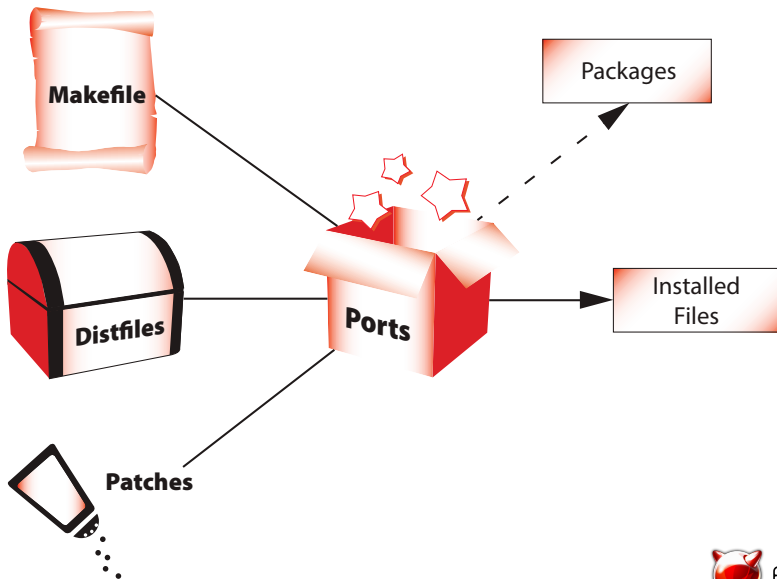
ruBSD conference 14 Декабря 2013

# Порты и пакеты

Порты - основа для создания пакетов.

- ▶ Устоявшаяся система
- ▶ Понятные и четкие правила
- ▶ Простые в создании и управлении (не всегда)
- ▶ Множество настроек

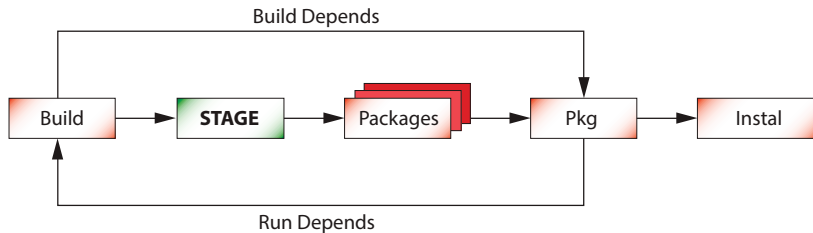
# Порты без pkg



# Недостатки предыдущей схемы

- ▶ Make не может полноценно разрешать зависимости и конфликты
- ▶ Сложно обновлять и держать в актуальном состоянии
- ▶ Проблемы при миграции между версиями ОС
- ▶ Процесс компиляции из исходников занимает время

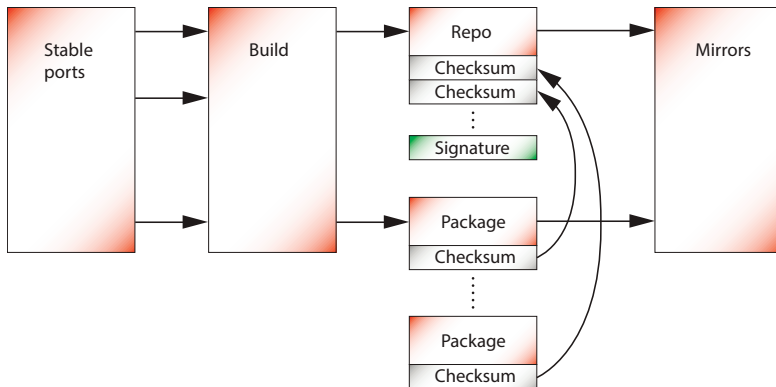
# Планируемая архитектура взаимодействия pkg и портов



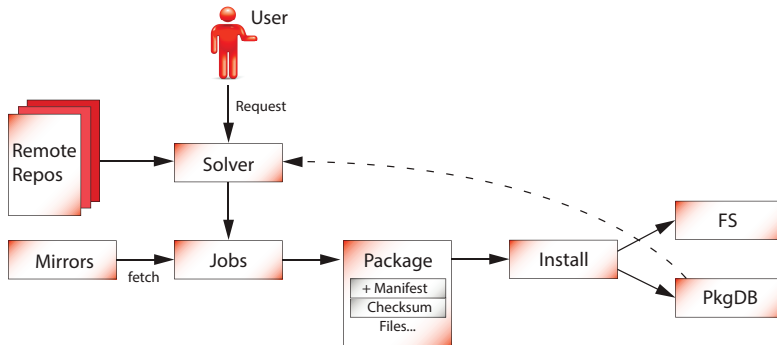
# Порты и пакеты

- ▶ Порты используются для построения пакетов
- ▶ Зависимости и конфликты обрабатываются pkg, а не make
- ▶ Стабильная ветка портов соответствует стабильной ветке пакетов
- ▶ Пользователям рекомендуется ставить ПО из пакетов
- ▶ Не запрещая установку из портов

