

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО**

**ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Брянский государственный технический университет

Утверждаю

Ректор университета

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.Н.Федонин

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

**UNIX-СИСТЕМЫ**

**УСТАНОВКА ПРИЛОЖЕНИЙ В DEBIAN-ПОДОБНЫХ СИСТЕМАХ, УСТАНОВКА и СОЗДАНИЯ ПАКЕТОВ**

Методические указания

к выполнению лабораторной работы №7

для студентов очной формы обучения

по направлениям подготовки

09.03.01 «Информатика и вычислительная техника»,

09.03.04 «Программная инженерия»,

02.03.03 «Математическое обеспечение и администрирование информационных систем»

**Брянск 2020**

**УКД 004.43**

UNIX-системы. Установка приложений в Debian-подобных системах, установка и создание пакетов [Электронный ресурс]: методические указания к выполнению лабораторной работы №7 для студентов очной формы обучения по направлениям подготовки 09.03.01 «Информатика и вычислительная техника», 09.03.04 «Программная инженерия», 02.03.03 «Математическое обеспечение и администрирование информационных систем». – Брянск: БГТУ, 2020. – 29 с.

Разработал

Е.О.Трубаков,

канд. техн. наук, доц.

Рекомендовано кафедрой «Информатика и программное обеспечение» БГТУ (протокол №5 от 26.01.2020)

**Методические указания публикуются в авторской редакции**

# Цель работы

Цель работы – изучение основ установки и создания приложений в *Debian* подобных системах, а также установка пакетов rpm.

Продолжительность работы – 2 часа.

# Теоретическая часть

## Установка приложений в Debian подобных системах

Установка программ – одна из самых частых задач при администрировании системы и настройке ее под себя. В мире существует огромное количество различного программного обеспечения, но мы не устанавливаем его все одновременно, в системе собирается только то, что нам нужно на данный момент. В дистрибутивах *Linux*, установка новых программ выполняется из репозиториев с помощью специальных пакетов.

### Установка программ с помощью apt

Перед тем как говорить непосредственно про *Debian*, нужно рассмотреть *apt*. Этот пакетный менеджер был специально разработан для *Debian* и потом стал использоваться в других основанных на нем дистрибутивах. Синтаксис для работы с утилитой такой:

|  |
| --- |
| $ sudo apt опции команда имя\_пакета |

Вот основные команды *apt*, которые вы можете использовать для управления пакетами:

* *download* – скачать пакет, но не устанавливать;
* *update* – используется для обновления информации о списках пакетов в репозиториях;
* *upgrade* – обновление системы без удаления пакетов;
* *full-upgrade* – полное обновление системы с удалением конфликтующих зависимостей;
* *install* – установка пакета;
* *remove* – удаление пакета без удаления конфигурационных файлов;
* *purge* – полное удаление пакета;
* *autoremove* – автоматическое ненужных удаление пакетов;
* *search* – поиск пакета в локальной базе данных;
* *show* – посмотреть информацию о пакете.

Кроме того, утилите можно передать некоторые опции, основные:

* *-c* – сторонний конфигурационный файл;
* *-o* – строка конфигурации;
* *-t* – версия релиза для которой устанавливать пакет;
* *-f* – выполнить операцию принудительно.

Таким образом, для установки пакета из репозиториев достаточно набрать:

|  |
| --- |
| $ sudo apt install имя\_пакета $ sudo apt install vlc |

Результат представлен на рис. 1.

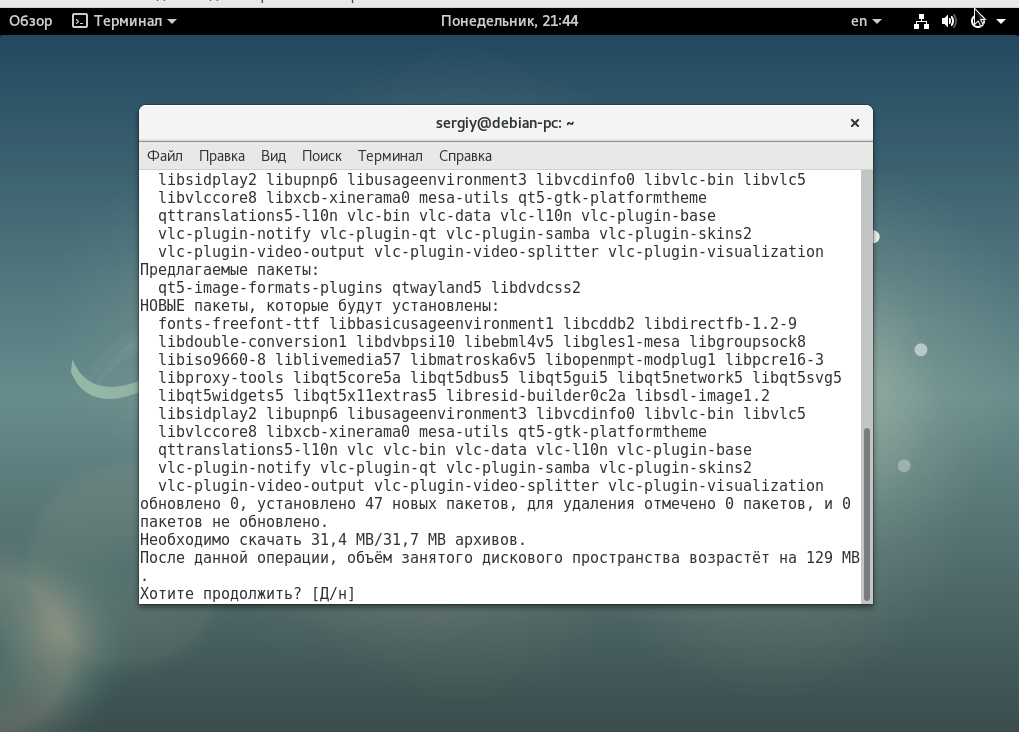


Рис. 1. Установка пакетов из репозиториев

А для удаления пакета наберите:

|  |
| --- |
| $  sudo apt purge vlc |

Вы можете искать пакеты с помощью команды *search*:

|  |
| --- |
| $ sudo apt search vlc |

Но куда более интересны политики управления репозиториями и установка отдельных пакетов из определенных хранилищ. Особенно это актуально для *Debian*, чтобы не добавлять нестабильный репозиторий полностью, а устанавливать из него только отдельные пакеты. Для решения таких задач существует настройка политики управления репозиториями.

С помощью политик вы можете настроить из каких репозиториев будет выполняться установка *deb* пакетов в *debian* в первую очередь. Допустим, вы добавили *backports*, *testing* и *unstable*. Теперь, чтобы настроить для них приоритеты откройте конфигурационный файл */etc/apt/preferences* и добавьте в него строки:

|  |
| --- |
| $  sudo vi /etc/apt/preferences  Package: \* Pin: release a=stable Pin-Priority: 700  Package: \* Pin: release a=stretch-backports Pin-Priority: 650  Package: \* Pin: release a=testing Pin-Priority: 600  Package: \* Pin: release a=unstable Pin-Priority: 100 |

Чем выше значение *Priority*, тем выше приоритет. Сначала мы указываем с помощью инструкции *Package* имя пакета, а затем репозиторий и его приоритет. В данном примере самый высокий приоритет имеет репозиторий *stable*, если там пакета нет, он будет установлен из *backports*, и только потом из *testing* и из *unstable*. Если нужно установить пакет из нестабильного репозитория, то нужно указать опцию *-t*. Например, для *testing*:

|  |
| --- |
| $ sudo apt -t testing install firefox |

### Установка программ с помощью tasksel

Инструмент *tasksel* разработан специально для *Debian*. Он позволяет устанавливать пакеты группами. Разработчики сгруппировали пакеты в зависимости от их предназначения и вы можете установить одну из нужных групп. Именно эта утилита используется в установщике *Debian*.

Чтобы посмотреть список доступных групп используется команда:

|  |
| --- |
| $ tasksel --list-tasks |

Результат представлен на рис. 2.

