Информационный Технологический Бюллетень

№ 7

24 июня 2013 г.

РОСА

Just Open

Оглавление

Вступительное слово

Здравствуйте! В этот летний день предлагаем вам освежиться бодрящей информацией нашего нового бюллетеня. Это седьмой по счёту выпуск «Точки РОСЫ» Вообще, семь — необычное число: в индуистской мистике это символ здоровья, 7 – число тайны и мистического знания, число сказок и древних трактатов: семь планет, семь цветов радуги, семь нот, семь дней в неделе. Мы надеемся, что материалы этого выпуска также помогут вам приобщиться к техническим тайнам СПО и узнать самые актуальные новости компании «РОСА». Редакция всегда готова ответить на ваши вопросы, пишите нам rosa-point@rosalab.ru. Приятного чтения!

Desktop

Центр управления LXDE, или маленький эксперимент на острие тенденций

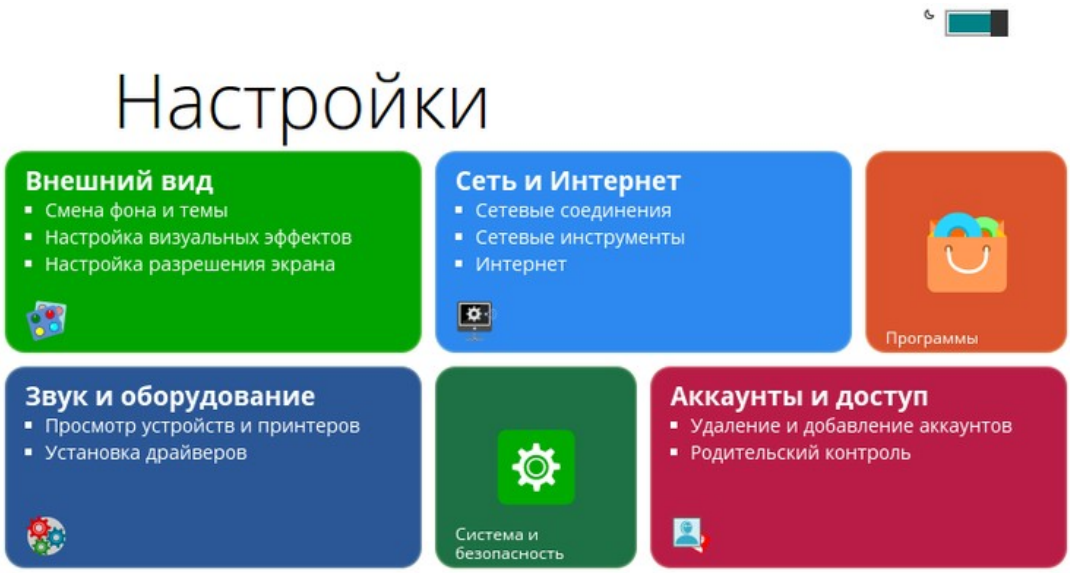
В LXDE никогда не было своего центра управления. Среда минималистична по сути, и ее авторы предлагали использовать отдельные утилиты, как свои (lxappearance, lxrandr и другие), так и входящие в другие DE или вообще системные.

С другой стороны, попытки сделать единый центр предпринимались в разных дистрибутивах. Одним из таких был аргентинский дистрибутив Toquito (http://distrowatch.com/table.php?distribution=tuquito), в котором сделали простой Центр управления на основе Python и WebKit. Но он был написан «для себя» и имел привязку к вещам, которые есть только в deb-дистрибутивах.

В 2011 году этот центр был адаптирован для, тогда еще, Mandriva и в репозиториях последней появилась первая реализация, которая впоследствии была переделана под ROSA.

Центр был источником запуска как утилит drakx, так и утилит настройки LXDE и дополнительного окружения (к примеру, openbox или NetworkManager). В дальнейшем форки центра появились в Mageia и в MagOS.

В этом году было решено поставить эксперимент и сделать рестайлинг Центра, а также заложить основы для будущего расширения последнего и избавления от старых утилит, доставшихся в наследство от Mandriva.



Что было сделано:

1. Во-первых, код был подвергнут рефакторингу и из него были выкинуты лишние и неиспользуемые части с целью упрощения работы с программой - оставлен только один вид, убраны настройки, путающие пользователя, почищен python-код.

