

Перевод дистрибутива ROSA на пакетные менеджеры



RPM 4 и DNF



Михаил Новосёлов

`m.novosyolov@rosalinux.ru`

`mikhailnov@nixtux.ru`

`nixtux.ru`

Алексей Киселёв

`a.kiselev@rosalinux.ru`

Дистрибутив ROSA

- Mandrake → Mandriva + Connectiva → ROSA
- Самостоятельная пакетная база
- Международное сообщество мейнтейнеров (преимущественно сотрудники ROSA)
- Повышенное внимание качественной русской локализации
- Исторически сложившаяся ориентированность на десктопное применение

Репозиторий ROSA

- Разбивка на подпакеты, выделение библиотек в отдельные пакеты в соответствии с `so name`
- Префикс `lib64` у пакетов с 64-битными библиотеками и `lib` у 32-битных
libfoo.so.2 → *lib64foo2*, *libfoo-1.24.so.2* → *lib64foo1.24_2*
- Один из самых больших репозиториев, но мало мейнтейнеров
- Стилистическое единообразие сборочных инструкций (*.spec)
- Строгое соблюдение принятых политик упаковки во всех пакетах
- Разделение на репозитории: `main`, `contrib`, `non-free`, `restricted`
- Защита от недолинковки во флагах линковщика

Жизненный цикл ROSA

- Платформа — базовая система с зафиксированными версиями системных компонентов (glibc, gcc и др.)
rosa2016.1 — стабилизированная платформа (rpm5), **rosa2019.1 — в разработке (rpm4)**, rosa2019.05 — сертифицируемая платформа (rpm4)
- Пользовательские приложения обновляются роллингом (например, версия LibreOffice в платформе не зафиксирована)
- Платформа стабилизируется (1-2 года), на ней выпускается корпоративный дистрибутив, платформа переходит в фазу сопровождения, разработка переходит на новую платформу (технологический цикл нарушился в 2016 г.)

Задачи пакетной системы дистрибутива GNU/Linux

- Обеспечение целостности (работоспособности) системы
 - Установка необходимых для работы ПО зависимостей
 - Контролирование, что зависимости другого ПО не ломаются при установке нового пакета
 - Минимизация человеческого фактора — автоматическое прописывание и разрешение зависимостей
 - Предотвращение влияния одного ПО на другое (например, перезаписи файлов)

Задачи пакетной системы

- расман в Arch Linux: просто, быстро, ненадежно
 - Высокая скорость работы
 - Отсутствие разделения на подпакеты (/usr/bin/ffmpeg, /usr/lib64/libavcodec.so.N, /usr/include/ffmpeg и др. в одном пакете ffmpeg)
 - Зависимости не привязаны к конкретной so name
 - /usr/bin/mpv слинкован с libavcodec.so.N
 - но зависимости прописана от ffmpeg (причем **вручную**, могли бы и забыть прописать)
 - при смене N на N+1 зависимость не ломается → пакетный менеджер поставит несовместимые версии ffmpeg и mpv, не пикнув
 - а при запуске mpv libavcodec.so.N не оказывается в системе → *целостность (работоспособность) системы не проконтролирована пакетной системой*

Зависимости пакетов в мире RPM

- Встроенные и подключаемые внешние генераторы значений тегов RPM на основе файлов пакета:
Requires, Provides, OrderWithRequires
- Генерация сборочных зависимостей на основе исходников
BuildRequires: cargo(foo)
- Автоматическое проставление зависимостей для ELF:
libfoo.so.2()(64bit)
- Учитывание версионирования символов:
libc.so.6(GLIBC_2.17)(64bit)
- `python3egg(foo)`, `python3.8dist(foo) >= X`
- `(python3.8dist(foo) >= X and python3.8dist(foo) < Y)`
- `/usr/bin/foo`, `/usr/share/foo/file.png`
- Зависимости для нужного этапа жизни пакета:
Requires(pre), Requires(post), Requires(postun)
OrderWithRequires(pre), OrderWithRequires(post), OrderWithRequires(postun)
pretrans, pre, post, preun, postun, posttrans
- Генерация зависимостей для скриплетов (только в rpm-build 4.0 в ALT Linux) (можно сделать `OrderWithRequires`)
- Генерация зависимостей на основе используемого пакетом ABI (какие символы из зависимых библиотек используются (их много!)) (только в ALT Linux, не соответствует механизму `vercmp` в `librpmio`, в Fedora проверяется вне пакетной системы)
- `devel()`: зависимости для `devel`-пакетов от `devel`-пакетов с библиотеками, с которыми слинкованы библиотеки, симлинки на которые поставляются в `devel`-пакетах
<https://abf.io/import/devel-rpm-generators>
- Механизмы `Obsoletes`, `Conflicts`
- Механизм триггеров
- `dnf repoclosure` — проверка замкнутости репозитория по зависимостям



Статья [Обсуждение](#)

Переход ROSA с RPM 5 на RPM 4

Ссылки сюда (1) →

Начиная с gosa2019.1, дистрибутив ROSA Fresh переходит с пакетных менеджеров RPM 5 и игpm1 на RPM 4 и DNF.