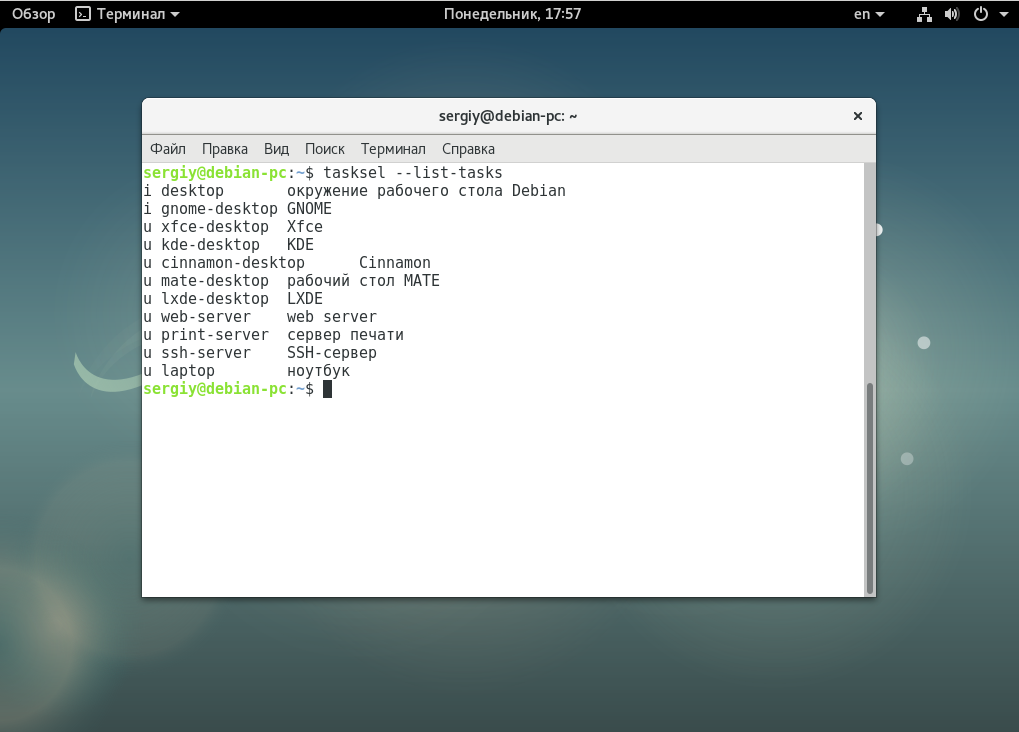


Рис. 2. Просмотр списка доступных групп используется команда

Узнать какие пакеты находятся в группе:

|  |
| --- |
| $  tasksel --task-packages web-server |

Описание группы пакетов:

|  |
| --- |
| $ tasksel --task-desc web-server |

Результат представлен на рис. 3.

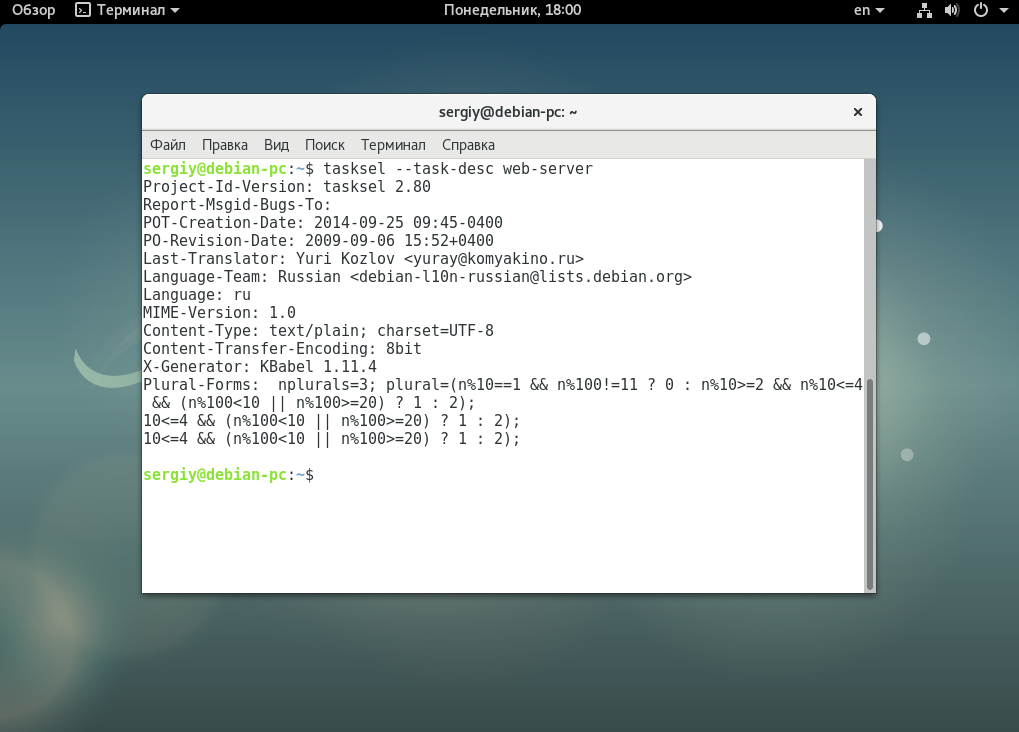


Рис. 3. Просмотр описания группы пакетов

Чтобы установить группу пакетов, например, веб-сервера, выполните:

|  |
| --- |
| $  tasksel install web-server |

Результат представлен на рис. 4.

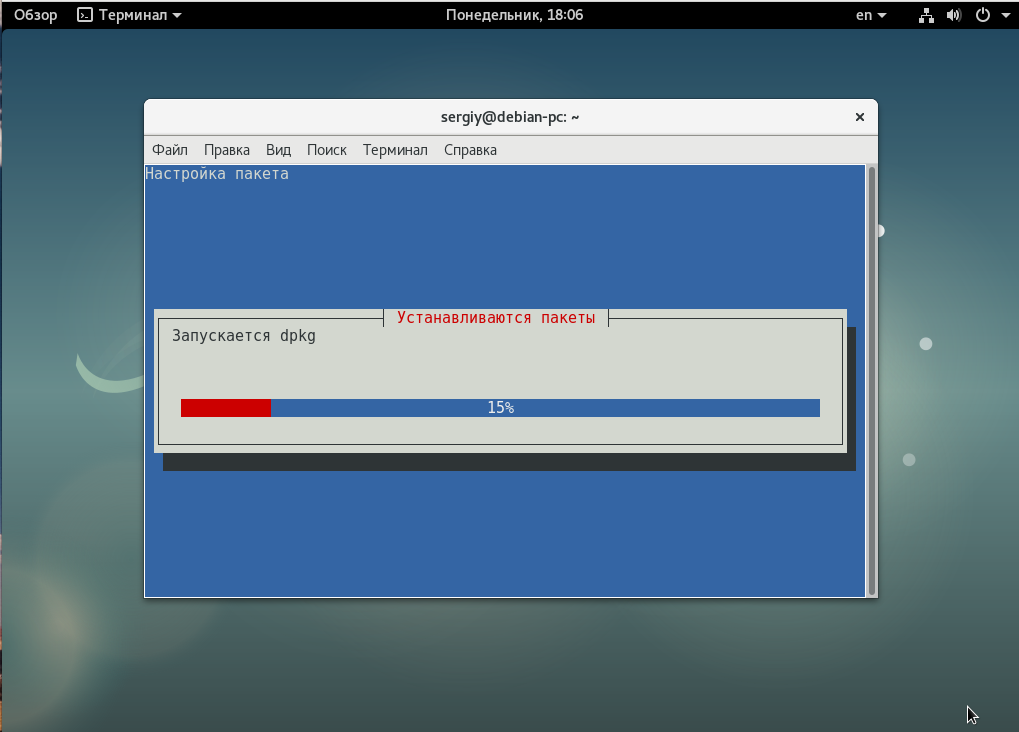


Рис. 4. Установка группы пакетов

Для удаления группы:

|  |
| --- |
| $  tasksel remove web-server |

Установка программ *Debian* таким способом достаточно проста, но здесь есть не так много пакетов.

