Разработка прикладного ПО, совместимого с продуктами экосистемы РОСА

Михаил Новоселов
m.novosyolov@rosalinux.ru
Алексей Киселев
a.kiselev@rosalinux.ru

rosalinux.ru

Экосистема РОСА

ОФИС

РОСА «КОБАЛЬТ»

РОСА «ХРОМ»

УДАЛЕННЫЙ ДОСТУП

РОСА «БАРИЙ»

ПУБЛИЧНОЕ ОБЛАКО

РОСА «ХРОМ»

ЦОД

УПРАВЛЕНИЕ

ЦЕНТР УПРАВЛЕНИЯ

МЕНЕДЖЕР РЕСУРСОВ

ФИЗИЧЕСКИЕ СЕРВЕРА

РОСА «КОБАЛЬТ»

РОСА «ХРОМ»

ВИРТУАЛИЗАЦИЯ

ROSA VIRTUALIZATION

РОСА «КОБАЛЬТ»

РОСА «ХРОМ»

Суверенный дистрибутив

- **Своя техническая политика**
  - Сами определяем целевой результат
    - По функционалу (наши идеи и пожелания пользователей и заказчиков)
    - По удобству для пользователя/администратора
    - По стабильности
    - По поддерживаемым архитектурам
  - Сами делаем пакеты
  - Сами определяем их версии
  - Сами проводим их «боевое слаживание»
  - Сами взаимодействуем с апстримами
- **Самопересобираемость всего репозитория**
  - В любой компонент можно внести правки и пересобрать пакет без доступа в интернет