# Создание репозитория пакетов



# Установка пакетов в систему





# Проблемы в pkg





- ▶ Поддержка устаревших портов (например, без stage directory)
- ▶ Крайне ограниченная система зависимостей
- ▶ Примитивный алгоритм разрешения зависимостей (solver)

# Проблемы с алгоритмом solver'a

- ▶ Нет поддержки конфликтов
- ▶ Не работает с альтернативными пакетами
- ▶ Может выполнять только одну задачу: установить, удалить или обновить некоторые пакеты

# Существующие системы управления пакетами

Другие системы управления пакетами используют различные подходы для решения данной задачи:

- ▶  Zypper/SUSE - построена на базе libsolv
- ▶  Yum/RedHat - планирует переход на libsolv
- ▶  Apt/Debian - базируется на собственной логике разрешения зависимостей и конфликтов
- ▶  Pacman/Archlinux - использует собственный наивный алгоритм

# Независимые solver'ы

Для взаимодействия с внешними solver'ами, предполагается использовать формат CUDF, разработанный в рамках исследовательского проекта Mancoosi <http://mancoosi.org>:

```
package: devel/libblah  
version: 1  
depends: x11/libfoo
```

```
package: security/blah  
version: 2  
depends: devel/libblah  
conflicts: security/blah-devel
```

# Но нам нужен собственный solver!

Варианты:

- ▶ Реализовать собственный алгоритм разрешения зависимостей?

# Но нам нужен собственный solver!

Варианты:

- ▶ Реализовать собственный алгоритм разрешения зависимостей?
- ▶ Использовать существующие решения?

# Но нам нужен собственный solver!

Варианты:

- ▶ Реализовать собственный алгоритм разрешения зависимостей?
- ▶ Использовать существующие решения?
- ▶ Использовать известный алгоритм?

# Но нам нужен собственный solver!

Варианты:

- ▶ Реализовать собственный алгоритм разрешения зависимостей?
- ▶ Использовать существующие решения?
- ▶ Использовать известный алгоритм?

SAT solver для управления пакетами

$$\overbrace{(x_1 \parallel \neg x_2 \parallel x_3) \& (x_3 \parallel \neg x_1) \& (x_2)}^{\text{SAT выражение}}$$

Условие



# Составление выражения SAT

- ▶ Каждой версии каждого пакета назначается независимая переменная:  $A \rightarrow a_1$ ,  $B \rightarrow b_1$
- ▶ Пользовательский запрос преобразуется в набор унарных условий:
  - ▶ Установить или обновить пакет  $A \rightarrow (a_1)$
  - ▶ Удалить пакет  $B \rightarrow (\neg b_1)$
- ▶ Преобразовать правила зависимостей и конфликтов в условия

# Преобразование конфликтов и зависимостей

- ▶ Если пакет  $A$  зависит от пакета  $B$  (версий  $B_1$  и  $B_2$ ), тогда либо пакет  $A$  не установлен, либо установлена одна из версий пакета  $B$ :

$$(\neg A \parallel B_1 \parallel B_2)$$

# Преобразование конфликтов и зависимостей

- ▶ Если пакет  $A$  зависит от пакета  $B$  (версий  $B_1$  и  $B_2$ ), тогда либо пакет  $A$  не установлен, либо установлена одна из версий пакета  $B$ :

$$(\neg A \parallel B_1 \parallel B_2)$$

- ▶ Если разные версии пакета  $B$  ( $B_1$ ,  $B_2$  and  $B_3$ ) конфликтуют между собой, тогда только одна из версий пакета  $B$  может быть установлена:

$$\underbrace{(\neg B_1 \parallel \neg B_2) \& (\neg B_1 \parallel \neg B_3) \& (\neg B_2 \parallel \neg B_3)}_{\text{Цепочка конфликтов}}$$

# Алгоритм решения задачи SAT

Способы упростить решение задачи SAT.

- ▶ Очевидные назначения
- ▶ Назначения юнитов - назначение переменных в условиях, содержащих только одну неназначенную переменную и не являющихся истинными
- ▶ Обучение на базе конфликтов: если в ходе назначения мы обнаружили конфликт, то движемся по дереву вверх, до тех пор, пока все выражение является ложным, после чего инвертируем назначение, вызвавшее конфликт
- ▶ Предположения, специфичные для задачи управления пакетами

# Назначения переменных

- Очевидные назначения - переменные, непосредственно входящие в запрос пользователя (установка или удаление)

$$(\neg A \parallel B) \& \underbrace{(A)}_{true} \& \underbrace{(\neg C)}_{false} \& (\neg A \parallel \neg D)$$

# Назначения переменных

- ▶ Очевидные назначения - переменные, непосредственно входящие в запрос пользователя (установка или удаление)

$$(\neg A \parallel B) \& \underbrace{(A)}_{true} \& \underbrace{(\neg C)}_{false} \& (\neg A \parallel \neg D)$$

