

# Руководство по сборке RPM-пакетов для дистрибутивов АЛЬТ

Валентин Соколов и другие.

# Оглавление

Введение в пакетные менеджеры .....	1
Вступление .....	2
PDF Версия .....	2
Файловая структура .....	2
Вклад в руководство .....	3
Система управления пакетами. Знакомство с APT .....	4
Установка необходимых пакетов для процесса сборки .....	5
Основные команды RPM .....	6
Программное обеспечение для упаковки .....	10
RPM-пакеты .....	10
Что такое RPM-пакет? .....	10
Инструменты для упаковки RPM .....	10
Рабочее пространство для упаковки RPM .....	10
Что такое SPEC-файл? .....	11

# Введение в пакетные менеджеры

**RPM** — это семейство пакетных менеджеров, применяемых в различных дистрибутивах GNU/Linux, в том числе и в проекте [Сизиф](#) и в дистрибутивах [Альт](#). Практически каждый крупный проект, использующий RPM, имеет свою версию пакетного менеджера, отличающуюся от остальных.

**Различия между представителями семейства RPM выражаются в:**

- наборе макросов, используемых в .спес-файлах,
- различном поведении RPM при сборке «по умолчанию» — при отсутствии каких-либо указаний в .спес-файлах,
- формате строк зависимостей,
- мелких отличиях в семантике операций (например, в операциях сравнения версий пакетов),
- мелких отличиях в формате файлов.

Для пользователя различия чаще всего заключаются в невозможности поставить «неродной» пакет из-за проблем с зависимостями или из-за формата пакета.

**RPM в проекте Сизиф также не является исключением. Основные особенности RPM в Альт и Сизиф от RPM других крупных проектов заключаются в следующем:**

- обширный набор макросов для упаковки различных типов пакетов,
- отличающееся поведение «по умолчанию» для уменьшения количества шаблонного кода в .спес-файлах,
- наличие механизмов для автоматического поиска межпакетных зависимостей,
- наличие так называемых set-version зависимостей (начиная с [4.0.4-alt98.46](#) ), обеспечивающих дополнительный контроль за изменением ABI библиотек,
- до [p8](#) и выпусков [8.x](#) включительно — очень древняя версия «базового» RPM (4.0.4), от которого началось развитие ветки RPM в Sisyphus (в Sisyphus и [p9](#) осуществлён частичный переход на [rpm 4.13](#)).

# Вступление

Руководство по сборке RPM пакетов:

## Как подготовить исходный код для сборки RPM.

Ссылка на раздел для тех, кто не имеет опыта разработки ПО. [\[preparing-software-for-packaging\]](#).

## Как собрать исходный код в RPM.

Ссылка на раздел для разработчиков ПО, которым необходимо собрать программное обеспечение в RPM пакеты. [Программное обеспечение для упаковки.](#)

## Расширенные сценарии сборки.

Эта ссылка на справочный материал для сборщиков RPM, работающих с расширенными сценариями сборки RPM. [\[advanced-topics\]](#).

# PDF Версия

Вы также можете скачать [PDF версию данного документа.](#)

# Файловая структура

Перед тем, как приступить к сборке, нужно создать структуру каталогов, необходимую RPM, находящуюся в Вашем «домашнем» каталоге:

- Вывод команд и содержимое текстовых файлов, включая исходный код, размещаются в каталогах:

```
$ tree ~/RPM/  
/home/user/RPM/  
|-- BUILD  
|-- BUILDROOT  
|-- RPMS  
|   |-- i586  
|   |-- x86_64  
|   `-- noarch  
|-- SOURCES  
|-- SPECS  
`-- SRPMS
```

```
Name:      bello  
Version:  
Release:   alt1  
Summary:
```

- Темы, представляющие интерес, или словарные термины упоминаются либо как ссылки

на соответствующую документацию или веб-сайт выделены **жирным** шрифтом, либо *курсивом*. Первые упоминания некоторых терминов ссылаются на соответствующую документацию.

- Названия утилит, команд и других элементов, обычно встречающихся в коде, написаны **моноширинным** шрифтом.