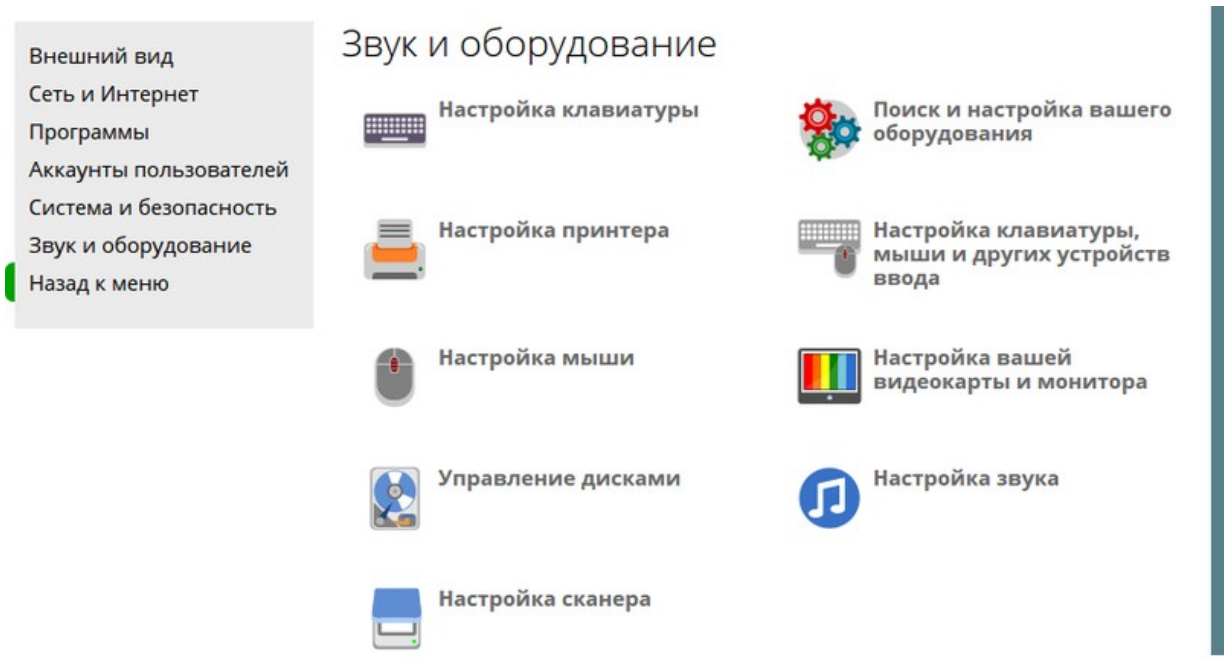
2. Веб-часть была переписана с использованием видоизменного bootstrap в стиле MetroUI с использованием наработок проекта http://metroui.org.ua/. Это дает нам более легкий дизайн, которому сейчас также следуют как Apple в новой iOS, так и Google, а также возможность построения единообразных веб-основанных приложений в будущем. Использование такого режима позволяет также быстро менять темы оформления.

3. Главная страница была видоизменена в с использованием «плиток», на которые в будущем планируется выводить полезную системную информацию, а также дать пользователю возможность настраивать внешний вид Центра (пока что пользователь может только переключать темы между светлой и темной).

4. Список утилит был приведен в соответствие с текущими положением дел в ROSA.

5. Все иконки были перерисованы в «плоском» дизайне. Полный набор иконок вы можете найти по этой ссылке: https://drive.google.com/folderview?id=0BxOLQQPjjQUrdEZwZHFKX1dmY28&usp=sharing

Что дальше? А дальше будет интереснее! Мы планируем переработку Центра и замену python на html+js части и, возможно, запуск в режиме как веб-кит окна, так и веб-сервиса, что позволит использовать его по сети.



Утилиты drakx и системные утилиты будут по возможности заменяться на аналоги на основе веб-технологий. Первой утилитой, над которой мы работаем сейчас, является замена system-config-printer на веб-аналог из CUPS, вызвываемый через localhost:631. Это даст большую гибкость в настройках системы. Также есть идея сделать в Центре определение DE и вывод тех инструментов, которые пристуствуют или необходимы для данной DE, а также сделать QT-версию основы Центра. Возможно, это позволит возродить старый добрый drakconf в новом виде, перейдя от парадигмы «утилиты в Центре управления окружения рабочего стола» к «Единый Центр, подстраивающийся под окружение рабочего стола». Но это всего лишь вероятное будущее, пока что Центр будет развиваться именно как часть LXDE.

