

# Администрирование Linux: Управление пакетами



Александр  
Зубарев



## **Александр Зубарев**

Преподаватель высшей квалификационной категории

**АКТ (ф) СПбГУТ им. проф. Бонч-Бруевича М.А.**

---

# Предисловие

**На этом занятии мы поговорим о:**

- об устройстве пакета;
- о репозиториях;
- о менеджерах пакетов;
- об исходных файлах.

**По итогу занятия** вы узнаете, как управлять программным обеспечением, используя менеджеры пакетов.



# План занятия

1. [Предисловие](#)
2. [Репозитории и пакеты](#)
3. [Пакетный менеджер APT](#)
4. [Пакетный менеджер YUM](#)
5. [Прочие пакетные менеджеры](#)
6. [Итоги](#)
7. [Домашнее задание](#)



# Репозитории и пакеты

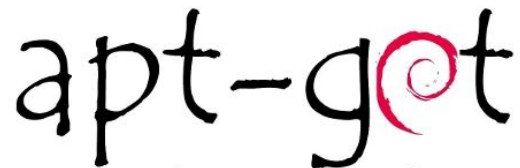
---

# Репозитории и менеджеры пакетов

**Репозиторий** — хранилище данных, в данном случае — пакетов ПО.

**Менеджер пакетов** — специальное программное обеспечение, которое управляет загрузкой, установкой, удалением пакетов, а также решением зависимостей.

Наиболее популярными менеджерами пакетов являются **APT** (семейство Debian) и **YUM** (семейство Red Hat)

The logo for APT (Advanced Package Tool) features the text 'apt-get' in a black, lowercase, sans-serif font. The 'o' in 'got' is replaced by a red circular icon with a white spiral inside.The logo for YUM (Yellowdog Updater Modified) features the text 'yum' in a large, bold, blue, lowercase, sans-serif font. Below it, in a smaller, lighter blue font, is the text 'yellowdog updater modified'.

---

# Пакет программного обеспечения

**Пакет** – это архив специального формата, который содержит:

- все необходимые приложению **бинарные и конфигурационные файлы**;
- информацию о том, как их следует разместить в файловой системе;
- данные о **зависимостях пакета**;
- **список действий**, которые необходимо выполнить в процессе установки.

**Метапакет** – пакет ПО, включающий в себя группу пакетов, объединенную по какому-то признаку.

# Зависимости

Практически любая достаточно объемная программа требует для своей работы дополнительные библиотеки и компоненты.

Такие **дополнительные материалы, необходимые для работы приложения**, устанавливаемого из пакетов, называют зависимостями.

Современные пакетные менеджеры решают эту проблему благодаря **описанию зависимостей в пакете**.

Но так было не всегда.



# Сторонние репозитории

Иногда крупные разработчики программного обеспечения (яркий пример – РНР) не хотят выкладывать свои приложения в базовые репозитории того или иного дистрибутива, а вместо этого организуют свои собственные репозитории, в которых хранят пакеты, доступные для установки на разные дистрибутивы Linux.

Такие **репозитории** называют **сторонними**.

Так же зачастую сторонние репозитории организуют разработчики проприетарного ПО.



## Подключение сторонних репозиториев

Для того чтобы иметь возможность устанавливать ПО из сторонних репозиториев, их надо указать в [source.list](#) вашего пакетного менеджера и скачать [gpg-ключ](#) для него.

# Подключение сторонних репозиториев

## Разберем на примере Debian и PHP:

1. Установим необходимое ПО:

```
sudo apt install apt-transport-https lsb-release ca-certificates
```

2. Скачаем gpg-ключ:

```
sudo wget -O /etc/apt/trusted.gpg.d/php.gpg  
https://packages.sury.org/php/apt.gpg
```

3. Добавим адрес репозитория в source.list:

```
sudo sh -c 'echo "deb https://packages.sury.org/php/ $(lsb_release  
-sc) main" > /etc/apt/sources.list.d/php.list'
```

---

# Компиляция пакета из исходников

В современном мире не всегда приложение упаковывается в **готовый пакет**.

Самые последние версии ПО, к примеру, зачастую распространяются в форме **исходного кода**.

---

# Компиляция пакета из исходников

Использование ПО из таких источников имеет некоторые **преимущества и недостатки:**

1. Вам надо самостоятельно скомпилировать приложение.  
➡ В случае коллизий – самостоятельно решить появившиеся проблемы.
2. Скомпилированное из исходников ПО не считается установленным для менеджера пакетов, он его просто не видит.  
➡ Удалять надо будет руками.
3. Это зачастую единственный способ получить последние или не очень популярные, но необходимые для работы программы

---

# Команда make

Для сборки нам нужны **компиляторы**: они прописаны в зависимостях пакета **build-essential**, так что достаточно установить его со всеми зависимостями.

Ещё нужны **autoconf** и **automake**.

Убедитесь в наличии файла **configure**, необходимого для **процесса сборки**.

Для его генерации надо выполнить:

**./bootstrap** или **./autogen.sh**.

# Команда make

`make` – сборка пакета, установка в нем нужных параметров и подготовка к установке.