## Вклад в руководство

Вы можете внести свой вклад в это руководство, отправив сообщение о проблеме или pull request на [GitHub репозиторий](#).

# Система управления пакетами.

## Знакомство с АРТ

Для установки, удаления и обновления программ и поддержания целостности системы в Linux в первую очередь стали использоваться *менеджеры пакетов* (такие, как RPM в дистрибутивах RedHat или dpkg в Debian GNU/Linux). С точки зрения менеджера пакетов программное обеспечение представляет собой набор компонентов — программных *пакетов*. Такие компоненты содержат в себе набор исполняемых программ и вспомогательных файлов, необходимых для корректной работы ПО. Менеджеры пакетов дают возможность унифицировать и автоматизировать сборку бинарных пакетов и облегчают установку программ, позволяя проверять наличие необходимых для работы устанавливаемой программы компонент подходящей версии непосредственно в момент установки, а также производя все необходимые процедуры для регистрации программы во всех операционных средах пользователя. Сразу после установки программа оказывается доступна пользователю из командной строки и появляется в меню всех графических оболочек.

Полное описание АРТ можно узнать, перейдя по ссылке: [Установка и удаление программ \(пакетов\)](#)

### ПРИМЕЧАНИЕ

Установка пакетов в АЛЪТ Линукс осуществляется с помощью утилиты АРТ

### ПРИМЕЧАНИЕ

Для сокращения команд, встречающихся в тексте, будет использоваться нотация:

- - команды без административных привелегий будут начинаться с символа “\$”
- - команды без административных привелегий будут начинаться с символа “\$”

### ПРИМЕЧАНИЕ

По умолчанию **sudo** может быть отключено. Для получения административных привелегий используется команда **su**. Для включения **sudo** в стандартном режиме можно использовать команду:

```
# control sudowheel enabled
```

# Установка необходимых пакетов для процесса сборки

Чтобы следовать данному руководству, Вам потребуется установить следующие пакеты:

## ПРИМЕЧАНИЕ

Некоторые из этих пакетов устанавливаются по умолчанию в [Альт](#). Установка проводится с правами суперпользователя.

```
# apt-get update
```

```
# apt-get install gcc rpm-build rpmlint make python gear hasher patch
```

# Основные команды RPM

Для ознакомления с данным разделом потребуется скачать пакет [Yodl-docs](#).

## Как узнать информацию об RPM-пакете без установки?

После скачивания пакета можно посмотреть данные о нём перед установкой. Для этого используется **-qip**, чтобы вывести информацию о пакете.

```
$ rpm -qip yodl-docs-4.03.00-alt2.noarch.rpm
```

Вывод:

```
Name       : yodl-docs
Epoch     : 1
Version    : 4.03.00
Release    : alt2
DistTag    : sisyphus+271589.100.1.2
Architecture: noarch
Install Date: (not installed)
Group      : Documentation
Size       : 3701571
License    : GPL
Signature  : DSA/SHA1, Чт 13 мая 2021 05:44:49, Key ID 95c584d5ae4ae412
Source RPM : yodl-4.03.00-alt2.src.rpm
Build Date : Чт 13 мая 2021 05:44:44
Build Host : darktemplar-sisyphus.hasher.altlinux.org
Relocations : (not relocatable)
Packager   : Aleksei Nikiforov <darktemplar@altlinux.org>
Vendor     : ALT Linux Team
URL        : https://gitlab.com/fbb-git/yodl
Summary    : Documentation for Yodl
Description :
Yodl is a package that implements a pre-document language and tools to
process it. The idea of Yodl is that you write up a document in a
pre-language, then use the tools (eg. yodl2html) to convert it to some
final document language. Current converters are for HTML, ms, man, LaTeX
SGML and texinfo, plus a poor-man's text converter. Main document types
are "article", "report", "book" and "manpage". The Yodl document
language is designed to be easy to use and extensible.
```

This package contains documentation for Yodl.