Инфраструктура разработки

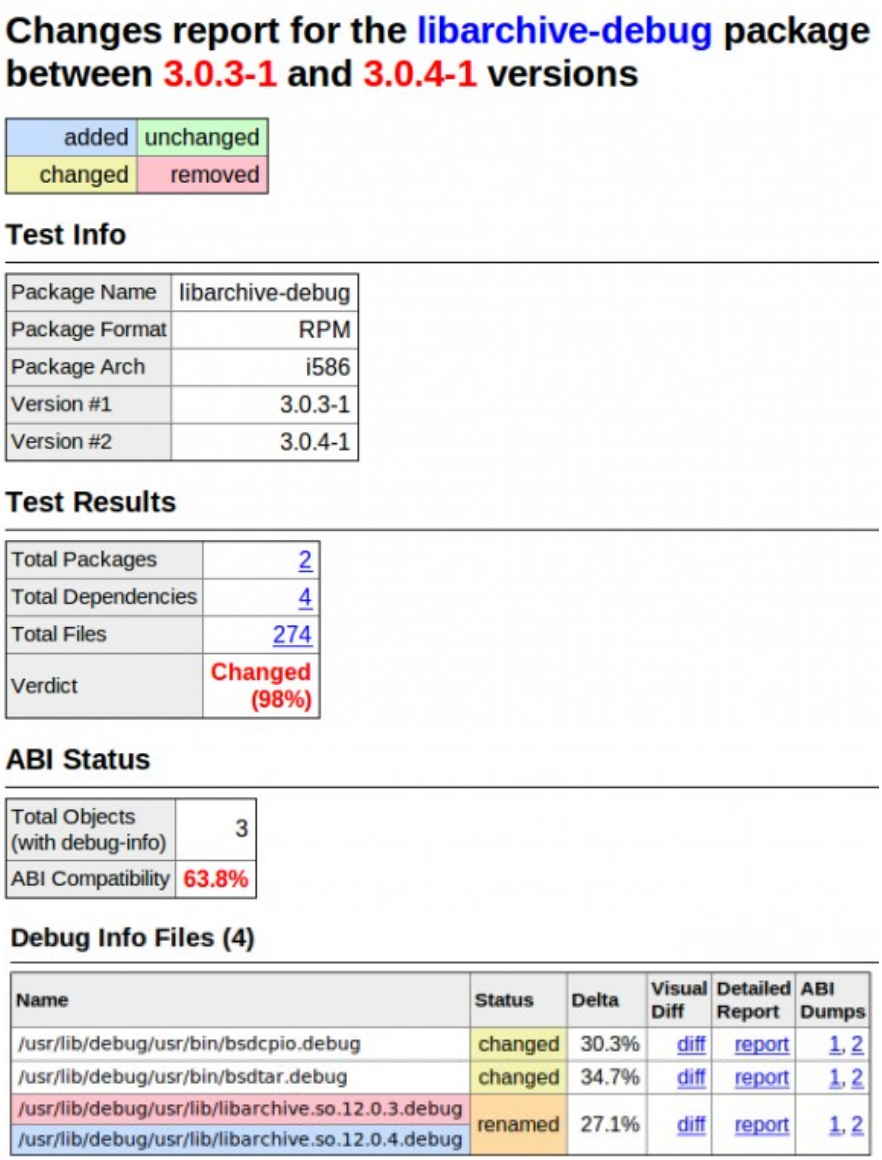
PkgDiff 1.6 - Добавлена проверка совместимости пакетов

Инструмент PkgDiff разработан для визуализации изменений в файлах и атрибутах пакетов любых форматов и предназначен для использования мэйнтейнерами и QA-инженерами с целью контроля изменений и предотвращения внесения незапланированных изменений в репозитории ПО, которые могут нарушить сборку или функционирование других пакетов. Одними из самых многочисленных элементов в структуре дистрибутива Linux являются системные библиотеки. Средний дистрибутив содержит несколько тысяч библиотек. При этом они имеют огромное число зависимостей между собой. По этой причине неаккуратное обновление одной библиотеки может нарушить сборку и функционирование других библиотек и в итоге привести к отказу пользовательских приложений.

В новую версию инструмента PkgDiff 1.6 мы добавили возможность проверки совместимости изменений в библиотеках. Это стало возможным благодаря новому инструменту ABI Dumper, который может извлекать информацию об ABI библиотеки из соответствующих debug-файлов, которая затем может быть проанализирована инструментом ABI Compliance Checker. Для проверки совместимости двух пакетов A и B (старой и новой версий одного пакета), пользователь должен подать на вход инструменту соответствующие debug-пакеты и запустить его с дополнительной опцией-details:

```
pkgdiff -old A-debuginfo.rpm -new B-debuginfo.rpm -details
```

В отчете добавлена секция ABI Status, в которой показан уровень обратной совместимости API и ABI библиотеки в процентах. Для просмотра детализированных отчетов о совместимости необходимо найти в отчете таблицу Debug Info Files и перейти по ссылкам в колонке Detailed Report.



Changes report for the libarchive-debug package between 3.0.3-1 and 3.0.4-1 versions

added	unchanged
changed	removed

Test Info

Package Name	libarchive-debug
Package Format	RPM
Package Arch	i586
Version #1	3.0.3-1
Version #2	3.0.4-1

Test Results

Total Packages	2
Total Dependencies	4
Total Files	274
Verdict	Changed (98%)

ABI Status

Total Objects (with debug-info)	3
ABI Compatibility	63.8%

Debug Info Files (4)

Name	Status	Delta	Visual Diff	Detailed Report	ABI Dumps
/usr/lib/debug/usr/bin/bsdcpio.debug	changed	30.3%	diff	report	1, 2
/usr/lib/debug/usr/bin/bsdtar.debug	changed	34.7%	diff	report	1, 2
/usr/lib/debug/usr/lib/libarchive.so.12.0.3.debug /usr/lib/debug/usr/lib/libarchive.so.12.0.4.debug	renamed	27.1%	diff	report	1, 2