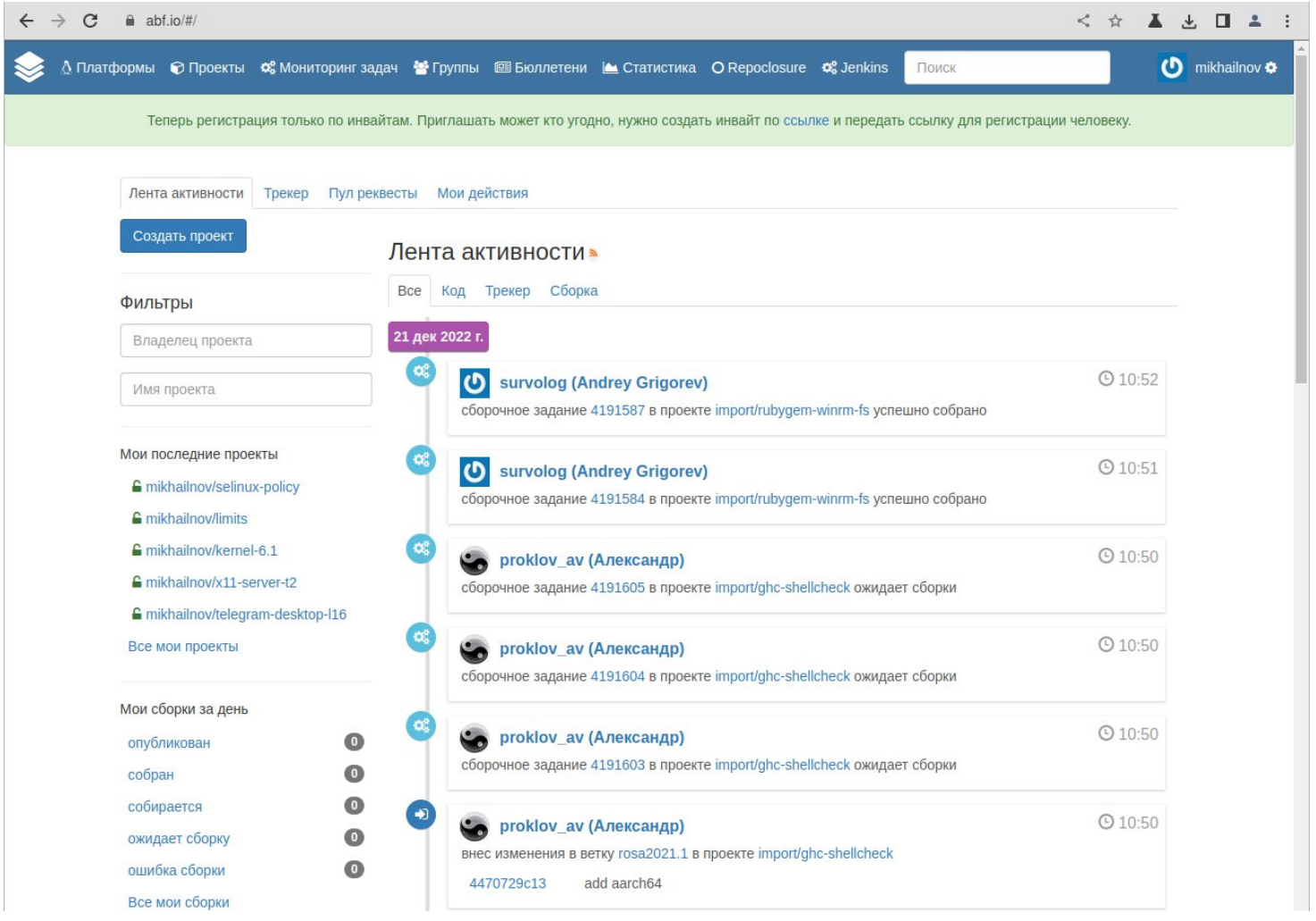
Платформа rosa2021.1

- Платформа = репозиторий
- 35.5 тысяч бинарных пакетов, 18 тысяч исходных
- Поддерживаются архитектуры:
  - i686 (32-битная x86)
  - x86_64 (64-битная x86)
  - aarch64 (ARMv8 64-битная)
  - e2kv4 (Эльбрус)
  - riscv64 (RISC-V 64-битная)

синхронная сборка

- Установочные образы:
  - Сервер (минималистичная система без DE)
  - KDE Plasma 5
  - XFCE
  - GNOME
  - LXQt

Среда разработки и сборки ABF.io



Id	Статус	Проект	Версия
4182315	[testing] опубликован	import/guestfs-tools	1.48.2-1
4191094	опубликован	import/kernel-6.1	6.1.0-3
4184879	опубликован	import/opensmtpd	6.8.0p2-5
4184878	опубликован	import/opensmtpd	6.8.0p2-5
4184877	опубликован	import/opensmtpd	6.8.0p2-5
4184876	опубликован	import/opensmtpd	6.8.0p2-5
4184875	опубликован	import/opensmtpd	6.8.0p2-5
4187888	ошибка сборки	import/qcalcfilehash	
4187845	ошибка сборки	import/qcalcfilehash	
4181319	опубликован	import/dracut	053-0.git5eb736.11

Дистроуниверсальная сборка

- Ваш исполняемый файл запускается на разных ОС
  - Во всех популярных дистрибутивах glibc
  - Основные библиотеки тоже бинарно совместимы
- Пакетные менеджеры + репозитории разных RPM-дистрибутивов понимают зависимости вашего пакета

Цель:

- Собрали бинарник
- Упаковали его в RPM (1 на все дистрибутивы)
- Упаковали его в DEB

Автоматизация зависимостей RPM

Provides (что можем дать)

В пакете файл `/usr/lib64/libfoo.so.0`

rpmbuild запускает генератор Provides

Генератор видит, что это:
- 64-битный ELF
- разделяемая библиотека
- soname (внутри ELF) `libfoo.so.0`

И выдает результат:
`libfoo.so.0()(64bit)`

Requires (что хотим взять)

В пакете файл `/usr/bin/foo`

rpmbuild запускает генератор Requires

Генератор видит, что это:
- 64-битный ELF
- В ELF зависимость от `libfoo.so.0`

И выдает результат:
`libfoo.so.0()(64bit)`

~~UPX~~

При запуске `/usr/bin/foo`:
- запускается интерпретатор (прописан в ELF) `ld-linux`
- он находит `/usr/lib64/libfoo.so.0`

Дистроуниверсальный RPM

- Исполняемые файлы в нем полагаются только на общие для дистрибутивов ABI
- Что ABI нестабилен — в целом скорее миф
- Зависимости дистроуниверсальны

Плохая зависимость	Дистроуниверсальная зависимость
• libgnomekeyring • lib64gnome-keyring0	libgnome-keyring.so.0()(64bit)
fonts-ttf-opensans	font(opensans)
libmtp-utils	/usr/bin/mtp-detect

`rpm -q pkg -P` показывает Provides пакета

Фильтруйте провайды

- В вашем пакете лежит файл `/opt/xxx/libfoo.so.0`
- `ld-linux` не будет искать `libfoo.so.0` в `/opt/xxx` при запуске других программ
- Но RPM автоматически добавляет Provides `libfoo.so.0()(64bit)`
- Зафильтруйте такие Provides и самозависимости (Requires) от них
- Не стоит полностью отключать AutoReq и прописывать [не] все зависимости от библиотек вручную

Избегайте скриптлетов

Скриптлет — это скрипт, выполняемый до или после установки пакета

НЕТ: `cp /opt/xxx/icon.png /usr/share/icons/`

ДА: упаковать `/usr/share/icons/hicolor/scalable/apps/xxx.svg`

```
rpm -q file.rpm --scripts
```

