ЛЕКЦИЯ 9. ПАКЕТЫ, ЗАВИСИМОСТИ, БИБЛИОТЕКИ В LINUX MINT

9.1. Пакеты

Пакеты — это своего рода программные кванты, на которые делится система или дистрибутив. Это могут быть и простые монофункциональные утилиты (например, строчный текстовый редактор ed или архиватор tar), более или менее обширные наборы функционально связанных программ (скажем, coreutils) или составные части огромных программных комплексов (примером чему — Cinnamon, о котором столько гворилось в прошлых очерках).

Термин пакет (английское *package*) употребляется в двух смыслах: как авторский набор исходных текстов, созданный разработчиком программы, и как комплект скомпилированных из него исполняемых программ и всех их служебных файлов, собранный майнтайнерами дистрибутива или вообще третьими лицами. Пакеты в первом смысле называются исходниками или вообще «сырцами» (от ангийского *Source*), во втором — бинарниками.

Бинарные пакеты специфичны для семейств родственных дистрибутивов, почему часто говорят о системах rpm based или deb based. Но даже если они собраны в одном формате (например, rpm или deb), бинарные пакеты из разных дистрибутивов далеко не всегда совместимы в рамках одной системы. Впрочем, к формату deb, принятому в дистрибутиве Mint и потому в интересующему нас больше всех остальных, вместе взятых, это относится в наименьшей степени: его пакеты сохраняют почти полную бинарную совместимость с пакетами родительской Ubuntu и частичную — с пакетами прародительского Debian'а.

Ключевым для бинарных, или дистрибутивных, пакетов является понятие зависимостей. Суть его в том, что пакет packagename1 для сборки, установки и (или) функционирования требует наличия в системе пакета packagename2, тот, в свою очередь, может потребовать пакета packagename3, и так далее.

Выделяются разные типы зависимостей. С одной стороны, различают зависимости при сборке (обычно называемые просто зависимостями — *depends*) и зависимости при запуске (по английски именуемые *run depends*).

Как следует из названий, теоретически зависимости первого рода необходимы только на стадии сборки пакета, тогда как зависимости второго рода действуют постоянно. На практике же в большинстве случаев *depends* и *run depends* если и не эквивалентны, то значительно перекрывают друг друга.

С другой стороны, следует различать зависимости жёсткие и «мягкие». Удовлетворение первых абсолютно необходимо для сборки и функционирования данного пакета. «Мягкие» зависимости данного пакета не критичны для его функционирования — удовлетворение их лишь добавляет ему дополнительные функции (например, печати и сканирования для офисных и графических приложений) или возможности (скажем, доступ к файлам данных определённых форматов для той же графики или мультимедиа).

Понятие зависимостей пронизывает насквозь UNIX-совместимые системы, и особенно важно для свободных их представителей. В то же время пользователи Windows с ним сталкиваются очень редко, и потому постижение его вызывает у них определённые трудности.

Традиционная модель разработки UNIX-программ (то, что задумчиво именуют UNIX Way) характеризуется ярко выраженным стремлением не множить сущности без крайней необходимости. Или, говоря попросту, не изобретать велосипеды. То есть: если требуемая разработчику данной программы функция уже реализована и включена в какую-либо распространенную библиотеку, то разработчик скорее всего этой библиотекой и воспользуется, а не будет переписывать ее с нуля. Благо, поскольку все

распространённые и общеупотребимые библиотеки открыты, он имеет полную возможность это сделать.

Ведь все программы, вне зависимости от их назначения, неизбежно должны выполнять некоторые однотипные действия, как то: открыть файл, записать его, вывести на экран его содержимое, и так далее, вплоть до закрытия. Сущность таких действий не меняется, что бы программа ни делала. И потому нет никакого смысла программировать такие манипуляции каждый раз заново.

Вот их, как правило, и не программируют «с нуля», а объединяют соответствующие команды в отдельные программные комплексы, именуемые библиотеками (libraries). Сами по себе они к автономному исполнению не пригодны. Однако любая программа, при необходимости совершить одно из типовых действий, вызывает из такой библиотеки некий фрагмент кода, содержащий требуемую последовательность команд.

9.2. Репозитоии

Пакеты, входящие в дистрибутив, организованы в репозитории. В переводе на русский язык слово репозиторий означает хранилище. Сам по себе репозиторий действительно можно в первом приближении определить как место хранения пакетов, специально собранных для данного дистрибутива, к которому возможен свободный (мы ведь ведём речь только о свободных системах) доступ.

Пакеты в репозитории должны быть структурированы по определённым, присущим данному дистрибутиву, принципам. Система хранения пакетов должна обеспечивать их пополнение, обновление, а главное — поддержку целостности и непротиворечивости пакетов в отношении зависимостей, причём для всех поддерживаемых на текущий момент версий дистрибутива. Иными словами, пакеты в репозитории должны сопровождаться базами данных — теми самыми, которые используются системой управления пакетами данного дистрибутива.

Кроме того, весьма желательно, чтобы репозиторий зеркалировался на нескольких независимых серверах — по вполне понятным причинам. Правда, это не является непременным требованием. Тем не менее, наличие зеркал — одно из оснований для употребления слова репозитории во множественном числе.

Официальные репозитории Ubuntu располагаются по адресу: archive.ubuntu.com/ubuntu. Это — «головное» хранилище пакетов, имеющее многочисленные региональные зеркала, принадлежность которых к стране указывается стандартным двухсимвольным префиксом, например ru.archive.ubuntu.com/ubuntu/ — российское зеркало.

Репозитории Mint организованы внешне сходно с таковыми Ubuntu, но на самом деле строятся по несколько иным принципам. В файле official-package-repositories.list они описываются двумя строками:

deb http://linux-mint.froonix.org rebecca main upstream import

deb http://extra.linuxmint.com rebecca main

Первая — определяет основной репозиторий для пакетов дистрибутива, распространяемых свободно. Как и в последних, в ней указывается URL подключённого по умолчанию или выбранного позднее сервера, а затем имя релиза (на текущий момент — rebecca). Далее следует список категорий, однако тут в это понятие вкладывается несколько иной смысл. Так, категория main включает в себя дистрибутив-специфичные пакеты, «фирменные» утилиты, MDA, Cinnamon, MATE. В категорию upstream входят пакеты, заимствованные из GNOME 3 и специально пересобранные для совместимости с Mint. Здесь же можно обнаружить пакеты для его Xfce-редакции. Категория же import образована пакетами для KDE-редакции, представленными во всей их полноте.

9.2.1. PPA-репозитории

Кроме официального репозитория, для Ubuntu существует централизованное хранилище репозиториев дополнительных, объединяемых понятием PPA — Personal Packages Archive, то есть входящих в персональный архив пакетов, пополняемый сторонними разработчиками и майнтайнерами. А их, вследствие популярности дистрибутива, очень немало. И поэтому свежие версии многих программ, как популярных (что важно для начинающих), так и весьма экзотических (что часто критично для многоопытных), в первую очередь появляются как бинарники в так называемых PPA-репозиториях Ubuntu.

Для доступа к PPA-репозиториям фирмой Canonical разработан специальный онлайновый инструмент — Launchpad, размещённый на одноимённом сайте. Это — не открытая и не свободная система. Более того, она имеет и платную версию, предназначенную для коммерческих пакетов.