### Установка пакетов aptitude

*Aptitude* – это более продвинутая программа для управления пакетами. Она поддерживает больше функций чем *apt*. Но для использования ее нужно установить:

|  |
| --- |
| $ sudo apt install aptitude |

Кроме консольных команд, утилита имеет псевдографический интерфейс, с помощью которого намного удобнее управлять пакетами. Возможно, установка пакетов *Debian* здесь не так удобна, зато удалить лишние пакеты и посмотреть уже установленные самое оно. Для запуска утилиты выполните:

|  |
| --- |
| $ aptitude |

Результат представлен на рис. 5.

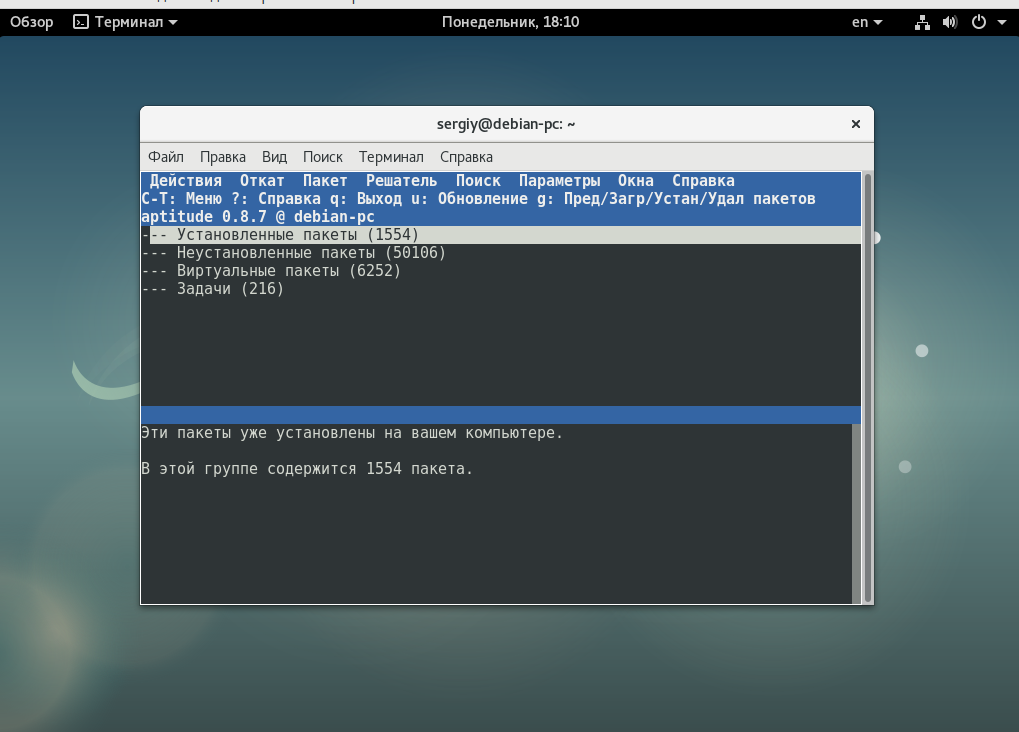


Рис. 5. Запуск утилиты *Aptitude*

Чтобы посмотреть информацию о пакете нажмите «*Enter*» (рис. 6), чтобы отметить пакет для установки, нажмите «+», а для удаления   
пакета – «-«:

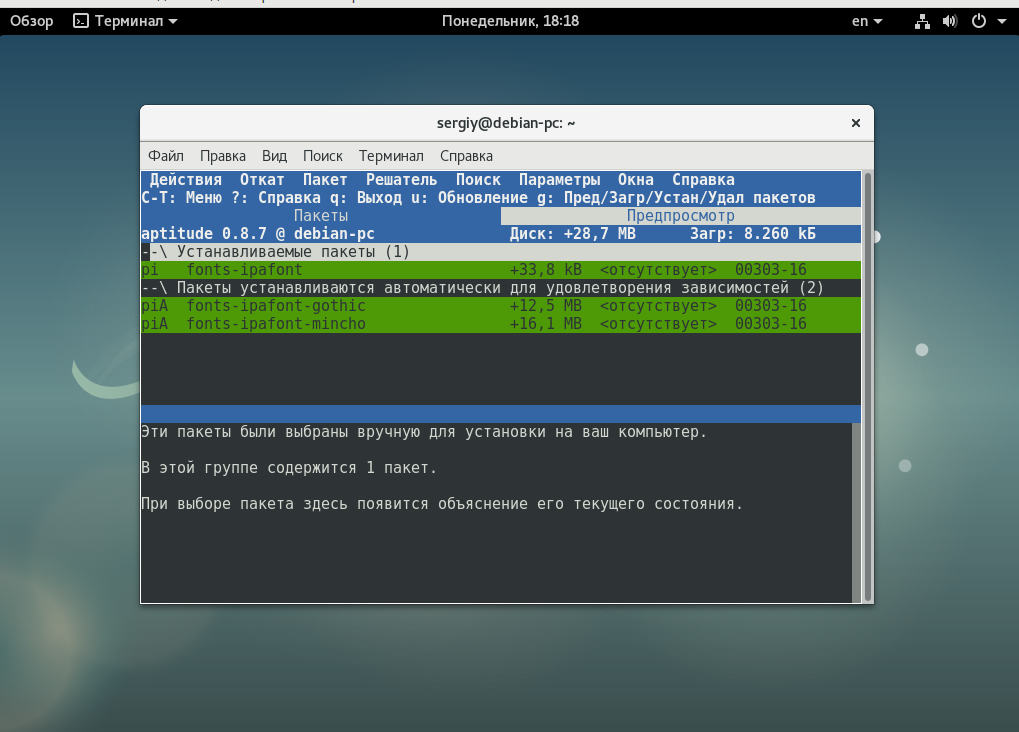


Рис. 6. Просмотр информации о пакете

Далее чтобы установить или удалить отмеченные пакеты нажмите кнопку «*g*». Затем еще раз «*g*» для подтверждения (рис. 7):

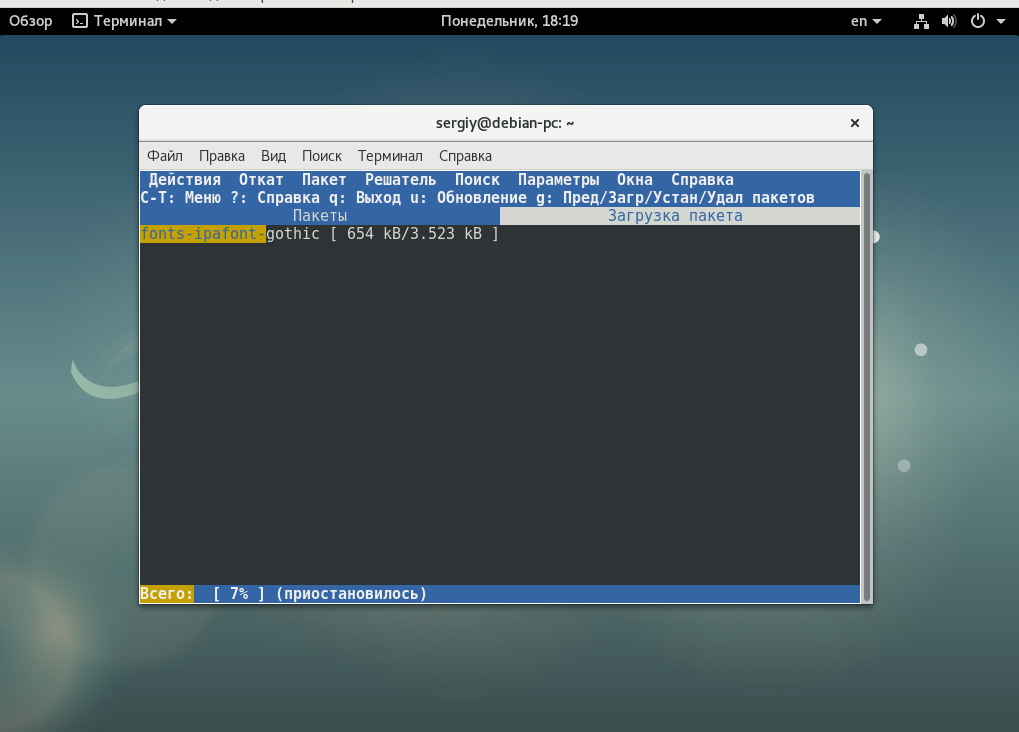


Рис. 7. Установка или удаление пакета

Необязательно использовать интерфейс *curses*, можно применять привычные консольные команды, утилита их тоже поддерживает. Синтаксис выполнения такой же, как и у *apt*, но команд намного больше:

* *Install* – установить пакет, поддерживает регулярные выражения;
* *remove* – удалить пакет;
* *purge* – удалить пакет вместе с его конфигурацией;
* *reinstall* – переустановить пакет;
* *build-depens* – отобразить зависимости пакета;
* *markauto* – пометить пакет, как установленный автоматически в виде зависимости;
* *hold* – не обновлять этот пакет;
* *keep* – если для пакета намечено какое-либо действие — отменяет его;
* *forget*-*new* – удаляет всю информацию о пакете;
* *update* – обновление системы;
* *safe-update* – обновление системы без удаления конфликтующих пакетов, такие пакеты не будут обновлены;
* *full-update* – полное обновление системы;
* *search* – поиск пакетов;
* *show* – посмотреть информацию о пакете;
* *source* – скачать исходный код пакета;
* *vesrsions* – отобразить доступные версии пакета;
* *why* – отображает причину, почему пакет не может быть установлен;
* *clean* – удаляет лишние *deb* файлы;
* *autoclean* – удаляет все пакеты из кэша;
* *download* – скачать пакет, но не устанавливать.

Установка *deb* пакетов в *debian* с помощью *aptitude* будет выглядеть аналогично *apt* (рис. 8):

|  |
| --- |
| $ sudo aptitude install vlc |

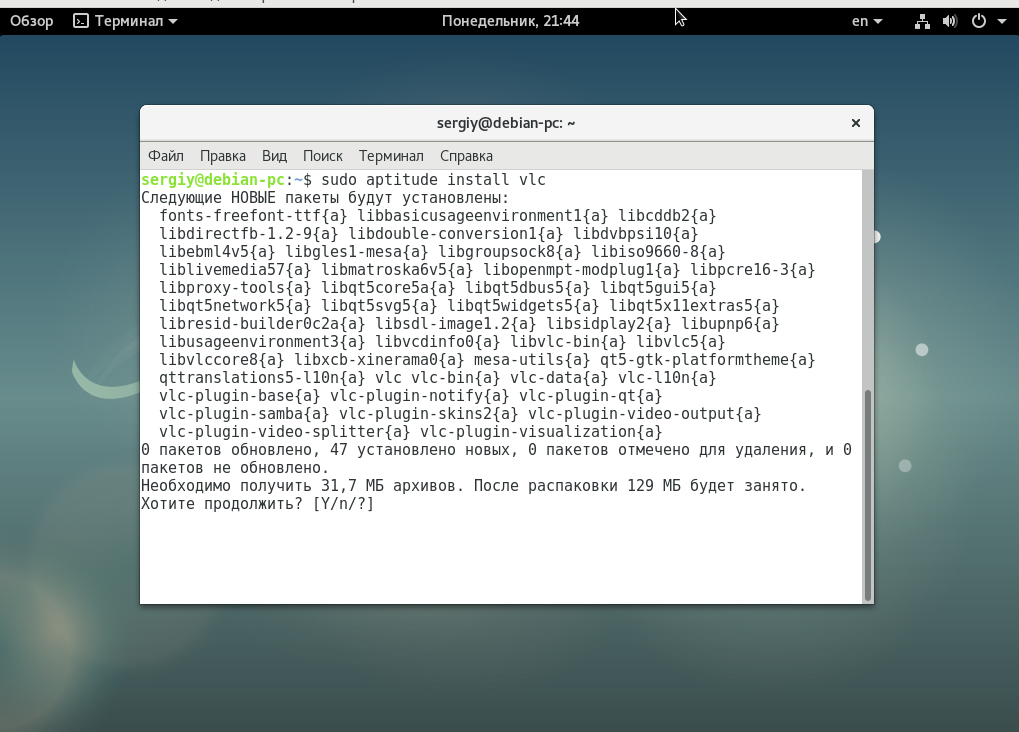


Рис. 8. Установка *deb* пакетов с помощью *aptitude*

### Установка программ Debian в Synaptic

*Synaptic* является приложением с графическим интерфейсом, которая поставляется вместе с дестрибутивом. Ее можно найти в главном меню (рис. 9)



Рис. 9. Synaptic

Для поиска нужного пакета нажмите кнопку поиск, потом введите его имя (рис. 10).

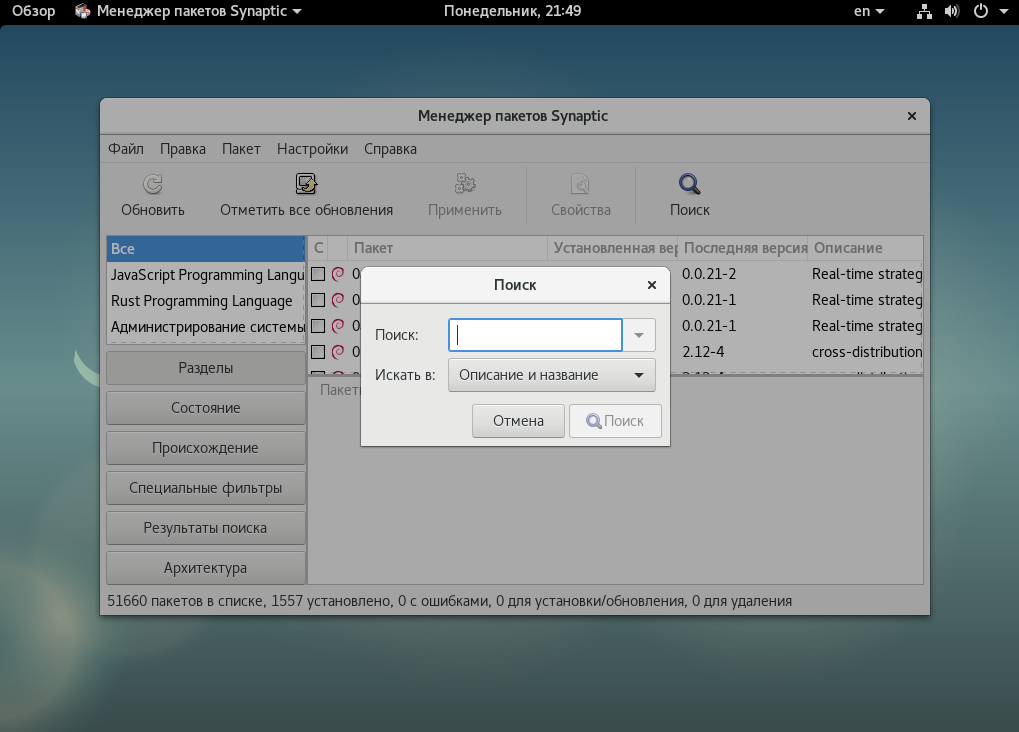


Рис. 10. Поиск пакетов

Далее, найденные пакеты будут отображены на панели (рис. 11).