## Как установить RPM-пакет?

Для установки используется параметр **-ivh**.

```
$ rpm -ivh yodl-docs-4.03.00-alt2.noarch.rpm
```



Вывод:

```
Подготовка...
##### [100%]
Обновление / установка...
1: yodl-docs-1:4.03.00-alt2
##### [100%]
Running /usr/lib/rpm/posttrans-filetriggers
```

**Проверка установки пакета в системе.**

```
$ rpm -q yodl-docs
```

Вывод:

```
yodl-docs-4.03.00-alt2.noarch
```

**Просмотр файлов пакета, установленного в системе.**

```
$ rpm -ql yodl-docs
```

Вывод:

```
/usr/share/doc/yodl
/usr/share/doc/yodl-doc
/usr/share/doc/yodl-doc/AUTHORS.txt
/usr/share/doc/yodl-doc/CHANGES
/usr/share/doc/yodl-doc/changelog
/usr/share/doc/yodl-doc/yodl.dvi
/usr/share/doc/yodl-doc/yodl.html
/usr/share/doc/yodl-doc/yodl.latex
/usr/share/doc/yodl-doc/yodl.pdf
/usr/share/doc/yodl-doc/yodl.ps
/usr/share/doc/yodl-doc/yodl.txt
/usr/share/doc/yodl-doc/yodl01.html
/usr/share/doc/yodl-doc/yodl02.html
/usr/share/doc/yodl-doc/yodl03.html
/usr/share/doc/yodl-doc/yodl04.html
/usr/share/doc/yodl-doc/yodl05.html
/usr/share/doc/yodl-doc/yodl06.html
/usr/share/doc/yodl/AUTHORS.txt
/usr/share/doc/yodl/CHANGES
/usr/share/doc/yodl/changelog
```

**Просмотр недавно установленных пакетов.**

```
rpm -qa --last|head
```

Вывод:

```
yodl-docs-4.03.00-alt2.noarch          Чт 22 дек 2022 18:09:10
source-highlight-3.1.9-alt1.git.904949c.x86_64 Вт 20 дек 2022 18:38:29
libsource-highlight-3.1.9-alt1.git.904949c.x86_64 Вт 20 дек 2022 18:38:29
gem-asciidoctor-doc-2.0.10-alt1.noarch  Вт 20 дек 2022 18:34:04
w3m-0.5.3-alt4.git20200502.x86_64     Вт 20 дек 2022 18:23:05
sgml-common-0.6.3-alt15.noarch         Вт 20 дек 2022 18:23:05
libmaa-1.4.7-alt4.x86_64              Вт 20 дек 2022 18:23:05
docbook-style-xsl-1.79.1-alt4.noarch    Вт 20 дек 2022 18:23:05
docbook-dtds-4.5-alt1.noarch           Вт 20 дек 2022 18:23:05
dict-1.12.1-alt4.1.x86_64             Вт 20 дек 2022 18:23:05
```

### Поиск пакета в системе.

Команда **grep** поможет определить, установлен пакет в системе или нет:

```
$ rpm -qa | grep yodl-docs
```

Вывод:

```
yodl-docs-4.03.00-alt2.noarch
```

### Проверка файла, относящегося к пакету.

Предположим, что нужно узнать, к какому конкретному пакету относится файл. Для этого используют команду:

```
$ rpm -qf /usr/share/doc/yodl-doc
```

Вывод:

```
yodl-docs-4.03.00-alt2.noarch
```

### Вывод информации о пакете.

Чтобы получить информацию о пакете, установленном в систему, используем команду:

```
$ rpm -qi yodl-docs
```

Вывод:

```

Name       : yodl-docs
Epoch     : 1
Version    : 4.03.00
Release    : alt2
DistTag    : sisyphus+271589.100.1.2
Architecture: noarch
Install Date: Чт 22 дек 2022 18:09:10
Group      : Documentation
Size       : 3701571
License    : GPL
Signature  : DSA/SHA1, Чт 13 мая 2021 05:44:49, Key ID 95c584d5ae4ae412
Source RPM : yodl-4.03.00-alt2.src.rpm
Build Date : Чт 13 мая 2021 05:44:44
Build Host : darktemplar-sisyphus.hasher.altlinux.org
Relocations : (not relocatable)
Packager   : Aleksei Nikiforov <darktemplar@altlinux.org>
Vendor     : ALT Linux Team
URL        : https://gitlab.com/fbb-git/yodl
Summary    : Documentation for Yodl
Description :
Yodl is a package that implements a pre-document language and tools to
process it. The idea of Yodl is that you write up a document in a
pre-language, then use the tools (eg. yodl2html) to convert it to some
final document language. Current converters are for HTML, ms, man, LaTeX
SGML and texinfo, plus a poor-man's text converter. Main document types
are "article", "report", "book" and "manpage". The Yodl document
language is designed to be easy to use and extensible.