Большинство пакетов в PPA-репозиториях собирается и поддерживается майнтайнерами-индивидуалами, и потому здесь нередко можно видеть их имена, фамилии или ники, например, ppa:andrew-crew-kuznetsov/crew — репозиторий, поддерживаемый Андреем Crew Кузнецовым, разработчиком программы XNeur. В других случаях это просто имя пакета, часто с отражением статуса разработки, например, ppa:marlin-devs/marlin-daily — репозиторий «ежедневных» сборок файлового менеджера Marlin. Репозиторий может включать несколько связанных друг с другом пакетов — и тогда называться по главному из них, например: ppa:zfs-native/stable и ppa:zfs-native/daily — репозитории пакетов поддержки ZFS on Linux стабильной и разрабатываемой ветки, соответственно.

Возможны и более причудливые имена, например, ppa:mystic-mirage/komodo-edit — репозиторий текстового редактора Komodo Edit. Важно, что они в обязательном порядке включают «префикс» ppa:, который в имени соответствующего list-файла отбрасывается. Зато завершается последний обязательным компонентом — именем релиза. Например, для Komodo Edit имя list-файла — mystic-mirage-komodo-edit-trusty.list.

Внутри такого файла — обычно две строки. Например, для пакета komodo-edit они будут такими:

```
deb http://ppa.launchpad.net/mystic-mirage/komodo-edit/ubuntu trusty main
deb-src http://ppa.launchpad.net/mystic-mirage/komodo-edit/ubuntu trusty main
```

То есть в одном файле описывается и репозиторий бинарников, и репозиторий исходников. Если последний отсутствует — соответствующей строки не будет. Впрочем, в PPA-репозиториях пакетов без исходников не водится. А вот среди «не вполне свободного» софта встречаются, примером чему — браузер Opera: файл opera-stable.list выглядит следующим образом:

```
deb http://deb.opera.com/opera-stable/ stable non-free #Opera Browser (final releases)
```

Однако случаи, когда приходится искать пакеты за пределами PPA-репозиториев, очень редки.

9.3. Установка пакетов

Основным средством управления пакетами во всех дистрибутивах deb based в настоящее время являются утилиты семейства Apt. Ранее наряду с ними широко применялась утилита aptitude, используемая в некоторых дистрибутивах и по сей день (а также обычно устанавливаемая по умолчанию).

Субкоманды утилиты apt разделяются на две группы, служащие различным целям — получению информации о пакетах, для которых достаточно прав обычного пользователя, и манипулированию ими, что требует прав администратора. Однако обратим внимание на

второй пример использования субкоманды help. Он показывает, что утилита apt позволяет не утруждать себя воспоминаниями о необходимости ввода команды sudo: при указании любой её субкоманды, связанной с манипулированием пакетами и потому требующей прав администратора, запрос пароля последует автоматически:

```
$ apt install geany
[sudo] password for alv:
```

Первая операция, с которой сталкивается пользователь при работе с пакетами — поиск нужного пакета. Этой цели служит субкоманда search с указанием в качестве аргумента ключевого слова, которым может быть имя пакета или его фрагмент, а также слово из краткого описания пакета. Выводом будет список подходящим пакетов, сопровождаемый тем самым кратким описанием. Например, для пакета zsh это выглядит так:

```
$ apt search zsh
…
p   fizsh                  - Friendly Interactive ZSHell
i   zsh                    - командная оболочка с большим набором возмо
p   zsh:i386               - командная оболочка с большим набором возмо
…
i   zsh-doc                - Документация к zsh - формат info/HTML
i   zsh-lovers             - Полезные советы и примеры для zsh
p   zsh-static             - shell with lots of features (static link)
p   zsh-static:i386        - shell with lots of features (static link)
p   zshdb                  - отладчик сценариев оболочки Zsh
```

В выводе, кроме всего прочего, содержится информация о состоянии пакета: i — установленный, p — не установленный или «чисто» удалённый, c — удалённый с сохранением конфигов.

Поиск пакетов происходит не в мировом масштабе, а только в подключённых репозиториях, как официальных, так и дополнительных. Причём иногда он может вывести пакеты если не с одинаковыми, то с похожими именами из разных источников. Так, например, в системе может быть установлена Opera 26.0 из репозитория разработчиков. И этот же пакет, версии 12.16, может имеется в extra-репозитории Mint. И потому вывод команды

```
$ apt search opera
```

будет таким:

```
p   opera                  - Fast and secure web browser and Internet s
p   opera:i386             - Fast and secure web browser and Internet s
c   opera-beta             - Fast and secure web browser
p   opera-developer        - Fast and secure web browser
p   opera-next             - Fast and secure web browser and Internet s
p   opera-next:i386        - Fast and secure web browser and Internet s
i   opera-stable           - Fast and secure web browser
```

Как тут определить, кто есть who? В данном случае (в том числе и) этой цели послужит субкоманда show: с именем пакета в аргументе она выведет подробные сведения о пакете. Для просто opera они будут выглядеть так:

```
$ apt show opera

Пакет: opera
Состояние: не установлен
Версия: 12.16.1860-1linuxmint
Приоритет: необязательный
Раздел: non-free/web
…
```

А для opera-stable — иначе:

```
$ apt show opera-stable

Новый: да
Состояние: установлен
Автоматически установлен: нет
Версия: 26.0.1656.60
Приоритет: необязательный
Раздел: non-free/web
…
```

В выводе субкоманды show имеется и другая информация — о жёстких и некоторых прочих зависимостях, конфликтующих пакетах, и так далее, а также описание пакета, более подробное, чем резюме. Важно, что вся она доступна как для установленных пакетов, так и для пакетов, имеющихся в репозиториях, но не установленных.

После того как нужный пакет был обнаружен с помощью субкоманды search, и его нужность была подтверждена просмотром вывода субкоманды show, его остаётся только установить — для этого предназначена субкоманда install, воспинимающая в качестве аргемента имя пакета, например:

```
$ apt install pepperflashplugin-nonfree
```

После чего, если пакет не связан никакими зависимостями, установка его начнётся немедленно. В противном же случае будет выведен список всех пакетов, которые он тянет за собой, с указанием объёма, и последует запрос, продолжать ли это дело. Ответ по умолчанию — Да, то есть при согласии достаточно просто нажать Enter. Однако, прежде чем это делать, со списком зависимостей желательно всё-таки ознакомиться.

По умолчанию apt install устанавливает только жёсткие зависимости, зависимости рекомендуемые и предлагаемые будут просто перечисленны.

Субкоманда install предназначена для установки пакетов из репозиториев, как удалённых, так и локальных. Однако локально пакеты могут быть установлены и другим способом — субкомандой deb. В отличие от субкоманды install, она в качестве аргумента требует не имени пакета, а полного имени его файла, при необходимости, с указанием пути. Например, так

```
$ apt deb path2/Yandex.deb
```

будет установлена бета-версия яндексового браузера для Linux. Разумеется, файл Yandex.deb предварительно должен быть скачан. Как любезно сообщит нам команда

```
$ apt help deb
```

В случае, если в системе установлена более старая версия запрашиваемого пакета,

обе субкоманды, и install, и deb, выполнят его обновление. А вот установить повторно ту же версию, что имеется, первая субкоманда откажется.