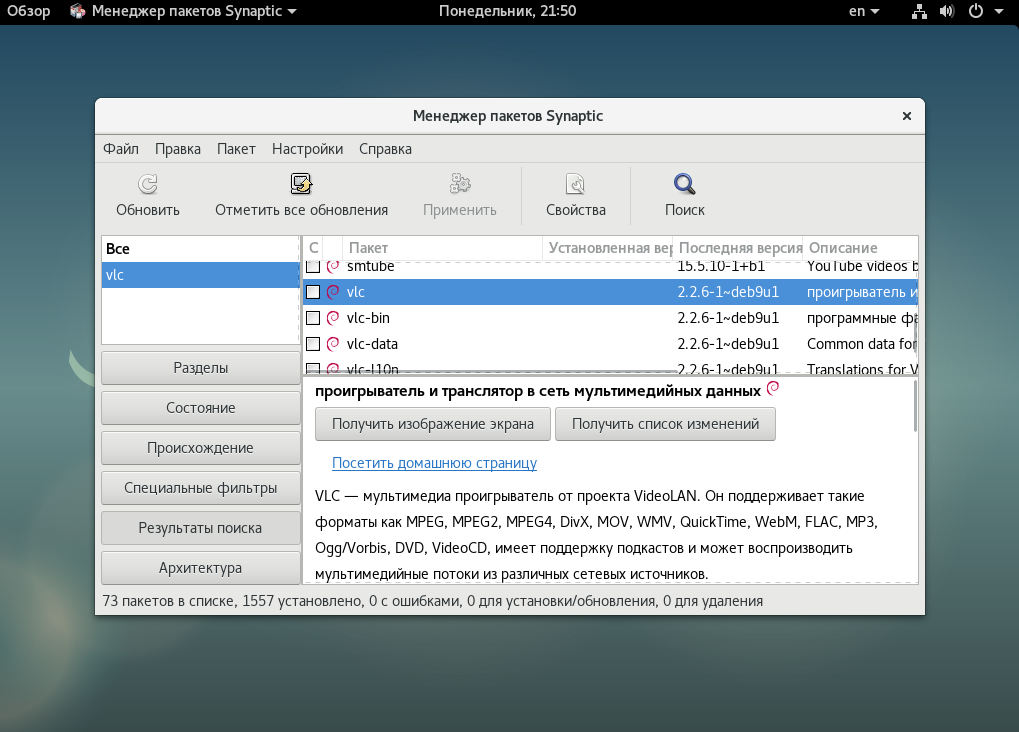


Рис. 11. Панель с найденными пакетами

Кликните по нужному пакету правой кнопкой и выберите «Отметить для установки» (рис. 12).

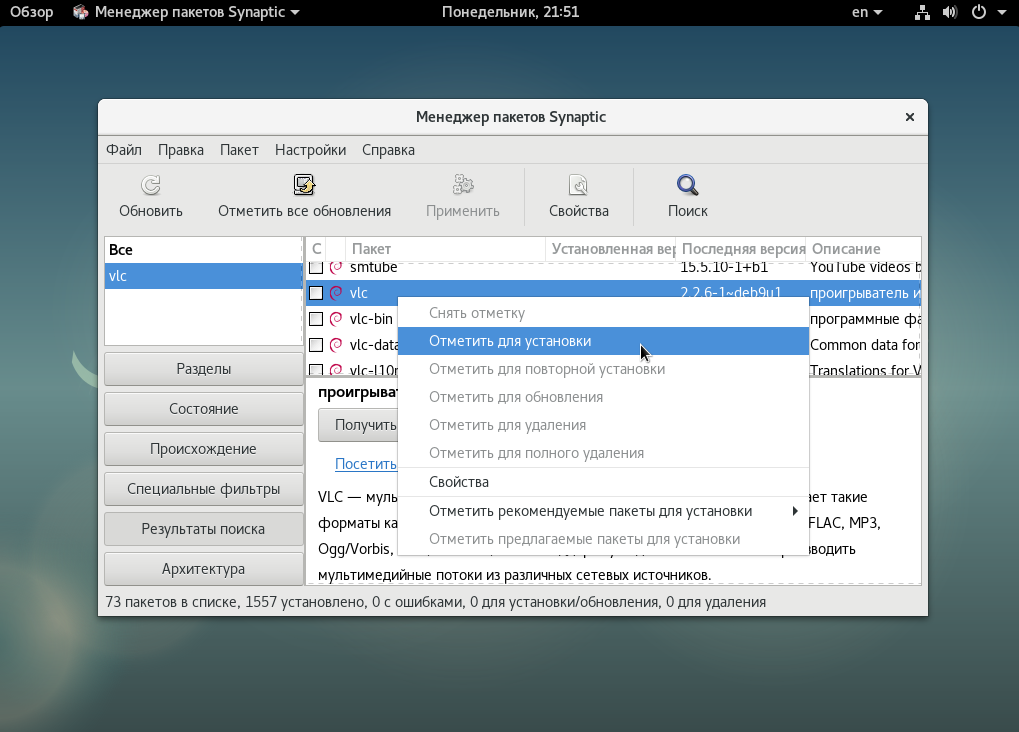


Рис. 12. Отметка пакета к установке

Утилита покажет какие еще пакеты будут установлены как зависимости (рис. 13).

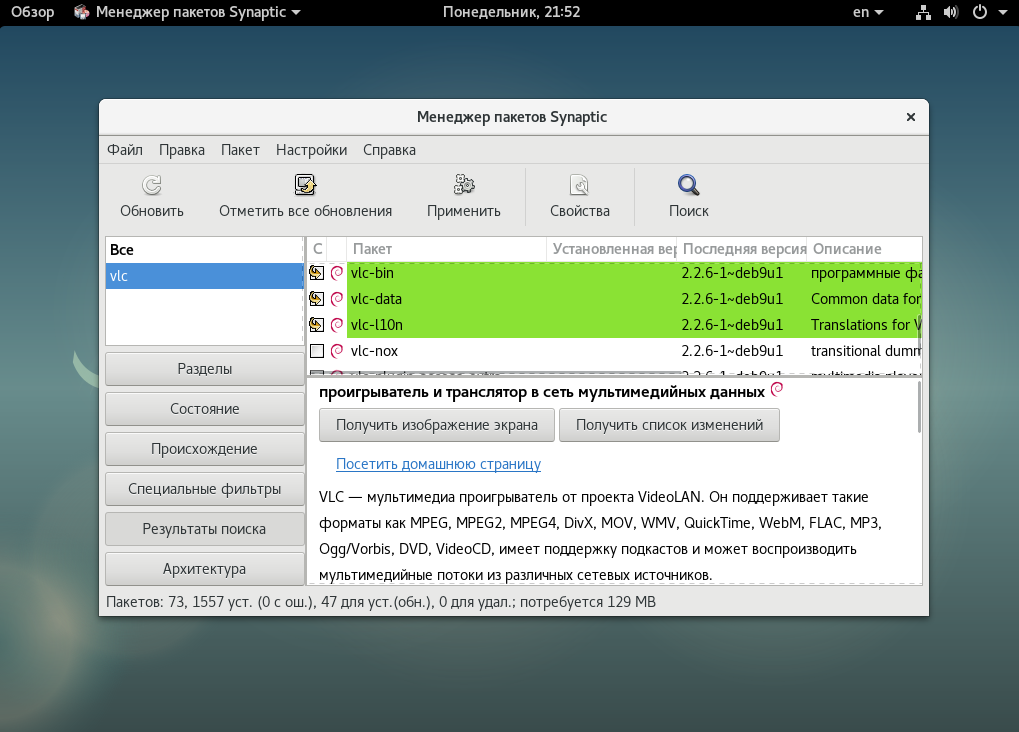


Рис. 13. Отображение зависимых пакетов

Чтобы установить выбранные пакеты, нажмите кнопку «Применить» (рис. 14).

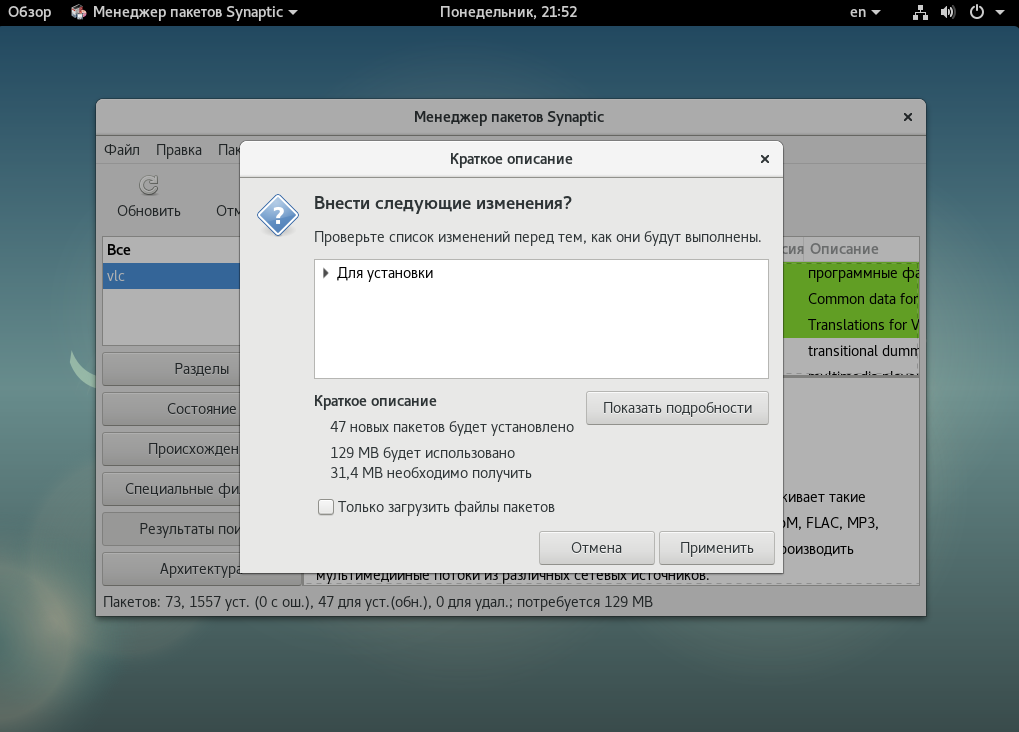


Рис. 14. Установка пакетов

Вам нужно будет подтвердить операцию и дождаться завершения установки.