ABI Dumper - Инструмент для создания дампа ABI из debug-информации ELF-объекта

При компиляции ELF-объектов, таких как разделяемые библиотеки, модули ядра Linux и др., с дополнительной опцией -g в них добавляется отладочная информация. Эта информация обычно используется стандартным отладчиком gdb для предоставления пользователю дополнительных возможностей при поиске ошибок во время выполнения программы. С помощью опции -debug-dump утилиты readelf или eu-readelf (из пакета elfutils) данная информация может быть представлена в удобном для прочтения виде.

Важной частью любого ELF-объекта является его бинарный интерфейс (ABI), предоставляемый использующим его приложениям. По сути, он является представлением API объекта на бинарном уровне (после компиляции). При обновлении ELF-объекта в

дистрибутиве важно поддерживать его ABI обратно совместимым, иначе это может привести к нарушению работы приложений. Изменения в ABI вызываются соответствующими изменениями в API объекта или изменением конфигурации сборки и опций компиляции. Чтобы отслеживать изменения в ABI объекта используется разработанный нами ранее инструмент ABI Compliance Checker. Но до настоящего времени он мог анализировать только разделяемые библиотеки посредством извлечения информации об ABI из заголовочных файлов.

Для того, чтобы расширить границы применения и упростить использование инструмента ABI Compliance Checker, мы разработали новый инструмент ABI Dumper для извлечения ABI-информации из отладочной информации объектов. Теперь, с помощью этого инструмента, можно отслеживать изменения в ABI не только библиотек, но и, например, модулей ядра. Типичный пример использования заключается в создании дампов ABI для старой и новой версии объекта:

```
abi-dumper libtest.so.0 -o ABIv0.dump
abi-dumper libtest.so.1 -o ABIv1.dump
```

и последующем их сравнении:

```
abi-compliance-checker -l libtest -old ABIv0.dump -new ABIv1.dump
```

К сожалению, у данного подхода есть свои недостатки. Пожалуй, главным недостатком является невозможность проведения некоторых проверок совместимости. Например, нет возможности проверки изменений значений констант (как дефайнов, так и глобальных данных), так как эти значения подставляются в код при компиляции и отсутствуют в отладочной информации. В целом, могут быть проведены около 98% всех проверок совместимости. Еще одним недостатком является долгое время работы на больших объектах (размером более 50 мб). Для сжатия debug-информации и, следовательно, уменьшения размеров входных объектов может быть использована утилита dwz.

Packaging-tools - набор полезных скриптов для облегчения задач мэйнтейнеров

В ROSA доступен packaging-tools - набор скриптов для мэйнтейнеров, изначально разработанный в Ark Linux.

Набор включает генератор spec-файлов для произвольных пакетов — vs, который создает заготовку spec-файла и открывает ее в vim (или в редакторе, заданном в переменных EDITOR или VISUAL). Также предоставляются специализированные генераторы spec-файлов для пакетов определенных типов:

- vl для библиотек
- vp для модулей Perl
- vj для Java-пакетов

Эти генераторы создают заготовки spec-файлов, учитывающие специфику конкретного типа пакетов (например, создаются необходимые подпакеты для библиотек, прописываются используемые в ROSA пути установки для Java и так далее).

Еще один полезный скрипт - это небольшая обертка для gendiff. Если вы хотите подготовить патч для некоторого пакета, то вам следует распаковать архив с исходным кодом и отредактировать необходимые файлы с её помощью. На самом деле, этот скрипт вызовет внешний редактор (указанный в переменных EDITOR или VISUAL; по умолчанию используется vim), сохранив перед этим исходный файл с суффиксом rosa2012.1~ (используемый суффикс можно переопределить с помощью опции -s). После внесения всех

необходимых модификаций в исходный код вызовите gendiff для создания патча.

Например, вот так можно подготовить патч для файла test.c из исходного кода someapp-1.2.3 с помощью редактора geany.

```
$ tar xzvf someapp-1.2.3.tar.xz
$ cd someapp-1.2.3
$ export EDITOR=geany
$ e test.c
$ cd ..
$ gendiff someapp-1.2.3 .rosa2012.1~ >my.patch
```

Этот способ может показаться немного сложным, но он действительно удобен, если вам необходимо подготовить небольшой патч для исходного кода большого объема.