Пакеты требуется не только устанавливать и обновлять, но иногда и удалять. Для этого предназначены две субкоманды — remove и purge, аргументами которых, очевидно, служат имена пакетов, подлежащих изничтожению. Разница между ними в том, что первая удаляет только основные файлы пакета, не затрагивая его общесистемных конфигов, вторая же удаляет и их.

При использовании обеих субкоманд удаления следует внимательно читать их вывод, прежде чем нажимать Enter для подтверждения серъёзности своих намерений: обе они автоматически удаляют пакеты, которые зависят от удаляемого, что далеко не всегда можно приветствовать.

А вот чего ни remove, ни purge не удаляют автоматически — так это пакеты, от которых зависит удаляемый, даже если они уже больше никем не используются.

Перед установкой пакетов из репозитория они предварительно скачиваются и помещаются в каталог /var/cache/apt/archives/, где со временем их скапливается преизрядное количество, и далеко не всегда эти пакеты потребуются в дальнейшем (точнее, почти всегда не потребуются). Для избавления от таких «отходов производства» существуют субкоманды autoclean и clean. Первая удаляет из кеша только пакеты устаревших версий, сохраняя те, версии которых установлены в системе. Вторая же полностью очищает каталог /var/cache/apt/archives/.

Сказанное выше касалось единичных пакетов или их серий — любая из перечисленных субкоманд принимает любое количество аргументов. Однако в утилите apt предусмотрены и субкоманды для тотального обновления системы. Однако, прежде чем выполнить любую из них, необходимо провести обновление локального кеша пакетов, то есть получить списки новых и обновлённых пакетов. Делается это субкомандой update:

```
$ apt update
```

Её же в обязательном порядке следует выполнять после каждого изменения в репозиториях — подключения новых или отключения имевшихся.

Для тотального обновления существует субкоманда upgrade, которая выявит все пакеты, для которых в репозиториях доступны более свежие версии, выведет их список, объём для скачивания и прирост объёма занятого дискового пространства после выполнения процедуры, а также запросит подтвержения:

```
$ apt upgrade
```

В ходе выполнения upgrade обновляются по возможности все пакеты, за исключением тех, для которых обновление потребует, для разрешения зависимостей, доустановки новых пакетов или удаления существующих; для них сохраняются текущие версии. Так что это не совсем тотальное обновление системы.

9.4. Управление пакетами Synaptic

Система управления пакетами Synaptic — графический фронт-энд для утилит семейства apt, обычно используемыми для работы с пакетами deb-формата, а в некоторых дистрибутивах — и с пакетами rpm.

В их числе:

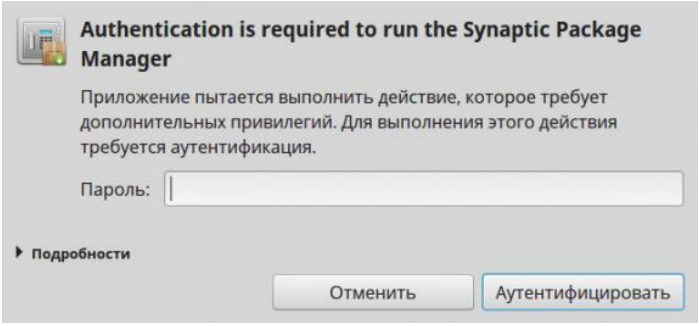
- поиск пакетов в репозиториях с определением их состояния и статуса;
- их установку и обновление с автоматическим разрешением зависимостей;
- удаление пакетов, в том числе и включая их зависимости;

- обновление базы данных пакетов из репозитория;
- тотальное обновление системы.

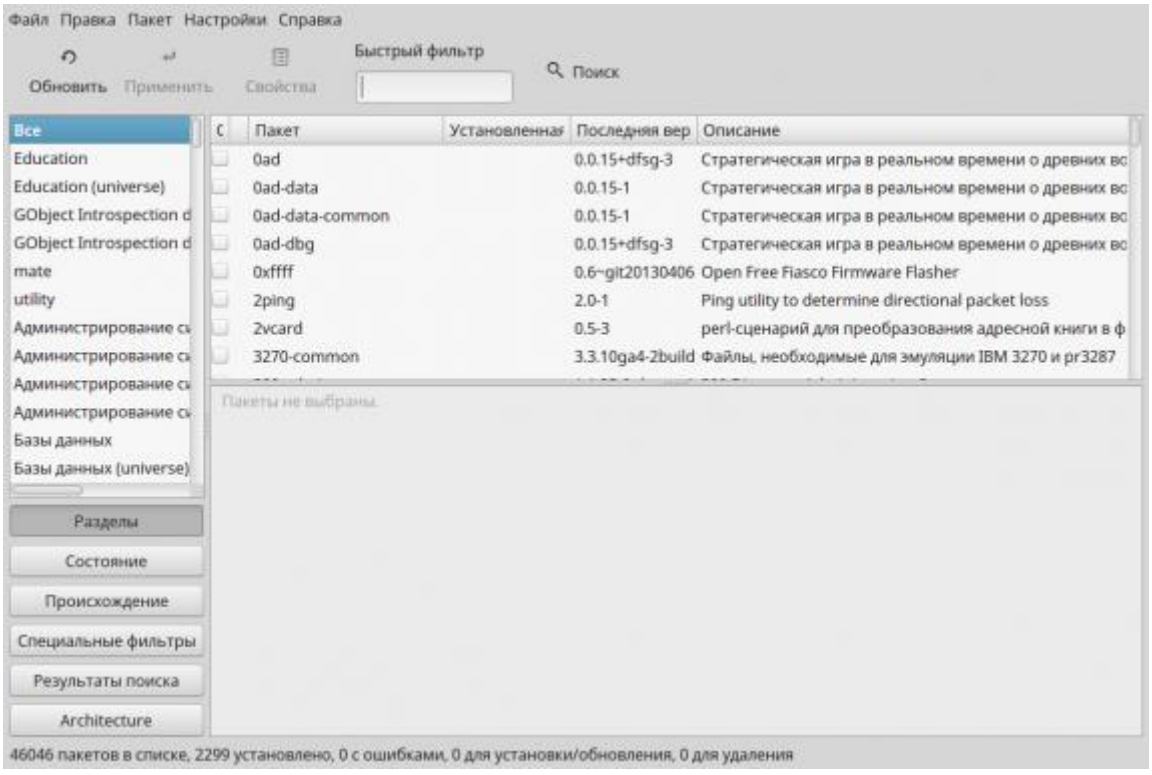
Кроме того, Synaptic включает средства настройки — в частности, доступа к репозиториям. В Mint для этой цели вызывается собственная утилита smintsource.

Запуск Synaptic'а выполняется через главное меню панели приложений (Администрирование -> Менеджер пакетов Synaptic) или любым другим традиционным для Mint способом.