## Установка пакетов rpm

*RPM* или *RPM* *Package Manager* – это пакетный менеджер, используемый в дистрибутивах *Linux*, основанных на *Red Hat*. Такое же название имеет формат файлов этого пакетного менеджера.

Этот формат не очень сильно отличается от того же самого *Deb*. Файл *rpm* – это обычный *cpio* архив, в котором содержатся сами файлы программы, а также метаданные, описывающие куда их нужно устанавливать. База всех установленных пакетов находится в каталоге */var/lib/rpm*. Из особенностей можно отметить, что *rpm* не поддерживает рекомендованные пакеты, а также зависимости формата или-или.

Для управления пакетами, так же как и в *Debian*-системах, здесь существует консольная, низкоуровневая утилита с одноименным названием – *rpm*. В разных системах используются разные пакетные менеджеры, например в *Red Hat* используется *Yum*, в *Fedora* – *DNF*, а в *OpenSUSE* – *zypper*, но во всех этих системах будет работать утилита *rpm*.

### Установка rpm пакетов в Linix

Синтаксис утилиты *rpm*:

|  |
| --- |
| $ rpm -режимопции пакет |

Утилита может работать в одном из режимов:

* *-q* – запрос, получение информации;
* *-i* – установка;
* *-V* – проверка пакетов;
* *-U* – обновление;
* *-e* – удаление.

Рассмотрим только самые интересные опции программы, которые понадобятся нам:

* *-v* – показать подробную информацию;
* *-h* – выводить статус-бар;
* *-force* – выполнять действие принудительно;
* *-nodeps* – не проверять зависимости;
* *-replacefiles* – заменять все старые файлы на новые без предупреждений;
* *-i* – получить информацию о пакете;
* *-l* – список файлов пакета.

Самая простая команда установки будет выглядеть вот так:

|  |
| --- |
| $  sudo rpm -i имя\_пакета.rpm |

Можно включить отображение статус бара в процессе установки:

|  |
| --- |
| $ sudo rpm -ivh имя\_пакета.rpm |

Удалить пакет, если он не нужен можно с помощью:

|  |
| --- |
| $  sudo rpm -e имя\_пакета |

Но у *rpm* так же как и у *dpkg*, есть один существенный недостаток. Программа не может разрешать зависимости. В случае отсутствия нужного пакета в системе, вы просто получите сообщение об ошибке и пакет не установится.

Для автоматической загрузки зависимостей во время выполнения установки *rpm* *linux* нужно использовать пакетный менеджер дистрибутива. Рассмотрим несколько команд для самых популярных *RPM* дистрибутивов. В *RedHat* и других дистрибутивах, использующих *Yum* используйте такую команду:

|  |
| --- |
| $  sudo yum --nogpgcheck localinstall имя\_пакета.rpm |

Первая опция отключает проверку *GPG* ключа, а вторая говорит, что мы будем выполнять установку локального пакета. В *Fedora*, с помощью *dnf* все делается еще проще:

|  |
| --- |
| $  sudo dnf install имя\_пакета.rpm |

Пакетный менеджер *Zypper* и *OpenSUSE* справляются не хуже:

|  |
| --- |
| $  sudo zypper install имя\_пакета.rpm |

### Установка rpm файла в GUI

Если вы используете *OpenSUSE*, то это делается очень просто. Универсальный конфигуратор системы *YaST*, кроме всего прочего позволяет установить *rpm* пакеты. Вы можете сделать это с помощью файлового менеджера, выбрав пункт контекстного меню для файла открыть с помощью *Yast* или выполнив команду:

|  |
| --- |
| $  yast2 -i имя\_пакета.rpm |

В *Fedora* для тех же целей вы можете использовать менеджер приложений дистрибутива. Раньше было еще несколько универсальных утилит для решения этой задачи, но сейчас они уже все устарели.

## Создание deb пакета

Рассмотрим основные преимущества *deb* пакетов:

* скрипты выполняются до и после установки пакета;
* автоматическое управление конфигурационными файлами: пакет не позволит затереть старые конфиги новыми без спроса;
* работа с шаблонами: возможность задавать пользователю вопросы при установке;
* изменение файлов других пакетов.

Итак, для создания полноценного пакета хватит архиваторов *tar*, *gz*, *ar*, но можно исключить лишнюю работу, и воспользоваться инструментами, созданными для облегчения жизни. Одним из таких инструментов является *dpkg* — это программное обеспечение, являющееся основой системы управления пакетами в *Debian* и ряде других операционных систем, основанных на *Debian*, например *Ubuntu*. *dpkg* используется для установки, удаления, и получения информации о *.deb* пакетах. Для того, чтобы его установить в терминале нужно указать:

|  |
| --- |
| $ sudo apt-get install dpkg debconf debhelper lintian |

Для примера будет рассмотрен некий скрипт */usr/bin/super.sh* с произвольным кодом. В данный момент неважно что выполняет этот скрипт, важно как он создастся в нужном месте.

В домашнем каталоге (или в другом месте) создаём папку, в которой будут лежать все файлы будущего пакета:

|  |
| --- |
| mkdir ~/supersh. |

Далее будем называть её корень пакета.

В корне пакета создаём папку «*DEBIAN*» (заглавными буквами, это важно). Эта папка содержит управляющую генерацией пакета информацию, и не копируется на диск при установке пакета.

Также корневая папка пакета содержит будущий «корень диска»: при установке пакета все файлы (кроме папки «*debian*») распаковываются в корень /. поэтому скрипт должен лежать по такому пути, относительно корня пакета: «*usr/bin/super.sh*»

|  |
| --- |
| mkdir -p ~/supersh/DEBIAN # управляющая папка  mkdir -p ~/supersh/usr/bin # путь к скрипту  cp super.sh ~/supersh/usr/bin/ # копируем наш скрипт в нужное место |

В итоге имеем:

*supersh/DEBIAN/*

*supersh/usr/*

*supersh/usr/bin/*

*supersh/usr/bin/super.sh*

Для создания полноценного пакета достаточно контрольного файла «*control*» в папке *DEBIAN*, все остальные используются либо для прикрепления текстовой информации (*changelog*, лицензия), либо для управления расширенными возможностями установки приложений.

Из описанных ниже файлов в папке *DEBIAN/\** нужно выбирать необходимые, и заполнить согласно инструкции.

В примере реально используется только обязательный *DEBIAN/control*.