`make install` – устанавливает сконфигурированное приложение в систему.

Этот способ вполне рабочий на случай, если вы **не планируете** дальше поддерживать программу.

Максимально правильным действием будет **собрать пакет и установить его при помощи менеджера пакетов.**

# Сборка пакета

`checkinstall` – программа для сборки **.deb-пакета** дистрибутива **Ubuntu**. Использование этой программы не всегда работает по причине отсутствия описания в исходниках у некоторых программ.

`dpkg-deb --build` – более надежный вариант создания **пакета** в дистрибутиве **Debian**.

`rpmbuild` – программа для сборки **.rpm-пакета**.

Для создания **пакета** необходимо иметь готовый **спес-файл** (control для **.deb**), который описывает пакет, его зависимости, имя, описание и так далее.

Собранный пакет дальше устанавливается при помощи менеджера пакетов.



# Пример сборки пакета на Debian

1. Скачать исходные файлы приложения.
2. Создать файл, описывающий пакет  
спес для .rpm, control для .deb.
3. Выполнить `dpkg-deb --build <source_dir>`  
Удобнее всего это сделать, поднявшись на уровень выше в иерархии каталогов.
4. Установить получившийся пакет при помощи пакетного менеджера.

# Пример сборки пакета на Debian

## Пример control-файла для утилиты bashtop:

```
Package: bashtop
Version: 0.0.0
Section: base
Priority: optional
Architecture: all
Depends: bash (>= 4.4), curl (>= 7.16.2), coreutils, sed, awk, grep
Maintainer: your_name <xxx@xxx.xxx>
Description: Resource monitor that shows usage
             and stats for processor, memory, disks, network and processes
Homepage: https://github.com/aristocratos/bashtop
```



# Пакетный менеджер ART

---

# Пакетный менеджер APT

**APT** (advanced packaging tool) — программа для установки, обновления и удаления программных пакетов в операционных системах **Debian** и основанных на них (**Ubuntu**, **Linux Mint** и т. п.).

Способна автоматически устанавливать и настраивать программы для UNIX-подобных операционных систем как из **предварительно откомпилированных пакетов**, так и из **исходных кодов**.

---

# Синтаксис построения команды

Синтаксис команды стандартно включает в себя:

- обращение к менеджеру;
- действие;
- ключи (или, другое название, опции);
- цель или объект команды.

# Синтаксис построения команды

## Пример:

`apt install -y vim`

Где:

- `apt` – пакетный менеджер;
- `install` – выполняемое действие;
- `-y` – ключ команды;
- `vim` – цель этой команды.

---

# Обновление пакетов

## Обновление списка пакетов:

**apt update** — необходимо для получения последней информации о хранящихся в репозитории пакетах.

## Обновление пакетов:

**apt upgrade** — обновляет все установленные в системе пакеты до актуальных версий.

# Информация о пакетах

- `apt search <pattern>` – полнотекстовый поиск в репозитории по паттерну;
- `apt show <packet_name>` – показывает информацию о пакете (пакетах);
- `apt list` – выводит список пакетов в зависимости от ключа:
  - `--installed` – установленных;
  - `--upgradeable` – доступных к обновлению;
  - `--all-versions` – всех доступных.



# Установка и удаление пакета из репозитория

- `apt install <packet_name>` – устанавливает в систему пакет программного обеспечения, разрешая, по возможности, зависимости;
- `apt remove <packet_name>` – удаляет файлы приложения из системы, но оставляет пользовательские файлы с настройками;
- `apt purge <packet_name>` – удаляет *все* файлы приложения, включая файлы настроек;
- `apt reinstall <packet_name>` – переустанавливает пакет в системе.

---

## Полезные команды

- `apt autoremove` – удаляет неиспользуемые пакеты, например, старые версии ядра;
- `apt -f install` – попыбует починить сломанную установку пакета, например, неудовлетворенные зависимости.



# Пакетный менеджер YUM

---

# Пакетный менеджер YUM

**YUM** (Yellowdog Updater, Modified) — открытый консольный менеджер пакетов для дистрибутивов Linux, основанных на **пакетах формата RPM** (RedHat, CentOS, Fedora, Oracle Linux).

Как и APT, менеджер YUM работает с репозиториями пакетов от производителя дистрибутива или от сторонних авторов, также способен автоматически устанавливать и настраивать программы.

# Синтаксис построения команды

Синтаксис команды стандартно включает в себя:

- обращение к менеджеру;
- действие;
- ключи (или, другое название, опции);
- цель или объект команды.

**Пример:**

**yum install -y vim**

Где:

- **yum** – пакетный менеджер;
- **install** – выполняемое действие;
- **-y** – ключ команды;

# Синтаксис построения команды

## Пример:

`yum install -y vim`

Где:

- `yum` – пакетный менеджер;
- `install` – выполняемое действие;
- `-y` – ключ команды;
- `vim` – цель этой команды.

➡ Как видите, синтаксис очень похож на APT.