Vtable Dumper - Инструмент для просмотра виртуальных таблиц библиотеки

Поддержка обратной совместимости внешних API интерфейсов является одной из наиболее важных задач при разработке библиотек. Это позволяет проще портировать существующие приложения на новые версии библиотек. Конечно, это относится в первую очередь к библиотекам, у которых уже есть применения, но и немаловажно для новых библиотек, поскольку при выборе библиотеки часто обращают внимание не только на функциональность, но и на стабильность. Совместимость бывает трех видов: на уровне исходных кодов, бинарная и функциональная. Наличие совместимости на уровне исходных кодов и функциональной совместимости позволяет портировать приложения путем их пересборки без необходимости модификации кода. Бинарная же совместимость является наиболее сильным уровнем совместимости и позволяет запускать старые приложения без их пересборки.

Наиболее сложно поддерживать совместимость C++ библиотек, так как, по сравнению с Си-библиотеками, дополнительно используются манглинг имен функций и виртуальные таблицы, создаваемые компилятором для реализации полиморфизма классов. Виртуальные таблицы могут меняться нелинейным образом при изменении в объявлении классов (добавление или удаление функций, базовых классов, атрибутов наследования и т.д.) и любое такое изменение может привести к неопределенному поведению или падению приложений. Для контроля за составом виртуальных таблиц мы создали специальный инструмент Vtable Dumper, который печатает содержимое всех виртуальных таблиц, найденных в бинарных файлах библиотеки. Таким образом можно сравнить вывод инструмента для старой и новой версий библиотеки, найти и предотвратить случайные изменения.

Пример использования инструмента:

```
vtable-dumper /usr/lib64/libstdc++.so.6
```

Стоит заметить, что существует альтернативный способ изучения состава виртуальных таблиц с помощью опции -fdump-class-hierarchy компилятора GNU G++. Однако входными данными для его работы являются заголовочные файлы библиотеки, и необходима их компиляция, которая может произойти с ошибками и потребовать дополнительных опций, порядка включения файлов и др. Кроме того, заголовочные файлы в отличие от бинарных файлов библиотеки не всегда установлены в системе. Форматы вывода обоих инструментов практически идентичны. Дополнительно инструмент Vtable-Dumper показывает параметры виртуальных функций в таблице и их mangled-имена (опция -mangled).

Далее приведен пример отображения инструментом содержимого виртуальной таблицы одного из классов библиотеки libQtGui:

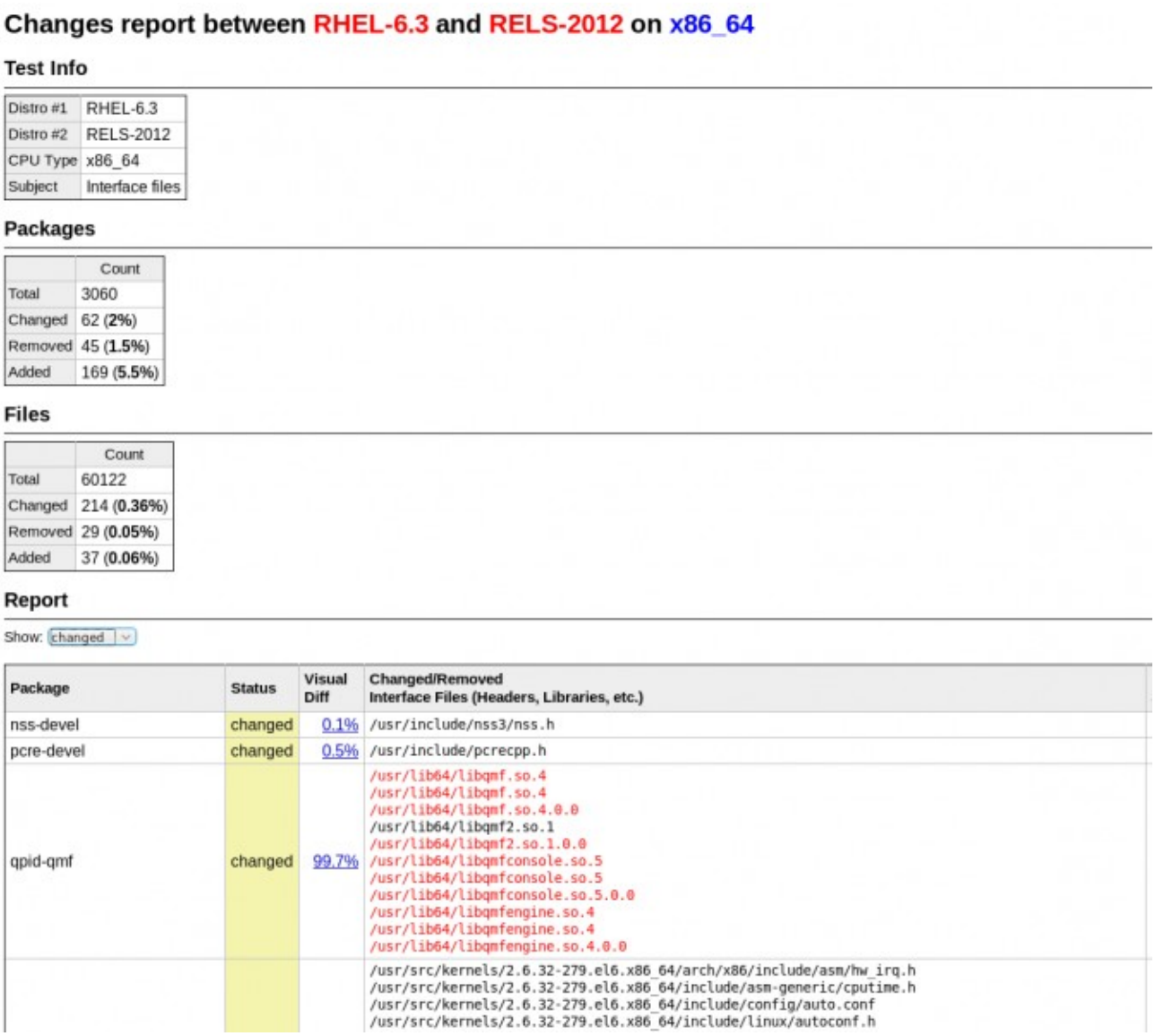
```
Vtable for QIconEnginePlugin
_ZTV17QIconEnginePlugin: 22 entries
```

```
  0     (int (*)(...)) 0
  8     (int (*)(...)) (& _ZTI17QIconEnginePlugin)
  16    (int (*)(...)) QIconEnginePlugin::metaObject() const
  24    (int (*)(...)) QIconEnginePlugin::qt_metacast(char const*)
  32    (int (*)(...)) QIconEnginePlugin::qt_metacall(QMetaObject::Call, int,
void**)
  40    (int (*)(...)) QIconEnginePlugin::~QIconEnginePlugin()
  48    (int (*)(...)) QIconEnginePlugin::~QIconEnginePlugin()
  56    (int (*)(...)) QObject::event(QEvent*)
  64    (int (*)(...)) QObject::eventFilter(QObject*, QEvent*)
  72    (int (*)(...)) QObject::timerEvent(QTimerEvent*)
  80    (int (*)(...)) QObject::childEvent(QChildEvent*)
  88    (int (*)(...)) QObject::customEvent(QEvent*)
  96    (int (*)(...)) QObject::connectNotify(char const*)
  104   (int (*)(...)) QObject::disconnectNotify(char const*)
  112   (int (*)(...)) __cxa_pure_virtual
  120   (int (*)(...)) __cxa_pure_virtual
  128   (int (*)(...)) -0x00000000000010
  136   (int (*)(...)) (& _ZTI17QIconEnginePlugin)
  144   (int (*)(...)) _ZThn16_N17QIconEnginePluginD1Ev
  152   (int (*)(...)) _ZThn16_N17QIconEnginePluginD0Ev
  160   (int (*)(...)) __cxa_pure_virtual
  168   (int (*)(...)) __cxa_pure_virtual
```