```

### Обновление пакета.

Для обновления пакета используется параметр **-Uvh**.

```
$ rpm -Uvh yodl-docs-4.03.00-alt2.noarch.rpm
```

Вывод:

```

Подготовка...
##### [100%]
пакет yodl-docs-1:4.03.00-alt2.noarch уже установлен

```

# Программное обеспечение для упаковки

## RPM-пакеты

В этом разделе рассматриваются основы сборки RPM-пакетов. Дополнительные сведения смотри в разделе [Дополнительные материалы](#).

### Что такое RPM-пакет?

RPM-пакет - это архив, содержащий в себе архив [.cpio](#) с файлами (скомпилированные исполняемые файлы, библиотеки и данные), а также метаданные - имя пакета, его описание, зависимости и т.д. Менеджер пакетов RPM использует эти метаданные для проверки наличия необходимых пакетов из списка зависимостей, исполнения инструкций по установке файлов и сохранения общей информации о пакете у себя в базе.

Существует два типа RPM-пакетов:

- SRPM-пакеты (исходники) - архив с расширением [.src.rpm](#)
- RPM-пакеты (бинарники) - архив с расширением [.rpm](#)

SRPM и RPM-пакеты имеют общий формат и инструментарий, но имеют разное содержимое и служат разным целям. SRPM содержит исходный код, при необходимости патчи к нему и спес-файл, в котором описывается, как собрать исходный код в бинарный RPM-пакет. Бинарный RPM-пакет содержит бинарные файлы, созданные из исходных текстов и патчей, если таковые имелись.

## Инструменты для упаковки RPM

### Рабочее пространство для упаковки RPM

Чтобы создать дерево каталогов, которое является рабочей областью упаковки RPM, используйте утилиту [rpmdev-setuptree](#):

```
$ rpmdev-setuptree

$ tree ~/rpmbuild/
/home/user/rpmbuild/
|-- BUILD
|-- RPMS
|-- SOURCES
|-- SPECS
-- SRPMS
```

Созданные каталоги служат следующим целям:

Каталог	Назначение
---------	------------

BUILD	Содержит все файлы, которые появляются при сборке пакета.
RPMS	Здесь появляются готовые бинарные RPM-пакеты ( <b>.rpm</b> ), в подкаталогах для разных архитектур, например, в подкаталогах <b>x86_64</b> и <b>noarch</b> .
SOURCES	Здесь находятся архивы исходного кода и патчи. Утилита <b>rpmbuild</b> ищет их здесь.
SPECS	Здесь хранятся спес-файлы.
SRPMS	Здесь находятся пакеты с исходниками ( <b>.src.rpm</b> ).

## Что такое SPEC-файл?

Спес-файл можно рассматривать как "инструкцию", который утилита **rpmbuild** использует для фактической сборки RPM-пакета. Он сообщает системе сборки, что делать, определяя инструкции в серии разделов. Разделы определены в *Преамбуле* и в *Основной части*. *Преамбула* содержит ряд элементов метаданных, которые используются в *Основной части*. Тело содержит основную часть инструкций.

### Пункты преамбулы

В этой таблице перечислены элементы, используемые в разделе преамбулы файла спецификации RPM:

SPEC Директива	Определение
<b>Name</b>	Базовое имя пакета, которое должно совпадать с именем спес-файла.
<b>Version</b>	Версия upstream-кода.
<b>Release</b>	Релиз пакета используется для указания номера сборки пакета при данной версии upstream-кода. Как правило, установите начальное значение равным 1%{?dist} и увеличивайте его с каждым новым выпуском пакета. Сбросьте значение до 1 при создании новой версии программного обеспечения.
<b>Summary</b>	Краткое, в одну строку, описание пакета.
<b>License</b>	Лицензия на упаковываемое программное обеспечение.
<b>URL</b>	Полный URL-адрес для получения дополнительной информации о программе. Чаще всего это веб-сайт upstream-проекта для собираемого программного обеспечения.
<b>Source0</b>	Путь или URL-адрес к сжатому архиву исходного кода (не исправленный, исправления обрабатываются в другом месте). Этот раздел должен указывать на доступное и надежное хранилище архива, например, на upstream-страницу, а не на локальное хранилище сборщика. При необходимости можно добавить дополнительные исходные директивы, каждый раз увеличивая их количество, например: Source1, Source2, Source3 и так далее.

Patch0	Название первого патча, который при необходимости будет применен к исходному коду. При необходимости можно добавить дополнительные директивы PatchX, увеличивая их количество каждый раз, например: Patch1, Patch2, Patch3 и так далее.
BuildArch	Если пакет не зависит от архитектуры, например, если он полностью написан на интерпретируемом языке программирования, установите для этого значение BuildArch: noarch. Если этот параметр не задан, пакет автоматически наследует архитектуру компьютера, на котором он собран, например x86_64.
BuildRequires	Разделённый запятыми или пробелами список пакетов, необходимых для сборки программы, написанной на скомпилированном языке. Может быть несколько записей BuildRequires, каждая в отдельной строке в SPEC файле.
Requires	Разделённый запятыми или пробелами список пакетов, необходимых программному обеспечению для запуска после установки. Это его зависимости. Может быть несколько записей Requires, каждая в отдельной строке в SPEC файле.
ExcludeArch	Если часть программного обеспечения не может работать на определенной архитектуре процессора, Вы можете исключить эту архитектуру здесь.

Директивы Name, Version и Release содержат имя RPM-пакета. Эти три директивы часто называют N-V-R или NVR, поскольку имена RPM-пакета имеют формат NAME-VERSION-RELEASE.

Вы можете получить пример NAME-VERSION-RELEASE, выполнив запрос с использованием rpm для конкретного пакета:

```
$ rpm -q rpmdevtools
rpmdevtools-8.10-alt2.noarch
```

Здесь rpmdevtools - это имя пакета, 8.10 - версия, а alt2 - релиз. Последний маркер noarch - сведения об архитектуре. В отличие от NVR, маркер архитектуры не находится под прямым управлением сборщика, а определяется средой сборки rpmbuild. Исключением из этого правила является архитектурно-независимый пакет noarch.

### Составляющие основной части

В этой таблице перечислены элементы, используемые в разделе Body (Тело, основная часть) файла спецификации RPM:

SPEC Директива	Определение
----------------	-------------

<code>%description</code>	Полное описание программного обеспечения, входящего в комплект поставки RPM. Это описание может занимать несколько строк и может быть разбито на абзацы.
<code>%prep</code>	Команда или серия команд для подготовки программного обеспечения к сборке, например, распаковка архива в Source0. Эта директива может содержать сценарий оболочки.
<code>%build</code>	Команда или серия команд для фактической сборки программного обеспечения в машинный код (для скомпилированных языков) или байт-код (для некоторых интерпретируемых языков).
<code>%install</code>	Команда или серия команд для копирования требуемых артефактов сборки из <code>%builddir</code> (где происходит сборка) в <code>%buildroot</code> каталог (который содержит структуру каталогов с файлами, подлежащими упаковке). Обычно это означает копирование файлов из <code>~/rpmbuild/BUILD</code> в <code>~/rpmbuild/BUILDROOT</code> и создание необходимых каталогов <code>~/rpmbuild/BUILDROOT</code> . Это выполняется только при создании пакета, а не при установке пакета конечным пользователем. Подробности см. в разделе <a href="#">Работа со SPEC файлом</a> .
<code>%check</code>	Команда или серия команд для тестирования программного обеспечения. Обычно включает в себя такие вещи, как модульные тесты.
<code>%files</code>	Список файлов, которые будут установлены в системе конечного пользователя.
<code>%changelog</code>	Запись изменений, произошедших с пакетом между различными <code>Version</code> или <code>Release</code> сборками.