Очевидно, что установка и удаление пакетов потребует прав администратора, запрос на получение каковых (посредством механизма sudo, то есть с вводом пользовательского пароля) и последует после вызова Synaptic'а через меню:



И после ввода пароля пользователя появляется окно примерно такого вида:

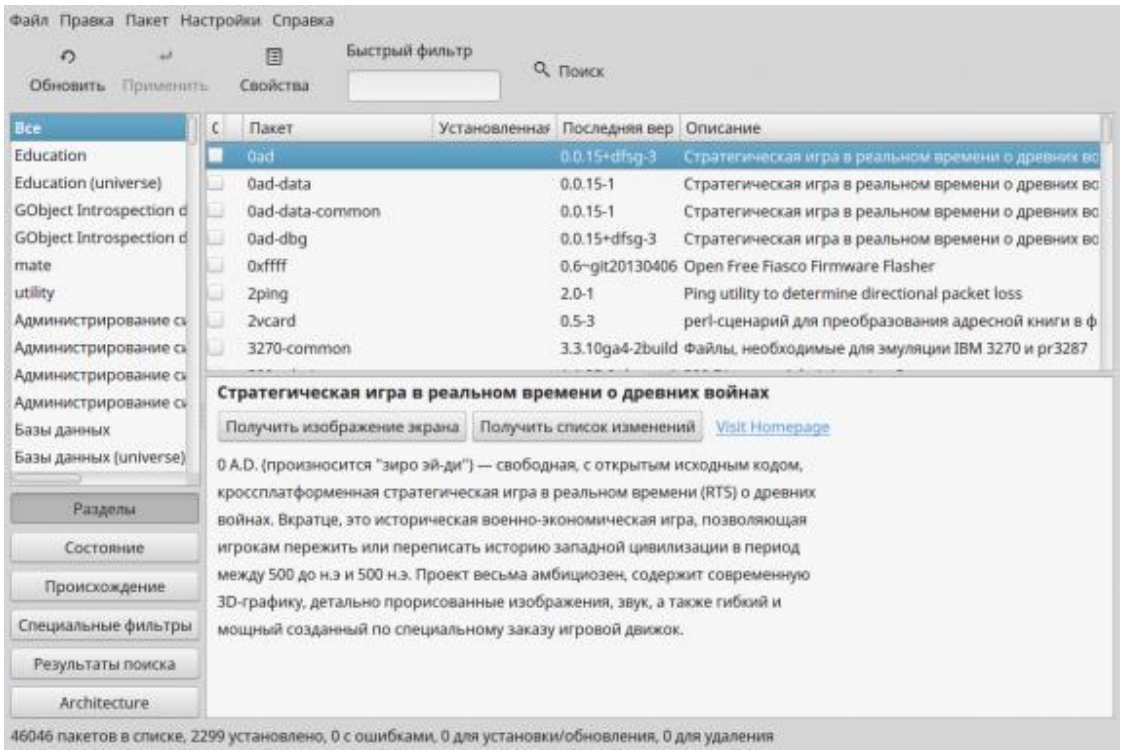


Как явствует из скриншота, в окне Synaptic'а мы имеем следующие основные элементы интерфейса:

- строку меню;
- панель инструментальных кнопок;
- два главных фрейма — список разделов репозитория и список пакетов выбранного раздела (по умолчанию показываются все пакеты);
- фрейм с кнопками выбора критериев для вывода пакетов;

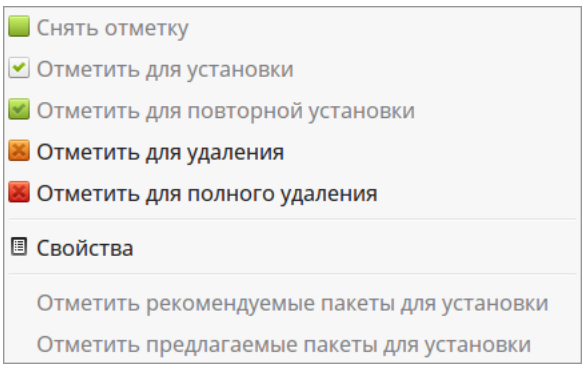
- фрейм свойств конкретного пакета.

Последний фрейм пуст, если в правом главном фрейме не отмечен ни один пакет, как на предыдущем скриншоте. Но заполняется контентом при свершении выбора: в правом нижнем фрейме мы увидим описание пакета (если доступно, на русском)

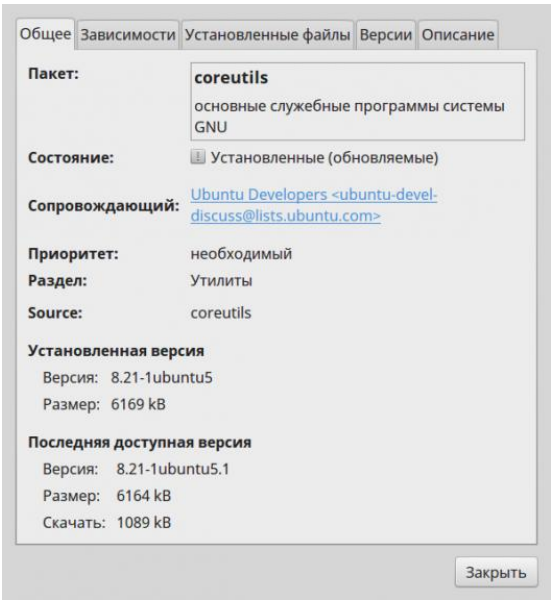


Если при этом нажать на кнопку Получить изображение экрана — то появится скриншот соответствующего пакета (буде таковой существует и имеет смысл):

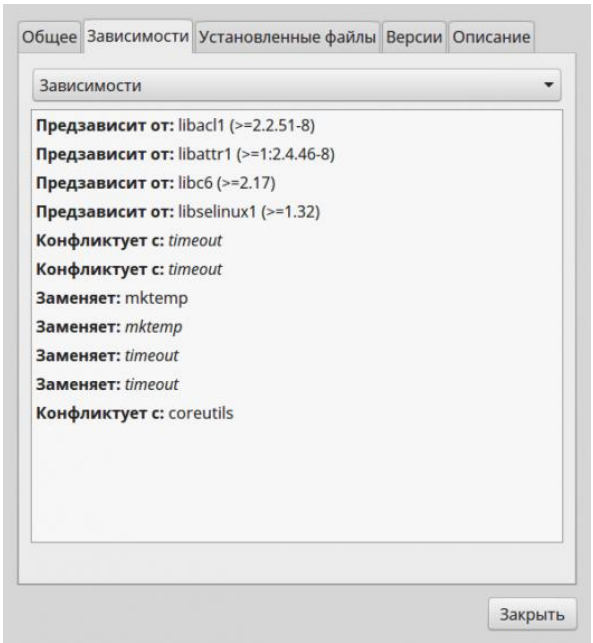
А при правом клике на имени пакета появляется контекстное меню:



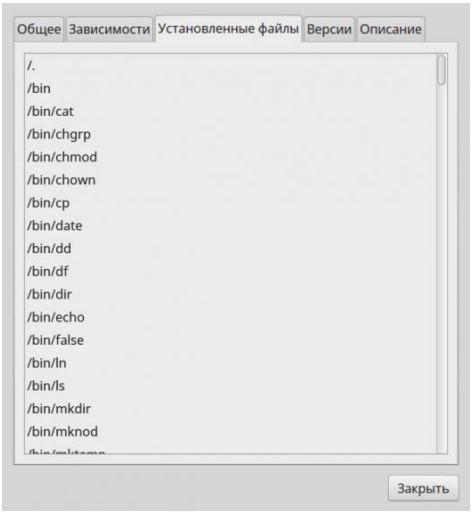
Здесь-то, в пункте Свойства, и содержится главная информация о пакете — общие сведения:



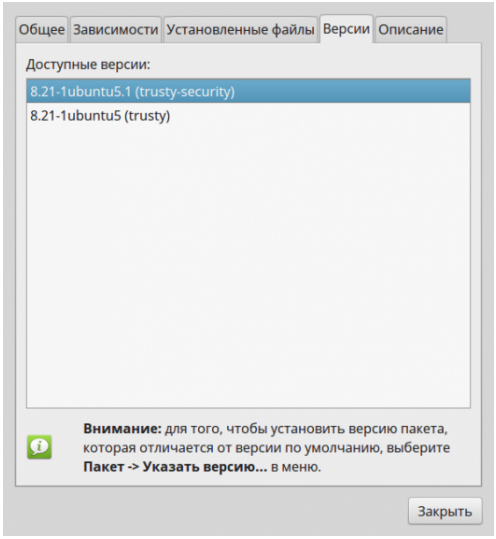
перечень зависимостей:



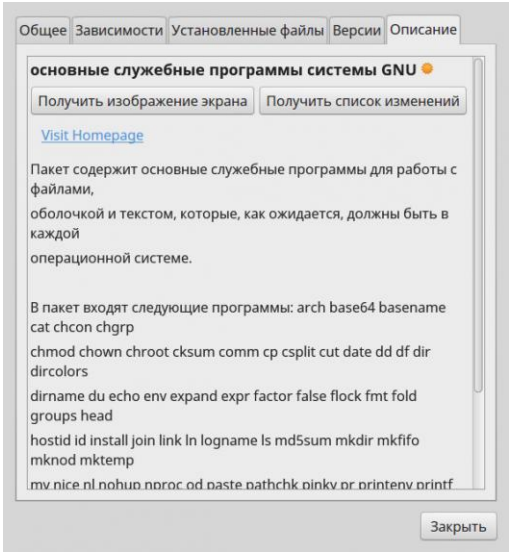
список установленных файлов и путей к ним (доступно только для установленных пакетов):



доступные версии

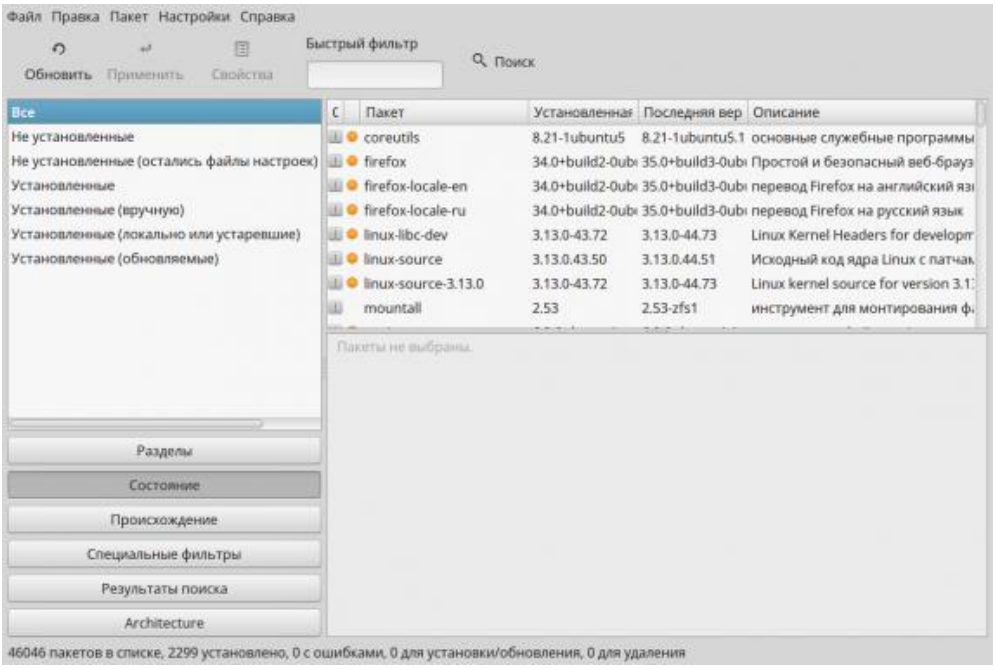


и, наконец, описание пакета

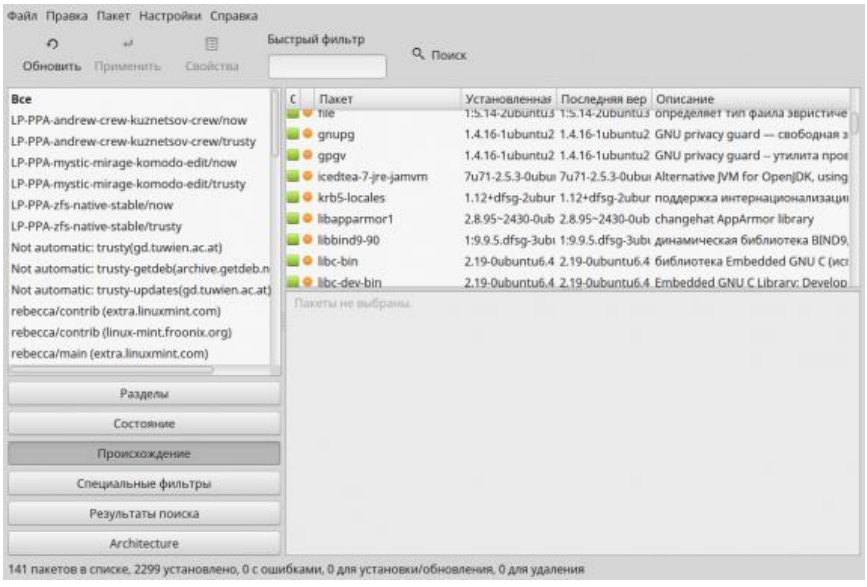


Теперь рассмотрим критерии вывода пакетов. С группировкой пакетов по разделам всё более-менее ясно, тем более, что названия разделов почти все даны в русском переводе, а те немногие, что оставлены в оригинале (например, World Wide Web), и без перевода понятны. Следующий критерий отбора — по состоянию пакетов. После нажатия соответствующей кнопки в левом главном фрейме выводятся следующие категории:

- Все;
- Не установленные;
- Не установленные (остались файлы настроек);
- Установленные;
- Установленные (вручную);
- Установленные (локально или устаревшие);
- Установленные (обновляемые).