Визуализация изменений в дистрибутиве Linux с помощью утилиты DistDiff

Выпуск стабильных обновлений для дистрибутива Linux - непростая задача. Необходимо убедиться, что все пользовательские приложения продолжат корректно функционировать после обновления. Их можно разбить на две группы - базовые (из дистрибутива) и персональные (самостоятельно установленные пользователем из других источников). Базовые приложения можно относительно легко проверить, установив их в обновленный дистрибутив и вручную проверив их функциональность. О приложениях же второй группы разработчикам дистрибутива ничего не известно. Поэтому проверяют не только работоспособность приложений, но и детально изучают список всех изменений в пакетах.

Совместимость изменений в системных библиотеках можно проверять с помощью инструмента ABI Compliance Checker. Для проверки изменений в остальных пакетах мы разработали инструмент DistDiff. С помощью этого инструмента можно визуализировать изменения во всех пакетах дистрибутива и быстро их просмотреть на предмет нарушения совместимости. На вход этому инструменту надо лишь предоставить две директории - со старыми и новыми пакетами. В режиме «по умолчанию» инструмент проверяет изменения в интерфейсных файлах (библиотеки, модули, скрипты, исполняемые файлы и т.д.), потенциально влияющих на совместимость, но может проверять и все файлы с помощью опции -all-files.



Changes report between RHEL-6.3 and RELS-2012 on x86_64

Test Info

Distro #1	RHEL-6.3
Distro #2	RELS-2012
CPU Type	x86_64
Subject	Interface files

Packages

	Count
Total	3060
Changed	62 (**2%**)
Removed	45 (**1.5%**)
Added	169 (**5.5%**)

Files

	Count
Total	60122
Changed	214 (**0.36%**)
Removed	29 (**0.05%**)
Added	37 (**0.06%**)