Содержание [\[убрать\]](#)

- 1 Откуда куда переход
- 2 Причины для перехода
- 3 Общий план перехода
- 4 Изменения во флагах компилятора (CFLAGS, %optflags) по умолчанию
- 5 Особенности перевода спеков на RPM 4

Откуда куда переход [\[править\]](#)

Более восьми лет в дистрибутивах ROSA Desktop использовался пакетный менеджер [RPM5](#) - форк RPM4, созданный Однако постепенно активность по разработке RPM5 угасла, а RPM4 наоборот - возродился и постепенно не только вообр а ROSA Fresh переходит обратно на RPM 4.

Было:

- низкоуровневая система управления пакетами RPM 5.4.10 (с более чем сотней патчей, специфичных для РОСЫ)
- высокоуровневый пакетный менеджер игpm1
- mosk-игpm

Стало:

http://wiki.rosalab.ru/ru/index.php/Переход_ROSA_с_RPM_5_на_RPM_4

Яндекс

переход на rpm4



Найти



Поиск

[Картинки](#)

[Видео](#)

[Карты](#)

[Маркет](#)

[Новости](#)

[Переводчик](#)

[Эфир](#)

[Кью](#)

[Услуги](#)

[Музыка](#)

[Все](#)

[Переход ROSA с RPM 5 на RPM 4](#) — Rosalab Wiki

[wiki.rosalab.ru](#) > [ru/index.php/Переход...RPM...на_RPM_4](#) ▾

Начиная с gosa2019.1, дистрибутив ROSA Fresh **переходит** с пакетных менеджеров RPM 5 и игpm1 на **RPM 4** и DNF. Эта статья описывает основные отличия для пользователей и сборщиков пакетов. [Читать ещё](#) >

Нашлось 18 млн результатов

[Дать объявление](#)

Было (пакетная система на базе RPM 5 + urpmi)

- Низкоуровневый пакетный менеджер RPM 5.4.10 с более чем сотней патчей
 - Форк RPM 4, сделанный основным разработчиков
 - Спешный переход Mandriva на RPM 5
 - Большое кол-во костылей, конфликт разработчиков
 - Запутанный код, не признание стилистики кода Джеффом
 - Нестабильность новых версий RPM 5, использование старой версии с накопившимися патчами
 - Зброшена разработка RPM 5, возрождена разработка RPM 4 (но не Джеффом)
- Высокоуровневый пакетный менеджер urpmi (+ perl-URPM)
 - Основная логика в perl-URPM
 - urpmi как высокоуровневый интерфейс для perl-URPM и качалка пакетов + другие утилиты
 - Блокировки в обход штатных механизмов RPM
- Средство воспроизводимой сборки пакетов mock-urpm
- Практически неработающие биндинги PackageKit<->urpmi

Стало (RPM 4 + DNF)

- Низкоуровневый пакетный менеджер rpm 4.16 (начинали с 4.15.1)
- Высокоуровневый пакетный менеджер DNF
- libsolv для разрешения зависимостей
- Исчез RPM-тег DistEpoch (не путать с Epoch), но появился осмысленный DistTag
- Вместо mock-urpm используется оригинальный mock
- Наиболее часто используемые в командах urpmi и urpme функции преобразовываются в команды dnf (dnf-URPM)
- Будет работающий PackageKit, на основе которого планируется графический «Магазин приложений»
- Из проблем: медленное обновление метаданных dnf на маломощных процессорах

Стабилизация разрешения зависимостей

- Проблема:
 - Образ ISO собирается из 3500 пакетов
 - На вход подается просто список пакетов, без специальной подготовки
 - Порядок установки пакетов должен быть:
 - Правильным (если пакет А в скриптлете вызывает утилиты из пакета Б, то пакет Б должен быть уже установлен перед началом установки пакета А)
 - Стабильным (одинаковым при каждом запуске сборки образа)