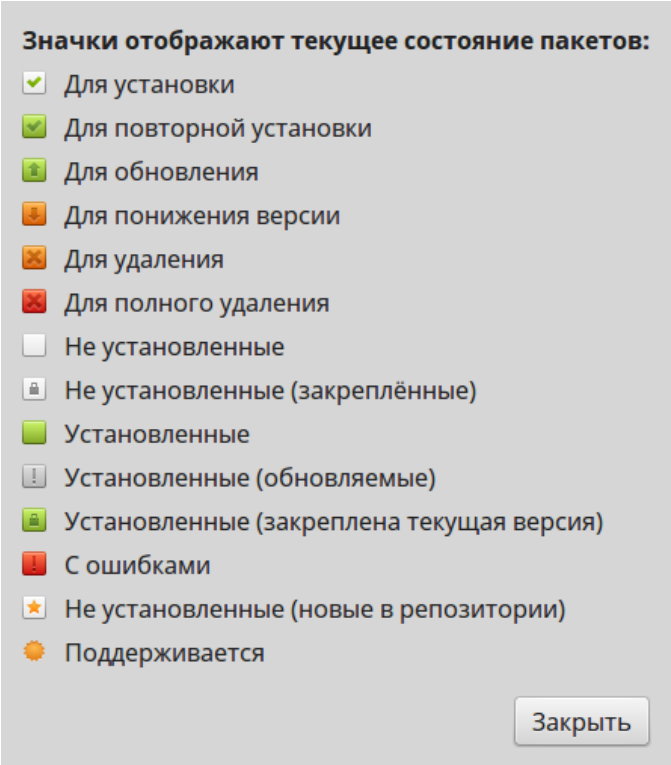


По происхождению пакеты разделяются на установленные локально (то есть с инсталляционного носителя или предварительно скачанные на диск) и из различных репозиториев — rebecca, trusty, PPA и так далее:



Обратимся к спискам файлов, выводимых в правом главном фрейме. Если поглядеть на него внимательно, то слева можно увидеть две колонки иконок, причём вторая может либо изображать нечто жёлтенькое, либо быть пустой. Факт наличия жёлтой иконки указывает, что данный пакет поддерживается официально разработчиками дистрибутива. А отсутствие пиктограммы во второй колонке говорит о том, что пакет либо поддерживается сообществом (точнее, некими конкретными его представителями), либо, в рамках дистрибутива, не поддерживается вообще.

Пиктограммы же первой колонки отражают статус пакет: установленный (зелёный квадратик), не установленный (квадратик не залитый), и так далее. Полную расшифровку значений пиктограмм можно получить через систему встроенной помощи: меню Справка -> Описание значков:



А теперь вернёмся к контекстному меню. Из приведённого скриншота можно видеть, что для установленного пакета активизированы пункты:

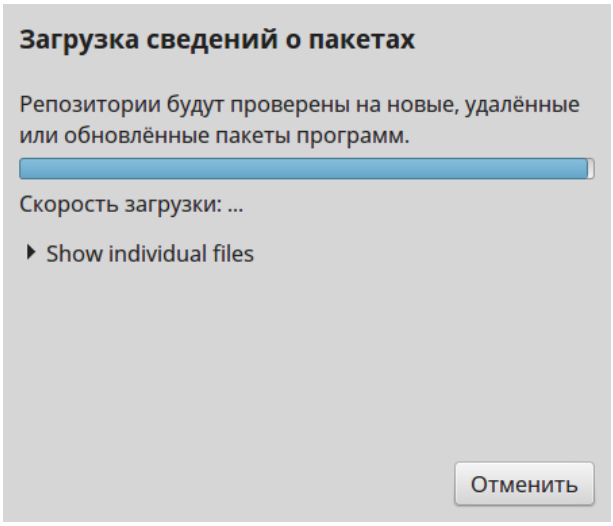
- Отметить для повторной установки — то есть реинсталляции, аналог команды apt reinstall;
- Отметить для удаления — удаление данного пакета, без конфигурационных файлов, аналог команды apt remove;
- Отметить для полного удаления — удаление данного пакета вместе с его конфигами, но не затрагивая зависимостей, аналог команды apt purge;
- свойства — его мы уже рассмотрели.

Для пакета не установленного по умолчанию доступны два пункта — Отметить для установки (аналог команды apt install) и всё те же Свойства. Не активизированы пункты Отметить для установки рекомендуемые (recommended) и предлагаемые (suggest) пакеты — они зависят от общих настроек Synaptic'а, и мы к ним ещё вернёмся.

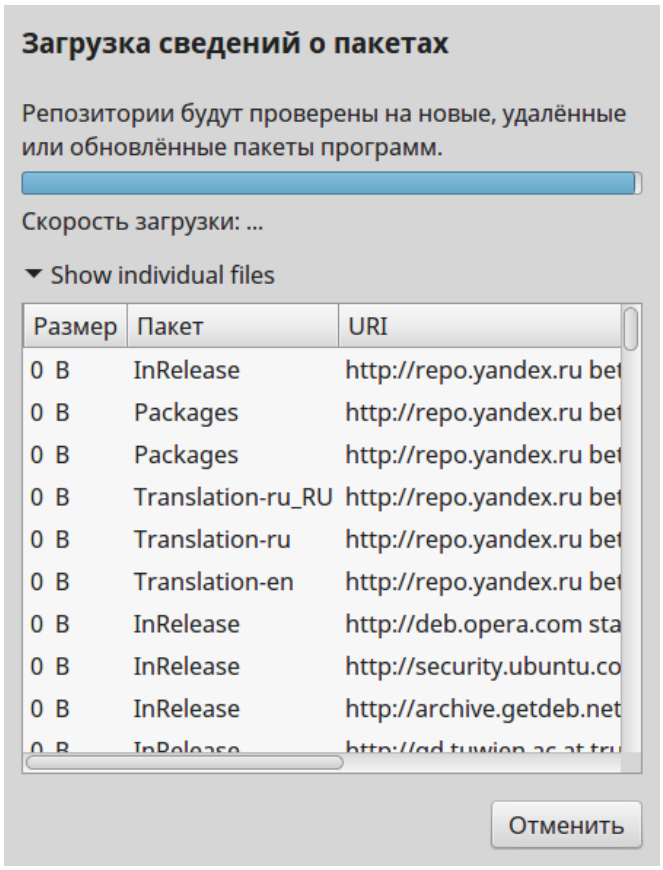
Все отметки через контекстное меню не влекут за собой немедленной установки, обновления или удаления пакетов — эти действия будут выполнены только после надатия кнопки Применить.