---

# Обновление пакетов

## Есть и отличия 😊

- `yum update` – проводит обновление всех пакетов в системе, перед этим выполняя обновление списка пакетов в репозитории;
- `yum update <packet_name>` – обновляет только выбранный пакет;
- `yum downgrade <packet_name>` – откатывает пакет к предыдущей версии.

# Информация о пакетах

- `yum search <pattern>` – полнотекстовый поиск в репозитории по паттерну;
- `yum list` – выводит список пакетов в зависимости от ключа:
  - `installed` – установленных;
  - `available` – всех доступных пакетов;
  - `all` – всех доступных и установленных пакетов.




## Установка и удаление пакета из репозитория

- `yum install <packet_name>` – устанавливает в систему пакет программного обеспечения, разрешая по возможности зависимости;
- `yum remove <packet_name>` – удаляет файлы приложения из системы, но оставляет пользовательские файлы с настройками;
- `yum reinstall <packet_name>` – переустанавливает приложение в системе.

---

## Полезные команды

- `yum autoremove` – идентичен применению в APT;
- `yum clean packages` – удалит пакеты из кеша.



# Прочие пакетные менеджеры

# Пакетный менеджер npm

**npm** (Node Package Manager) – пакетный менеджер для [Node.js](#), программной платформы языка JavaScript.

Используется в системах, работающих на [Node.js](#) и позволяет быстро оперировать пакетами этого фреймворка.

[npm](#) устанавливается вместе с [node](#).

➡ Чаще всего необходимости устанавливать его отдельно нет. Если же такая ситуация возникает, используйте:

```
curl https://npmjs.org/install.sh | sh
```



---

# Поиск и получение информации о пакете

- `npm search <pattern>` – поиск пакетов в базе данных репозитория.

Ищет как по названию, так и по описанию пакета.

- `npm view <pattern>` – просмотр информации о пакете.

# Установка и удаление пакетов

- `npm install <package_name>` – установка пакета локально;

Обратите внимание, в таком виде пакеты будут устанавливаться в `current work directory`, то есть в текущую рабочую директорию.

➡ Если вам надо установить пакет глобально, воспользуйтесь ключом `-g`.

- `npm uninstall <package_name>` – удаление локально установленного пакета.

Для удаления глобально установленного, воспользуйтесь ключом `-g`.

# Пакетный менеджер pip

**Pip** – пакетный менеджер языка Python, на нем же и написанный.

Версии Python, начиная с 2.7.9 и 3.4, содержат пакет pip по умолчанию\*.

Если же pip отсутствует, то его можно установить, скачав с официального сайта:

```
curl https://bootstrap.pypa.io/get-pip.py | python**
```

\* Или pip3 для Python 3, если одновременно установлен 2 и 3 Python.

\*\* Скрипт установки написан на Python, и для его исполнения нужен установленный в системе Python.



---

# Поиск и получение информации о пакете

- `pip search <pattern>` – поиск пакетов в базе данных репозитория.

Ищет как по названию, так и по описанию пакета.

- `pip list` – список установленных пакетов;
- `pip show <package_name>` – показывает информацию о пакете.



# Установка и удаление пакетов

- `pip install <package_name>` – установка выбранного пакета;
- `pip uninstall <package_name>` – удаление установленного пакета;

Пакеты устанавливаются глобально, для всей системы.

- `pip install -U` – обновление пакетов.

# Пакетный менеджер Gem

**Gem** – пакетный менеджер языка Ruby, созданный для упрощения процесса создания, распространения и установки библиотек.

С версии Ruby 1.9 RubyGems входит в стандартный пакет.

При необходимости установить его вручную:

- скачайте пакет по ссылке: [wget](https://rubygems.org/rubygems/rubygems-3.2.8.tgz)  
<https://rubygems.org/rubygems/rubygems-3.2.8.tgz>;
- распакуйте при помощи [tar](#);
- запустите установочный скрипт: [ruby setup.rb](#).

\* Скрипт написан на Ruby и для его выполнения в системе должен быть установлен Ruby.



# Поиск и получение информации о пакете

- `gem search -r <pattern>` – поиск пакетов в базе данных репозитория.

Ключ `-r` определяет, что поиск будет произведен в репозитории. Для поиска локально используйте ключ `-l`.

- `gem list` – список установленных пакетов.

# Установка и удаление пакетов

- `gem install <package_name>` – установка выбранного пакета;
- `gem uninstall <package_name>` – удаление установленного пакета;

Пакеты устанавливаются глобально, для всей системы.

- `gem update` – обновление пакетов.



# Итоги

---

# Итоги

## Сегодня мы узнали:

- что такое пакет;
- что такое официальные и сторонние репозитории;
- как работать со сторонними репозиториями;
- как пользоваться менеджерами пакетов;
- научились работать с исходными файлами;
- какие сторонние менеджеры пакетов еще есть для облегчения работы;

---

# Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте в чате учебной группы и/или в разделе “Вопросы по заданию”.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и  
пишите отзыв о лекции!**

**Александр Зубарев**