Report

Show: changed

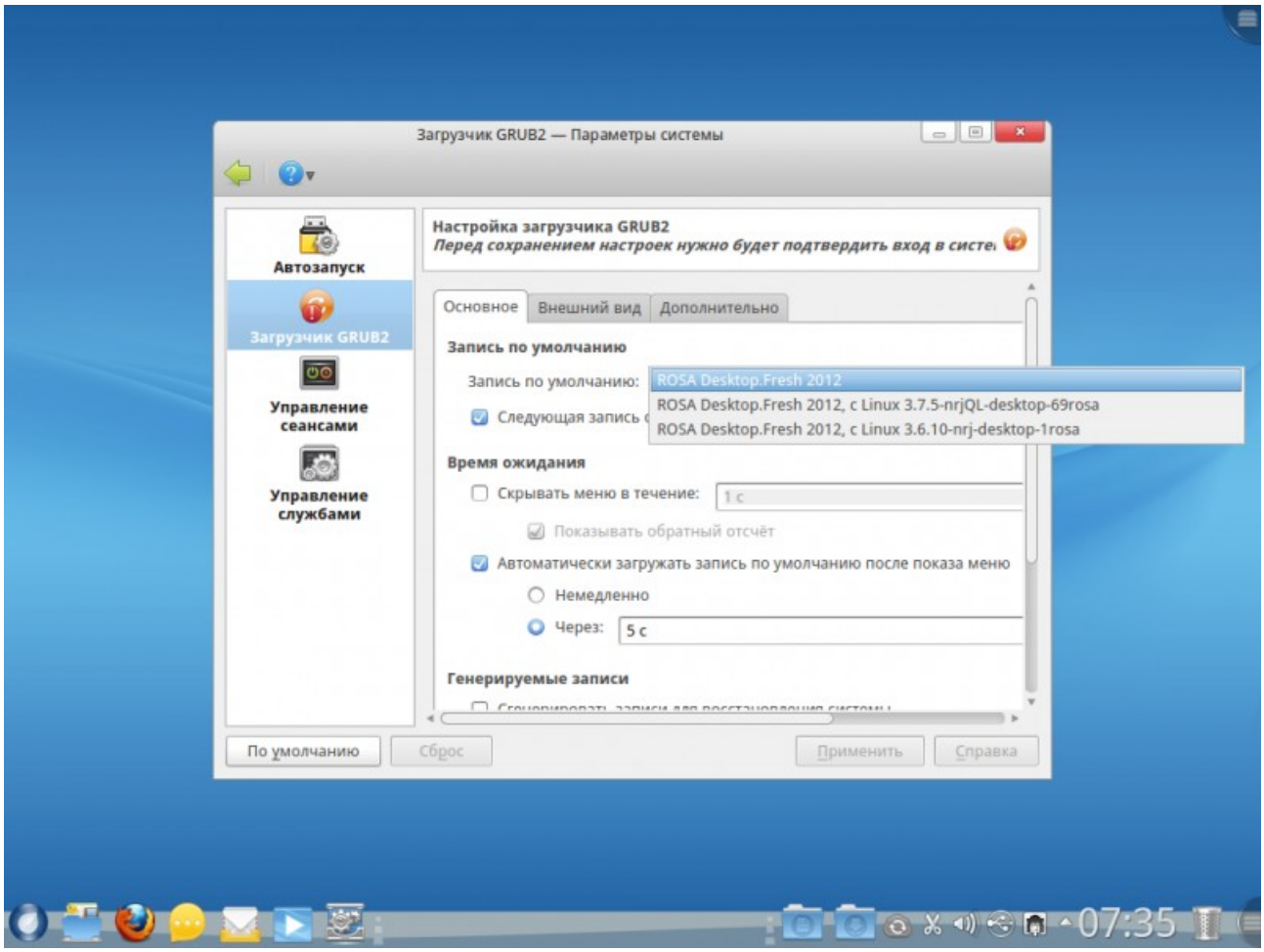
Package	Status	Visual Diff	Changed/Removed Interface Files (Headers, Libraries, etc.)
nss-devel	changed	0.1%	/usr/include/nss3/nss.h
pcre-devel	changed	0.5%	/usr/include/pcrecpp.h
qpid-qmf	changed	99.7%	/usr/lib64/libqmf.so.4 /usr/lib64/libqmf.so.4 /usr/lib64/libqmf.so.4.0.0 /usr/lib64/libqmf2.so.1 /usr/lib64/libqmf2.so.1.0.0 /usr/lib64/libqmfconsole.so.5 /usr/lib64/libqmfconsole.so.5 /usr/lib64/libqmfconsole.so.5.0.0 /usr/lib64/libqmfengine.so.4 /usr/lib64/libqmfengine.so.4 /usr/lib64/libqmfengine.so.4.0.0
			/usr/src/kernels/2.6.32-279.el6.x86_64/arch/x86/include/asm/hw_irq.h /usr/src/kernels/2.6.32-279.el6.x86_64/include/asm-generic/cputime.h /usr/src/kernels/2.6.32-279.el6.x86_64/include/config/auto.conf /usr/src/kernels/2.6.32-279.el6.x86_64/include/linux/autoconf.h

Работает инструмент на основе разработанного нами ранее инструмента PkgDiff для сравнения и визуализации изменений в пакетах.