- ▶ Назначения юнитов - простые зависимости и конфликты

$$\underbrace{(\neg A \parallel B)}_{\substack{\text{Зависимость} \\ B \rightarrow true}} \& \underbrace{(A)}_{true} \& \underbrace{(\neg C)}_{false} \& \underbrace{(\neg A \parallel \neg D)}_{\substack{\text{Конфликт} \\ D \rightarrow false}}$$

# Обучение на основе конфликтов

Для обработки альтернативных решений необходимо перебирать все возможные варианты назначений:

1. полный перебор в глубину
2. возврат, если был обнаружен конфликт
3. определить, какое назначение вызвало конфликт
4. инвертировать такое назначение и продолжить поиск, отсекая конфликтную ветвь

# Особенности задачи управления пакетами

Решение задачи SAT в чистом виде может быть сильно ускорено, если принимать во внимание специфику исходной задачи:

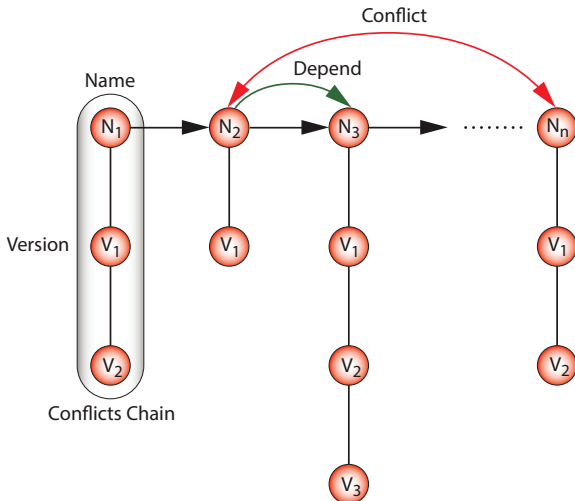
- ▶ не пытаться удалить уже установленные пакеты (если не обнаружено конфликтов)
- ▶ не устанавливать пакеты, если они явно не нужны
- ▶ предпочитать установку более приоритетных пакетов, учитывая приоритет репозиторий

Эти предположения также улучшают начальное назначение, отсекая заведомо неверные ветви дерева.



# Множество пакетов

Мы представляем все пакеты, вовлеченные в запрос пользователя (все прямые и обратные зависимости, а также конфликты) в виде множества пакетов:



# Задача управления пакетами

- ▶ Запрос пользователя превращается в набор задач по установке или удалению пакетов
- ▶ Все явные и неявные конфликты проверяются во время создания репозитория
- ▶ Создается множество пакетов на основании всех зависимостей и конфликтов пакетов, входящих в запрос
- ▶ Множество пакетов строится на базе имени и уникальной версии, которая определяется всеми значащими полями пакета (версия, ревизия, контрольные суммы)

# Применение алгоритмов solver'ов в pkg

- ▶ Pkg может использовать множество пакетов для работы с внешним solver'ом:
  - ▶ формализация версий
  - ▶ генерация запроса
  - ▶ обработка решения
- ▶ Также можно использовать встроенный SAT solver:
  - ▶ преобразовать множество пакетов в булево выражение
  - ▶ присоединить запрос
  - ▶ ???
  - ▶ PROFIT

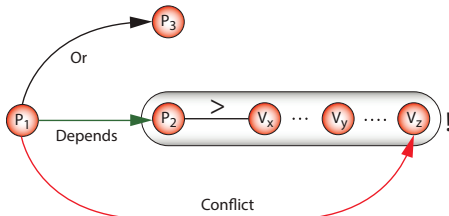
# Возможности

- ▶ Использование pkg для управления портами
- ▶ Лучшая поддержка нескольких репозиториев
- ▶ Проверка алгоритмов управления пакетами (через CUDF формат)
- ▶ Продвинутый формат зависимостей и конфликтов
- ▶ Альтернативные пакеты

# Новый формат зависимостей

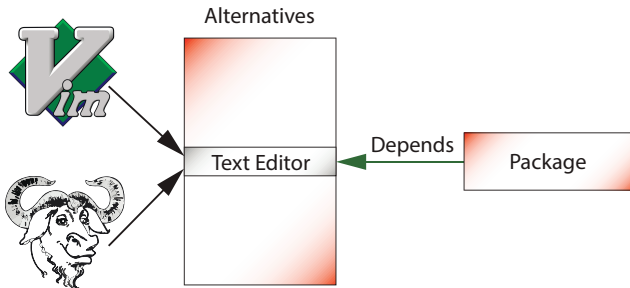
*libblah*  $\geq 1.0 + option_1, + option_2 || libfoo! = 1.1$

- ▶ Зависимости от реальных и виртуальных пакетов
- ▶ Более гибкий вариант определения версий
- ▶ Использование альтернатив



# Альтернативные пакеты

- ▶ Используются для определения некоторой общей функциональности (например, браузер)
- ▶ Могут использоваться в зависимостях (т.н. виртуальные зависимости)





freeBSD

Спасибо за внимание!

*Вопросы?*

vsevolod@FreeBSD.org