Избегайте скриптлетов

Файловые триггеры сделают все нужное и без ваших **кривых** скриптов

Их работу видно: `rpm -Uvvvh file.rpm`

НЕТ: `xdg-mime install --mode system /opt/<...>/xxx-docxf.xml`

ДА: упаковать `/usr/share/mime/packages/xxx-docxf.xml`

```
%transfiletriggerin -- /usr/share/mime/packages/
update-mime-database -n /usr/share/mime > /dev/null

%transfiletriggerun -- /usr/share/mime/packages/
update-mime-database -n /usr/share/mime > /dev/null
```

Избегайте скриптлетов

Не влезайте в файлы других пакетов!

НЕТ: скриптами править `/usr/share/applications/mimeapps.list`

ДА: `MimeType=поддерживаемые_MIME-типы` в `*.desktop` достаточно

См. стандарты FreeDesktop:
https://specifications.freedesktop.org/

RPATH для библиотек с собой

- RPATH/RUNPATH — в каких нестандартных папках ld-linux будет искать библиотеки при запуске бинарника
- (ld-linux и glibc системные, вы же не понесете их с собой...?)
- ✖ НЕТ: переменная `LD_LIBRARY_PATH=/opt/xxx`
- ✔ ДА: `RPATH /opt/xxx` или `$ORIGIN`
- Переменная окружения наследуется при запуске дочерних процессов, которым ваши библиотеки не подходят

  Офис -> браузер -> VLC -> Qt не подходит

```
patchelf --print-rpath file
```

ROSA

Осторожно с libstdc++.so.6

- libstdc++.so.6 может быть старой в целевых ОС
- Носить ее с собой можно только если вы не зависите от системных библиотек, зависящих от [более новой, чем ваша] системной libstdc++.so.6
- Потому что прогрузится только первая libstdstdc++.so.6 — ваша
- Вариант: статическое связывание
- Вариант: переименовать ее в libstdstdc++-gccX.so.6 и положить с собой

Бывают нюансы в glibc

В большинстве дистрибутивов:

```
$ readelf -a /lib64/libc.so.6 | grep __libc_enable_secure
00000017fdb8  000600000401 R_AARCH64_GLOB_DA
0000000000000000 __libc_enable_secure@GLIBC_PRIVATE + 0
```

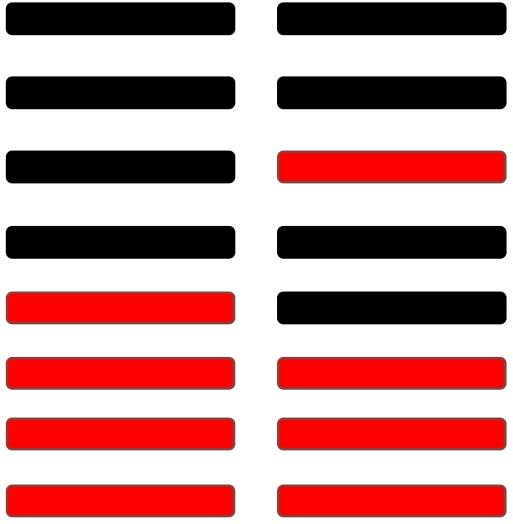
В ALT Linux:

```
$ readelf -a /lib64/libc.so.6 | grep libc_enable_secure
00000000019ce78  0000000c00000401 R_AARCH64_GLOB_DAT
0000000000000000 __libc_enable_secure@GLIBC_2.17 + 0
    12: 0000000000000000     0 OBJECT  GLOBAL DEFAULT  UND
__libc_enable_secure@GLIBC_2.17 (20)
```

https://lists.altlinux.org/pipermail/devel/2020-April/210682.html

ROSA

Цепочки библиотек



Черным — свои библиотеки

Красным — системные

Избегайте вклинивания в цепочку

```
$ patchelf --print-needed ./usr/local/bin/[redacted]/lib/libxkbcommon-x11.so.0
libxkbcommon.so.0
libxcb.so.1
libxcb-xkb.so.1
libc.so.6
```

Однако в поставке нет библиотеки `libxcb-xkb.so.1`:

```
$ find . -name '*.so*' | grep libxcb-xkb.so
$
```

Дистроуниверсальная сборка

- Ваш исполняемый файл запускается на разных ОС
  - Во всех популярных дистрибутивах glibc
  - Основные библиотеки тоже бинарно совместимы
- Пакетные менеджеры + репозитории разных RPM-дистрибутивов понимают зависимости вашего пакета

Цель:

- Собрали бинарник
- Упаковали его в RPM (1 на все дистрибутивы)
- Упаковали его в DEB