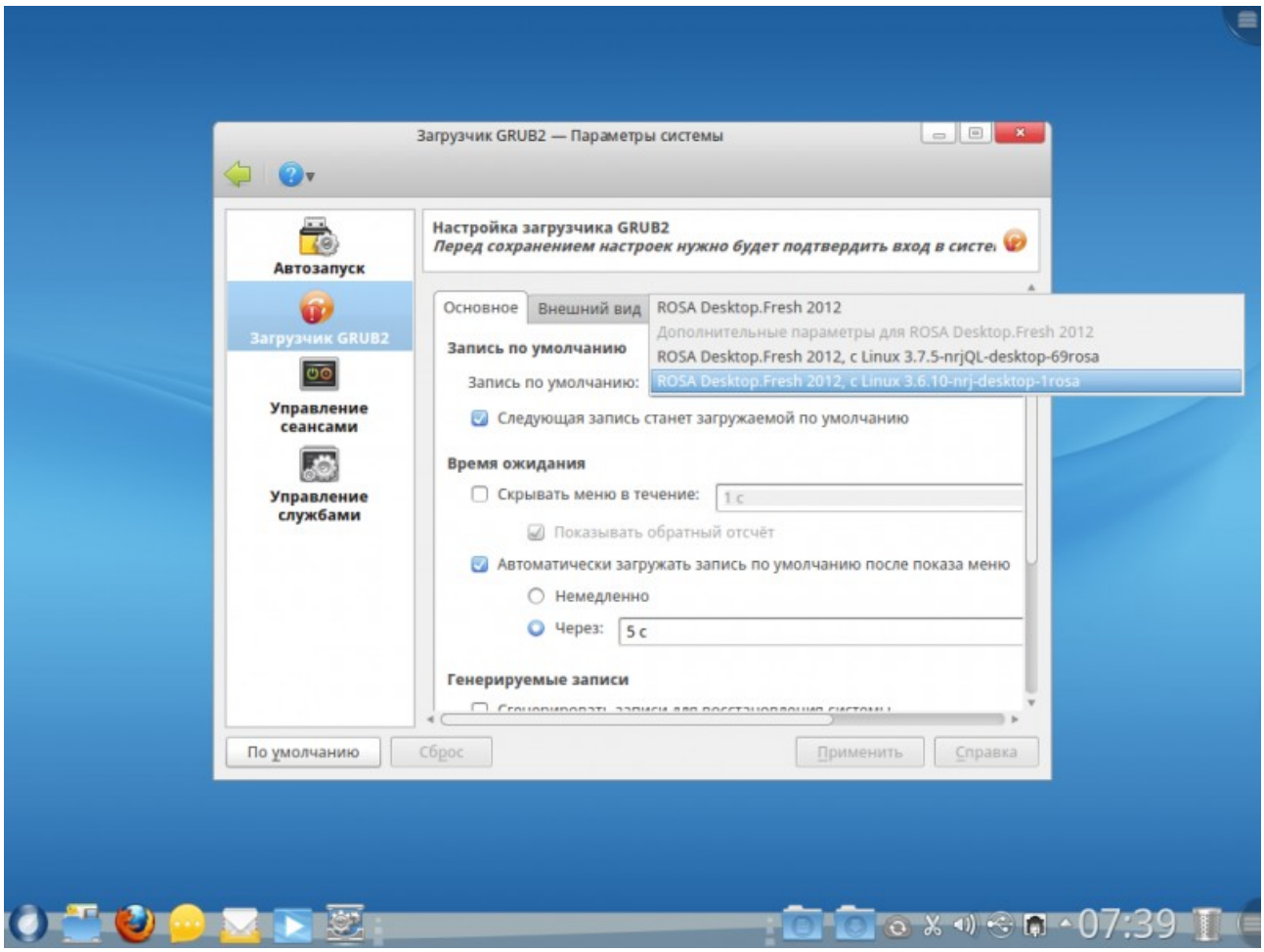
Upstream

Вклад в апстрим утилиты kcm-grub2

Нашими разработчиками был сделан патч для настройщика GRUB2 - kcm-grub2. Патч исправляет ошибку, тянущуюся с момента выхода GRUB2 версии 2.00.
Начиная с версии 2.00, изменилась структура меню загрузчика - появились вложенные меню. В связи с этим стало невозможно с помощью настройщика GRUB2 выбрать пункт по умолчанию, когда этот пункт находится во вложенном меню.



После применения патча эта опция заработала. Также список пунктов загрузки в настройщике теперь строится с учётом вложенных меню.



Патч был отослан в апстрим настройщика и будет обязательно встроен в следующую версию.

Авторы выпуска

Денис Силаков
Антон Чернышов
Александр Казанцев
Владимир Тестов
Андрей Пономаренко
Фомичев Петр