* ***DEBIAN/control***: Основная информация. Центральный файл пакета, описывающего все основные свойства. Файл — текстовый, состоящий из пар «Атрибут: значение». Можно использовать комментарии: символ "#" в начале строки (возможность была добавлена в версии dpkg >= 1.10.11) (описание его полей см. в прил.1). Пример:

|  |
| --- |
| Package: supersh Version: 1.0-1 Section: misc Architecture: all Depends: bash, sed (>= 3.02-8) Maintainer: o\_O Tync <o-o-tync.habrahabr.ru> Description: Super Shell Script ␣A super example script. ␣. ␣It does nothing |

* ***DEBIAN/copyright***: Текст лицензии. Файл не обязателен, но лучше подчеркнуть своё авторство.
* ***DEBIAN/changelog***: история изменений. *Changelog* в специальном формате: используется *dpkg* для получения номера версии, ревизии, дистрибутива и важности пакета. Пример:

|  |
| --- |
| supersh (1.0-1) stable; urgency=medium \* Testing. -- o\_O Tync <o-o-tync.habrahabr.ru> Sun, 13 Dec 2009 00:11:46 +0300 |

* ***DEBIAN/rules***: правила компиляции. Используется для управления компиляцией пакета (При *Architeture: source*).
* ***DEBIAN/conffiles***: список файлов конфигурации. Обычно пакеты содержат болванки конфигурационных файлов, например, размещаемых в */etc*. Очевидно, что если конфиг в пакете обновляется, пользователь потеряет свой отредактированный конфиг. Эта проблема легко решается использованием папок типа «*config.d*», содержимое которых включается в основной конфиг, заменяя собой повторяющиеся опции.  
  Файл «*DEBIAN/conffiles*» позволяет решить проблему иначе: он содержит список файлов конфигурации (по одному на строке). Если в текущей версии пакета один из этих файлов обновляется, то пользователь получает предупреждение о конфликте версий конфигов, и может выбрать: удалить, заменить, или сделать *merge*.  
  На каждой строке должен быть полный абсолютный путь до каждого конфига. Например:

|  |
| --- |
| /etc/supersh/init.conf  /etc/supersh/actions.conf |

* ***DEBIAN/dirs***: список папок для создания. «Список абсолютных путей к папкам, которые требуются программе, но по каким-либо причинам не создаются.» — гласит официальная документация. На практике – здесь перечисляются все папки, так или иначе используемые программой: и где лежат бинарники, которые используются программой. Удобно использовать для создания нескольких пустых папок. По одной на строке. Например:

|  |
| --- |
| /var/log/supersh  /var/lib/supersh |

* ***DEBIAN/menu***: создание пунктов меню Файл для создания пунктов меню. Подробно написано в *man5* *menufile*.  
  В */usr/share/applications* видим кучку *\*.desktop* файлов: это и есть пункты меню. Они представляют собой текстовые файлы с синтаксисом наподобие *ini*-файла. Создаем нужный пункт меню *\*.desktop* и кладем файл в *usr/share/applications/*. Иконка для него должна лежать в *usr/share/pixmaps*.  
  После этого в *postinst* скрипт нужно добавить выполнение команды обновления меню *update-menus*:

|  |
| --- |
| if [ "$1" = "configure" ] && [ -x "`which update-menus 2>/dev/null`" ] ; then  update-menus  fi |

* ***DEBIAN/md5sums***: контрольные суммы файлов. Используется для проверки целостности пакета. Заполняется так (*cwd*=корень пакета):

|  |
| --- |
| $ md5deep -r usr > DEBIAN/md5sums |

* ***DEBIAN/watch***: мониторинг сайта. Файл содержит инструкции для программ uscan и uupdate. Используя эту возможность, можно следить за сайтом, откуда были получены исходники пакета, и обеспечивать контроль качества дистрибутива в целом. Пример:

|  |
| --- |
| # Site Directory Pattern Version Script ftp.obsession.se /gentoo gentoo-(.\*)\.tar\.gz debian uupdate |

* ***DEBIAN/cron.d***: инсталляция заданий *cron*.

|  |
| --- |
| # # Launches super.sh periodically # 0 4 \* \* \* root [ -x /usr/bin/super.sh ] && usr/bin/super.sh |

* ***DEBIAN/inid.d****: init*-скрипт. В этот файл пишется содержимое *init*-скрипта. Скрипты позволяют управлять установкой, переустановкой и удалением пакета, выполняя действия, которые нельзя сделать простым копированием файлов в правильные места. Это может быть скачивание дополнительных файлов (как это делает *flash-installer*), изменение существующих, а также — вывод интерактивных (*GUI* или *ncurses*) диалогов, позволяющих пользователю сконфигурировать пакет под себя: например, *mysql* спрашивает какой установить пароль для *root*.  
  Все скрипты выполняются от пользователя *root* (а как же ещё :). Также они получают аргументы (которые обрабатывать не обязательно), конкретизирующие на каком именно этапе находится установка.
* ***DEBIAN/(preinst|postinst|prerm|postrm)***: скрипты установки. Всего можно создать до четырёх скриптов в одном пакете:

|  |  |
| --- | --- |
| **Скрипт** | **Назначение** |
| DEBIAN/preinst | Выполняется перед установкой пакета: он может подготовить что-либо для успешной установки |
| DEBIAN/postinst | Выполняется сразу после установки пакета: он настраивает установленный пакет так, чтоб он был готов к работе. Здесь также выполняется интерактивная конфигурация пакета: это делается при помощи dh\_input и файла DEBIAN/templates |
| DEBIAN/prerm | Выполняется непосредственно перед удалением пакета: обычно этот скрипт подчищает установочные пути пакета так, чтоб ничего лишнего не осталось. |
| DEBIAN/postrm | Выполняется сразу после удаления пакета: вычищает остатки |

Обратите внимание, что ошибки, возникающие в этих скриптах никак не логируются: возвращается только код возврата скрипта, и логирование необходимо делать вручную. Для этого можно использовать в начале каждого скрипта следующий шаблон: он будет сохранять в *syslog* все возникающие ошибки.

|  |
| --- |
| #!/bin/bash set -e # fail on any error set -u # treat unset variables as errors # ======[ Trap Errors ]======# set -E # let shell functions inherit ERR trap # Trap non-normal exit signals: # 1/HUP, 2/INT, 3/QUIT, 15/TERM, ERR trap err\_handler 1 2 3 15 ERR function err\_handler { local exit\_status=${1:-$?} logger -s -p "syslog.err" -t "ootync.deb" "supersh.deb script '$0' error code $exit\_status (line $BASH\_LINENO: '$BASH\_COMMAND')" exit $exit\_status }  ... Ваш код установочного скрипта ...  exit 0 |

***DEBIAN/templates***: шаблоны для диалогов. Файл *DEBIAN/templates* содержит данные, используемые при выводе диалоговых окон (*GUI* или *ncurses*). Файл содержит блоки, разделённые пустой строкой. Каждый блок определяет ресурсы, используемые в одном конкретном диалоговом окне. Шапка для всех типов диалогов стандартная:

|  |
| --- |
| Template: supersh/template-name Type: string Default: Default-value Description: Dialog-title ␣Dialog-text |

|  |  |
| --- | --- |
| Тип | Описание шаблона |
| string | Приглашение на ввод текстовой строки. |
| password | Приглашение на ввод пароля.  Для этого типа шаблона нет значения Default. |
| boolean | Имеет строковое значение «true» или «false» |
| select | Возможность выбора одного из нескольких вариантов. Варианты предлагаются в дополнительном атрибуте шаблона:  Choices: yes, no, maybe |
| multiselect | Возможность выбора нескольких вариантов галочками. Варианты предлагаются в дополнительном атрибуте шаблона:  Choices: sex, drugs, rock-n-roll |
| text | Выводит на экран дополнительный текст |
| note | Выводит на экран текст: важная информация |
| error | Выводит на экран текст: очень важная информация, критическая. |

Для шаблонов *text*, *note*, *error* также нет значения *Default*.

Для сборки пакета первое, что нужно сделать — это рекурсивно выставить всем файлам в корне пакета пользователя и группу *root:root* (или другие, если потребуется). Это нужно затем, что файлы пакета упаковываются в *tar.gz* архив который сохраняет и права доступа к файлам, и владельца. Однако делать это не обязательно. Есть отличная команда *fakeroot* которая при создании архива подменит владельца файлос *root*-ом.

В примере ниже, скрипт должен иметь бит выполнимости.

Затем нужно перейти на папку назад, чтоб было видно корневую папку пакета, и пакет создаётся сам:

|  |
| --- |
| $ fakeroot dpkg-deb --build supersh |

Созданный пакет необходимо переименовать, чтобы он соответствовал порядку именования *\*.deb* пакетов: <имя пакета>\_<версия>\_<архитектура>.*deb*.

|  |
| --- |
| $ mv supersh.deb supersh\_1.0-1\_all.deb |

# ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Необходимо установить несколько браузеров (*Chromium*, *Firefox*, Яндекс браузер).