Стабилизация разрешения зависимостей

- Сборка образа из 3500 пакетов с помощью `urpmi`:
 - Сначала ставим пакет `basesystem` с зависимостями
 - Затем все остальные пакеты
 - В обоих случаях много раз в цикле запускаем `urpmi`, который успешно ставит N пакетов и падает из-за невозможности разрешить зависимости, затем еще M пакетов, снова падает и т. д., пока он не поставит все пакеты
 - Закольцованные зависимости разрываются случайным образом, раз от раза разрывы разные
 - Ошибки плавающие, раньше не было автоматизированного перезапуска `urpmi`
 - Часть пакетов могут оказаться установлены, но отсутствовать в БД RPM (о БД позже)

Стабилизация разрешения зависимостей

- Сборка образа из 3500 пакетов с помощью dnf:
 - Запускаем установку сразу всех пакетов, не имея готового chroot с целевой системой
 - dnf ставит все пакеты в не меняющемся раз от раза порядке
 - Чистый лог, удобство отслеживания ошибок скриплетов

Этапы стабилизации разворачивания системы из множества пакетов с нуля

- Обеспечить установку filesystem (костяк файловой системы) и setup (/etc/passwd и пр.) в самом начале транзакции
 - Доработка RPM для возможности генерировать OrderWithRequires <https://abf.io/import/order-rpm-generators>
 - «OrderWithRequires: setup filesystem» — во все пакеты
- Обеспечить установку systemd до манипуляций с сервисами и до создания пользователей через systemd-sysusers (обеспечение правильных владельцев файлов и каталогов)
 - OrderWithRequires: systemd
 - OrderWithRequires: chkconfig
- Исправить ошибки в скриптелях разных пакетов, вызванных недостатком зависимостей
 - Requires(pre): grep, если в %pre вызывается grep
- Разорвать закольцованные зависимости
 - Ручная аналитическая проработка выявленных узких мест

Бутстрап RPM 4

- Имелась работающая система на RPM 5 + urpmi
- Задача:
 - Собрать rpm4
 - Просто компилирование кода на Си
 - Собрать dnf
 - Сборка зависимостей на Си и пакетов на Python
 - Обеспечить работу с пакетами, собранными RPM 5
 - Игнорирование distepoch в конструкции epoch:version-release:distepoch (где epoch есть не всегда) в rpm4, libsolv
 - Обеспечить конвертирование БД RPM 5 в БД RPM 4
 - Сделать chroot с rpm4+dnf+mock+createrepo_с+старыми пакетами остальных компонентов
 - Развернуть билдер ABF (docker-контейнер) с этой системой
 - Адаптировать RPM-спеки (*.spec)
 - Пересобрать весь репозиторий

<https://pagure.io/omv-urpmi-to-dnf-migration> (Neal Gompa, ngompa@, Conan-Kudo)

Обеспечение преемственности

- dnf-URPM для конвертирования команд `urpmi` и `urpme` в `dnf`
- Rpm Drake → Dnfdragora
- Совместимость `rpm4+dnf` с пакетами, собранными на RPM 5
 - Полезно для старых библиотек (`lib64fooN` с `/usr/lib64/libfoo.so.N`) и проприетарных пакетов (например, `WINE@Etersoft`)
 - Игнорирование `distepoch` в конструкции `epoch:version-release:distepoch` (где `epoch` есть не всегда) в `rpm4`, `libsolv`
- Совместимость новых пакетов со старым стеком `rpm5+urpmi` не обеспечивается
 - Нет `boolean`-зависимостей
 - Нет `OrderWithRequires`
 - Нет поддержки сжатия `ZSTD` (но применяется `xz -T0 -2` для `SRPM` и `xz -T0 -6` для `RPM`)
 - Нет потребности в такой совместимости
- Совместимость макросов
- В RPM 5: `Recommends` → `Requires(missingok)` ← `Suggests`
- В RPM 5:
 - `%__requires_exclude(_from)` → `%__noautoreq(files)`
 - `%__provides_exclude(_from)` → `%__noautoprov(files)`

<https://abf.io/soft/rpm5>

<https://abf.io/import/rpm>

Перевод дистрибутива ROSA на пакетные менеджеры



RPM 4 и DNF



Михаил Новосёлов

`m.novosyolov@rosalinux.ru`

`mikhailnov@nixtux.ru`

`nixtux.ru`

Алексей Киселёв

`a.kiselev@rosalinux.ru`