Теперь двинемся вверх по основным элементам интерфейса главного окна synapeic'а. Как уже говорилось, выше двух главных фреймов обнаруживается

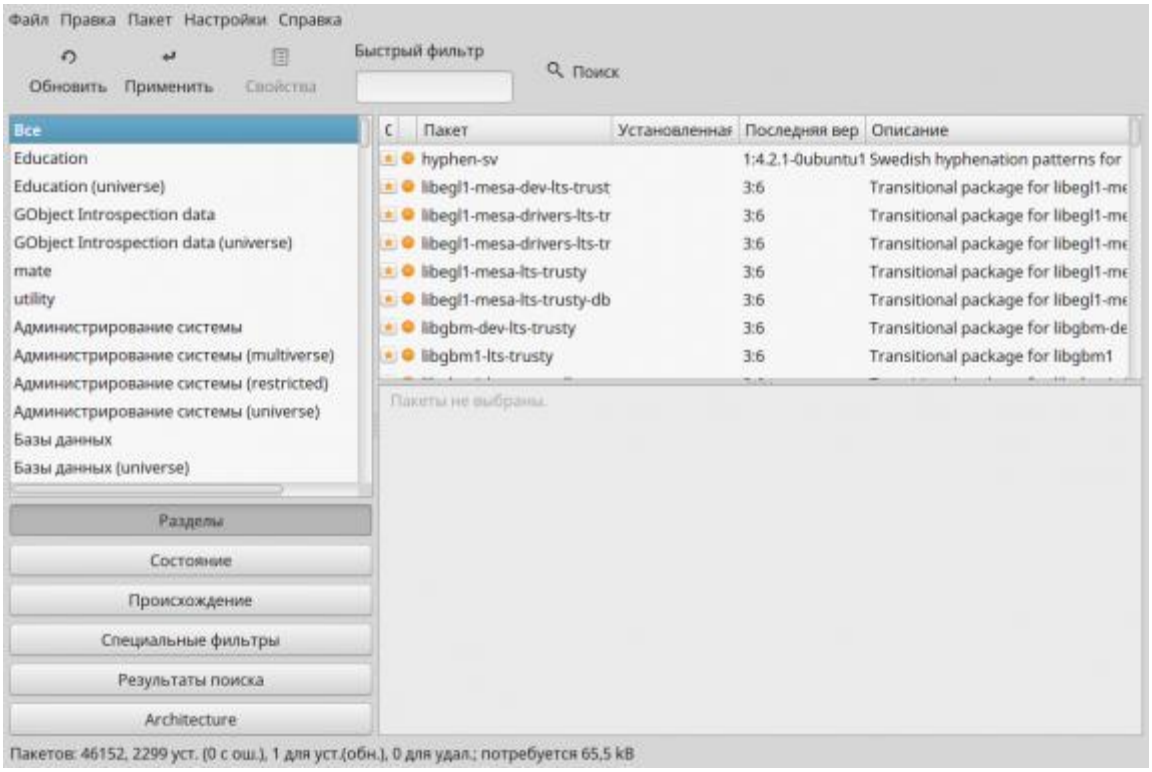
инструментальная панель, а на ней кнопки. Первой из них идёт кнопка **Обновить** — это ни что иное, как перечитывание базы данных репозиториев пакетов, тех, которые были определены в настройках:



Здесь же можно наблюдать за ходом процесса попакетно:



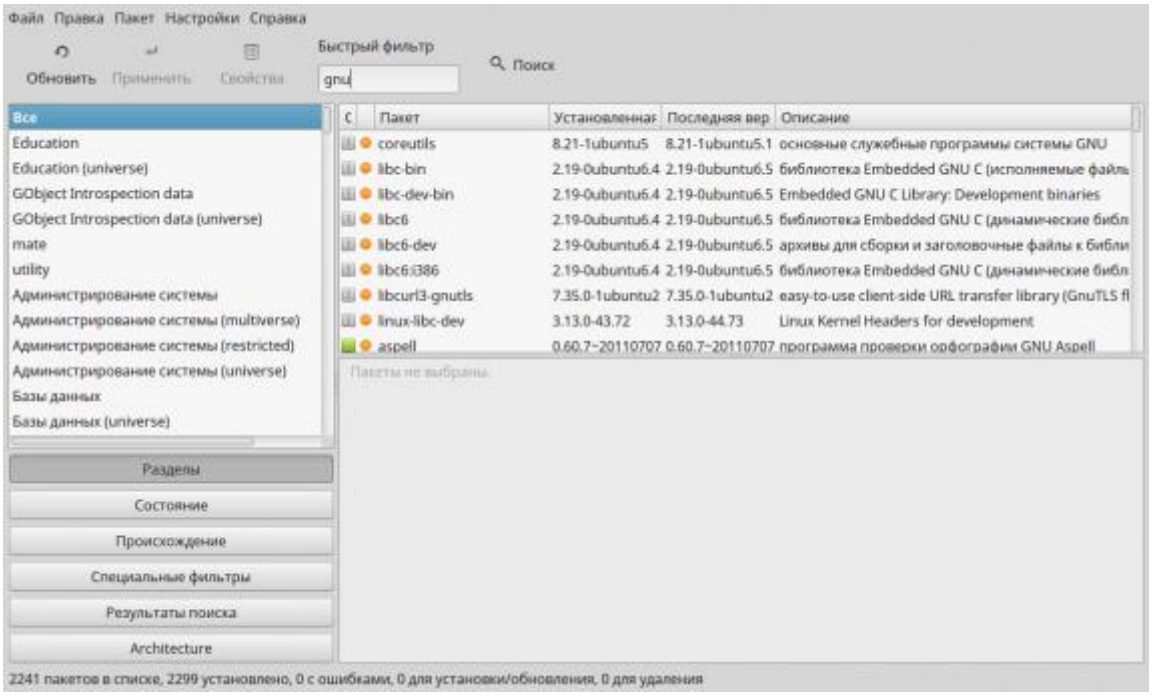
А по его завершении пакеты, для которых доступны обновления, автоматически помечаются жёлтыми звёздочками в первой колонке:



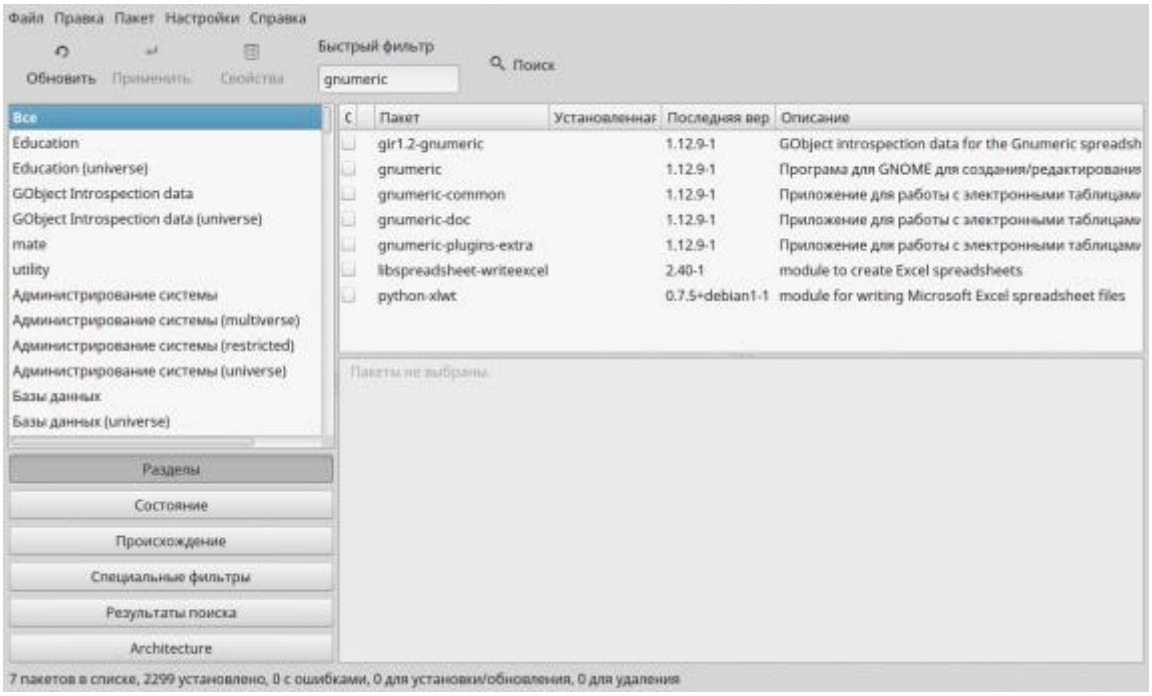
Для претворения помеченного в действительность предназначена следующая кнопка, Применить — предложение выполнить над пакетами, отмеченными для установки, обновления или удаления, соответствующие действия.