1. Установить пакет при помощи apt.
2. Установить пакет при помощи tasksel.
3. Установить пакет при помощи aptitude.
4. Разработанные в рамках прошлой лабораторной работы утилиты оформить в виде установочных пакетов под используемую операционную систему.
5. Приложение должно устанавливать в директорию /*opt*
6. При установке приложение должно компилироваться из исходных кодов.
7. При установке пакеты все необходимые для его сборки утилиты, библиотеки и другие ресурсы должны автоматически устанавливаться.

# Контрольные вопросы

1. Установка приложений в Debian подобных системах?
2. Установка программ с помощью apt.
3. Установка программ с помощью tasksel.
4. Установка пакетов aptitude.
5. Установка программ Debian в Synaptic.
6. Что содержится в папке *DEBIAN* при создании *deb* пакета?
7. Какого файла достаточно для создания пакета *deb*, и что содержит этот файл?
8. Для каких шаблонов в файле *DEBIAN/templates* обязательно нужно устанавливать значение по умолчанию, а для каких нет? Почему?
9. Что такое *dpkg*?
10. Какой синтаксис для работы с утилитой *apt*? и какие ее основные команды?
11. С помощью каких средств можно устанавливать программы в *Debian*? Расскажите о каждом в нескольких предложениях.
12. Что такое *RPM*? В каких дистрибудивах используется данная утилита?
13. Какой синтаксис утилиты *rpm*? Перечислите ее режимы и основные опции.
14. Какая команда используется для создания пакета *deb*? Какая команда используется для установки программ в *Debian*? Какая команда используется для установки rpm пакета?

# Список рекомендуемой литературы

## Основная литература

1. Курячий Г.В. Операционная система UNIX: учебное пособие [Электронный ресурс] / Курячий Г.В.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 258c. – Режим доступа: http://www.iprbookshop.ru/52199.
2. Администрирование ОС Unix [Электронный ресурс] / – 2-е изд. –М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 303 c. – Режим доступа: http://www.iprbookshop.ru/73659.html.

## Дополнительная литература

1. Командная строка UNIX: практикум [Электронный ресурс] — М.: Московский государственный строительный университет, ЭБС АСВ, 2013. – 44c. – Режим доступа: http://www.iprbookshop.ru/23729.
2. Мошков, М.Е. Введение в системное администрирование Unix [Электронный ресурс] / М.Е. Мошков. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 208 c. – Режим доступа: http://www.iprbookshop.ru/73672.html.

***Приложение 1***

***Поля файла DEBIAN/control***

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Описание** | **Пример** |
| **Основные** | | |
| Package:  (обязательное поле) | Имя пакета: [a-zA-Z0-9-] — только латиница, цифры, и дефис. Имя используется при установке:  apt-get install <package> | Package: supersh |
| Version:  (обязательное поле) | Версия пакета (и программы внутри). Используется для определения «обновлять ли».  Формат принят такой:  <версия\_программы>-<версия\_пакета>.  Рекомендую всегда указывать версию пакета: при изменении структуры пакета цифра увеличивается на единичку.  Допустимые символы: можно использовать дату и буквы. | Version: 1.0-1<br>Version: 2009.12.12-1 |
| Provides | Имя приложения (возможно, виртуальное), регистрируемое в системе в результате установки этого пакета.  Используется редко: в основном, если нужно изменить имя пакета, или если более одного пакета предлагают одинаковый функционал. Например, пакеты Apache и nginx предоставляют возможность демона httpd:  Provides: httpd | Provides: supersh |
| Maintainer  (обязательное поле) | Имя и почта мэйнтейнера пакета: человека, который «дебианизировал» приложение.  Формат произвольный, но принято имя <e-mail> | Maintainer: o\_O Tync <o-o-tync.habrahabr.ru> |
| Architecture  (обязательное поле) | Архитектура процессора, для которой предназначен пакет.  Допустимые значения: i386, amd64, all, source  all используется для скриптов  source используется для компилируемых пакетов с исходниками | Architecture: all |
| Section  (обязательное поле) | Определяет задачу, для которой приложение обычно используется (группа приложений).  Возможные значения: admin, base, comm, contrib, devel, doc, editors, electronics, embedded, games, gnome, graphics, hamradio, interpreters, kde, libs, libdevel, mail, math, misc, net, news, non-free, oldlibs, otherosfs, perl, python, science, shells, sound, tex, text, utils, web, x11 | Section: misc |
| Description  (обязательное поле) | Описание пакета.  Описание состоит из двух частей: короткое описание (70 символов) на той же строке, и длинное описание на последующих строках, начинающихся с пробела.  В расширенном описании все переводы строки игнорируются. Для вставки \n используется одиночная точка. | Description: Short.<br>␣Long <br>␣goes here.<br>␣.<br>␣New line. |
| **Связи и зависимости** | | |
| Depends | Список пакетов через запятую, которые требуются для установки этого пакета.  После имени пакета можно в круглых скобках указать ограничение на версию, используя операторы: <<, =, >>, <=, >=. Если оператор не указан — используется >= | Depends: dpkg, libz (>= 1.2.3), jpeg (= 6b), png (< 2.0) |
| Pre-Depends | Список пакетов, которые требуются в процессе установки этого пакета.  Эти зависимости могут потребоваться для скриптов установки пакета: например, пакет flash-installer требует wget  Можно использовать ограничения на версию (см. Depends). | Pre-Depends: wget (>= 1.0) |
| Conflicts | Список пакетов, которые не могут быть установлены одновременно с этим.  Установка не удастся, если хоть один из перечисленных пакетов уже будет установлен. | Conflicts: crapscript |
| Replaces | Список пакетов, файлы которых модифицируются этим пакетом.  Требуется в случае создания «пакета-патча», изменяющего что-либо: в противном случае при замене файлов чужого пакета возникнет ошибка при установке. | Replaces: ut2004 |
| Recommends | Список пакетов, рекомендуемых к установке.  Эти пакеты не обязательны, но обычно используются вместе с текущим. | Recommends: superplatform |
| Suggests | Список пакетов, предлагаемых к установке.  Эти пакеты не обязательны, но с ними программа работает лучше. | Suggests: supersh-modules |
| Build-Depends | (Только для Architecture: source)  Список пакетов, требуемых для компиляции исходников.  То же, что и Depends, но логически отделено. | Build-Depends: cmake |
| **Экстра** | | |
| Installed-Size | Размер файлов пакета в килобайтах.  Просто цифра, округлённая до ближайшего целого. Используется менеджером пакетов для определения суммарного требуемого объёма на диске. | Installed-Size: 3 |
| Priority | Приоритет пакета: насколько он важен в системе.  Возможные значения: extra, optional, standard, important, required (такие пакеты не удаляются вообще). | Priority: optional |
| Esssential | Если установить этот атрибут в значение «yes», пакет нельзя будет удалить. | Esssential: yes |
| Origin | Строка: откуда получены программы в пакете. Обычно используется URL сайта автора, почта или имя. | Origin: brain |
| X-Source | Полная ссылка на \*.tar.gz архив с исходниками | X-Source: ...\*.tgz |

UNIX-системы. Установка приложений в Debian-подобных системах, установка и создание пакетов [Электронный ресурс]: методические указания к выполнению лабораторной работы №7 для студентов очной формы обучения по направлениям подготовки 09.03.01 «Информатика и вычислительная техника», 09.03.04 «Программная инженерия», 02.03.03 «Математическое обеспечение и администрирование информационных систем»

ТРУБАКОВ ЕВГЕНИЙ ОЛЕГОВИЧ

Научный редактор Д. А. Коростелев

Компьютерный набор Е.О. Трубаков

Иллюстрации Е.О. Трубаков

Подписано в печать \_\_.\_\_.\_\_. Усл.печ.л. 1,68 Уч.-изд.л. 1,68

Брянский государственный технический университет

241035, Брянск, бульвар 50 лет Октября, 7 БГТУ

Кафедра «Информатика и программное обеспечение», тел. 56-09-84