Кнопка Свойства вызывает ту же самую панель, что и пункт Свойства контекстного меню.

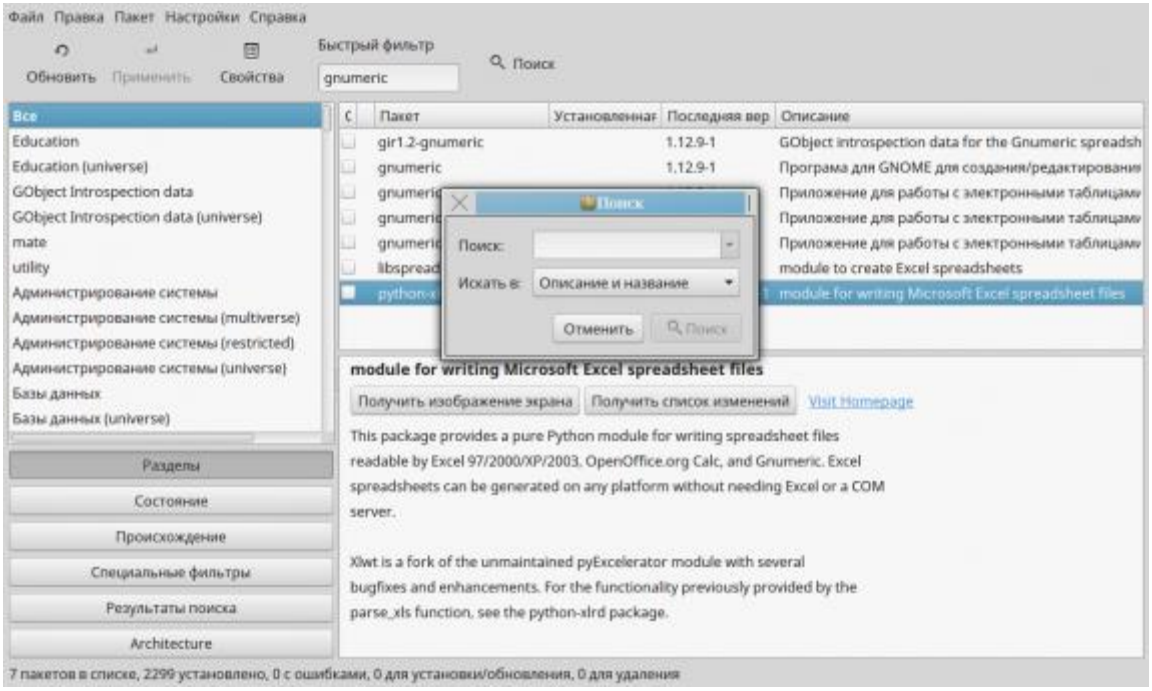
О поиске стоит поговорить отдельно. Поле Быстрый поиск предназначено для обычного наращиваемого поиска в списке правого главного фрейма в соответствие с разделами, выбранными во фрейме левом. То есть, если в последнем выбрать раздел Все, а в поле ввести gnu, мы получим список всех пакетов, сожержащих эти символы в имени, в резюме или в описании:



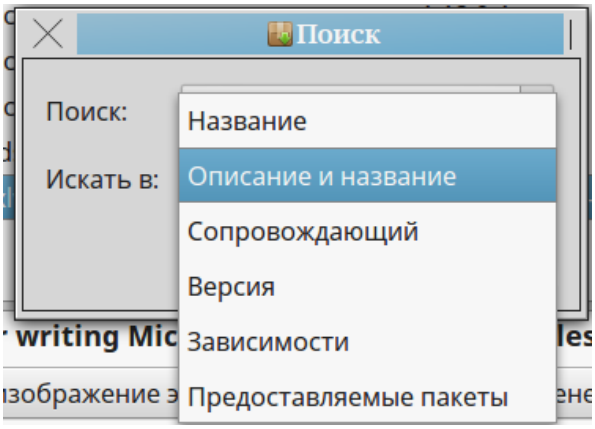
Если же мы укажем точное (или предполагаемое) имя пакета (например, gnumeric), то получим список всех пакетов, непосредственно с ним связанных:



Обратим внимание на последнюю строку в выводе результатов поиска на скриншоте: ни в имени этого пакета, ни в его кратком описании слова *gnumeric* мы не увидим — это как ра пример поиска в полных описаниях — тех самых, которые выводятся в нижнем правом фрейме (или в закладке Общие панели Свойства). А вот кнопка Найти как раз и позволяет варьировать область поиска и его критерии:



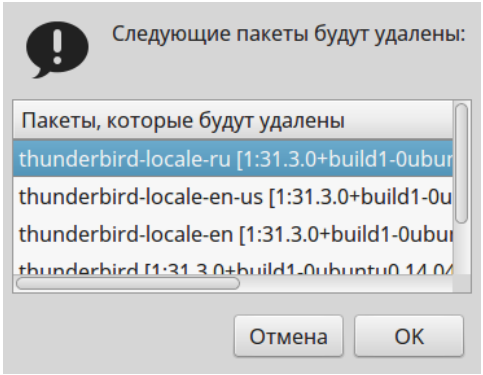
Как видно из скриншота, по умолчанию поиск проводится в именах пакетов и их описаниях. Однако обасть поиска можно ограничить только именами. Кроме того, поиск можно выполнять по имени майнтайнера пакета, номеру версии, зависимым или предоставляемым (suggest) пакетам:



9.5. Удаление пакетов: нетрадиционный метод

Среда Cinnamon в Mint предлагает несколько неожиданный метод удаления пакетов — не проверял, имеет ли он место быть в других средах и дистрибутивах. А именно — правым кликом на имени программы вызывается контекстное меню:

В котором легко видеть пункт Удалить. И это не удаление пункта из меню, что можно сделать в редакторе последнего, а именно удаление пакета (после запроса пароля), вместе со всеми теми, что от него зависят:



Однако пакеты, от которых зависит удаляемый пакет, остаются в неприкосновенности, даже если никем более не используются. Так что после удаления пакетов описанным способом не лишним будет выполнить команду

```
$ apt autoremove
```

Описанный метод по наглядности превосходит удаление через Synaptic, хотя он и не очень удобен для массового удаления пакетов (например, после стандартной инсталляции), так как каждый пакет надо удалять индивидуально, да ещё и вводя пароль на каждый чих. Так что в этой ситуации лучше воспользоваться «традиционным» методом — командой apt purge. Однако для удаления единичных пакетов, тем более поставленных «на посмотреть», он подходит как нельзя лучше.