Содержание

пакет ПО

Пакет ПО - это архив, содержащий ПО (файлы, конфиги, библиотеки и тп) и инструкции/метаданные для его установки, обновления и удаления. Распространены несколько форматов пакетов: DEB и RPM.

Бинарный пакет ПО - пакет, содержащий не исхдные коды и инструкии для их сборки, а уже готовый билд.

Исходные пакеты (*src-пакет*) - пакет, содержащий исходные коды и инструкции по сборке и установке программы.

Метапакеты - пакеты, которые сами по себе не содержат программ, но служат для установки других пакетов или зависимостей.

Помимо классических пакетов ПО есть более современные технологии упаковки ПО:

- **Flatpak**: Позволяет запускать приложения в изолированных контейнерах, что обеспечивает большую безопасность и совместимость между различными дистрибутивами.
- **Snap**: Разработан компанией Canonical, Snap также использует контейнеризацию для обеспечения изоляции приложений и их зависимостей, что позволяет легко устанавливать и обновлять программы.
- **AppImage**: Это формат, который позволяет запускать приложения без необходимости установки. AppImage упаковывает все необходимые зависимости в один файл, что делает его переносимым и удобным для использования на различных системах.

О назначениях директорий

(в контексте префиксов (PREFIX))

(в соответствии с «Filesystem Hierarchy Standard» (см `man 7 hier`))

- `/bin` — это бинарники (т. е. испоняемые файлы), важные для работы системы на ранних стадиях загрузки.
- `/sbin` — это то же самое с тем отличием, что эти бинарники обычно может запускать только root.
- `/lib` — бинарные файлы библиотек, нужные для программ из `/bin` и `/sbin`
- `/usr` — «*вторая иерархия файлов*», т. е. это как бы «ещё один `/`»
- `/usr/bin` — то же, что `/bin`, но на этот раз это бинарники, некритичные для загрузки
- `/usr/sbin` — то же, что `/sbin`, но опять-таки, некритичные для загрузки
- `/usr/lib` — бинарные файлы библиотек, которые не нужны для программ из `/bin` и `/sbin`
- `/usr/include` — хедеры библиотек
- `/usr/local` — «*третья иерархия файлов*», она опять содержит `/usr/local/bin`, `/usr/local/sbin`, `/usr/local/lib` и `/usr/local/include`, на этот раз эти каталоги предназначены для «локального администратора». Что это значит? Это значит, что (в случае дистрибутивов GNU/Linux с пакетным менеджером) каталоги, о которых речь шла до этого, находятся под контролем пакетного менеджера. А вот каталоги внутри

`/usr/local` находятся в распоряжении «локального администратора», т. е. вас. Если вы хотите поставить что-то из соурсов, в обход менеджера пакетов, то можно поставить туда.

Подробнее об иерархиях файлов (префиксах) далее.

Подробнее о прификсах

Префикс `/`
Вряд ли когда-нибудь вам придётся его выбирать. Он используется для программ, критичных для ранних стадий загрузки ОС (т. е. критичные элементы для загрузки находятся в `/bin`, `/lib` и т. д.) (впрочем, даже если вам нужно установить программу в `/`, её сперва устанавливают в /usr, т. е. собирают и устанавливают с префиксом `/usr`, а потом перемещают необходимое в `/` \[т. е. перемещают из /usr/bin в /bin, скажем], во всяком случае именно так поступают авторы Linux From Scratch 7.10 с пакетом, скажем, bash).

Префикс `/usr`
Стандартный префикс, используемый обычно для программ, установленных через менеджер пакетов. То есть если вы установили программу через менеджер пакетов, она ведёт себя так, словно она собрана и установлена на вашей системе с префиксом `/usr`. Самому устанавливать пакеты с префиксом `/usr` нельзя.

Префикс `/usr/local`
Отличный префикс для установки туда программ самостоятельно. Хорош тем, что `/usr/local/bin` есть в дефолтном PATH (во всяком случае в дебиане). То есть сразу после установки программы вы сможете просто запускать программу по названию. Потому что бинарник лежит в `/usr/local/bin`, а `/usr/local/bin` есть в PATH.

Префиксы, соответствующие домашним каталогам (`/home/username`)
Советую в случае, когда хочется поставить «только для себя», т. е. только для одного юзера. Или когда нет прав root.

При любой установке пакетов в один префикс (что почти неизбежно), там все программы лежат смешанно. Вот допустим, вы установили библиотеку «foo», а потом библиотеку «bar». Обе в этот prefix. Тогда дерево может выглядеть так (в совсем упрощённом виде):

```
/usr/local/include/foo.h
/usr/local/include/bar.h
/usr/local/lib/libfoo.so
/usr/local/lib/libbar.so
```

Всё смешано. Нет единой папки, которая бы содержала «всё, связанное с foo» и другой папки, которая бы содержала «всё, связанное с bar». (GoboLinux решает эту проблему)

Префиксы вида `/opt/XXX`. Папку /opt предполагается использовать следующим образом: в ней нужно создавать подкаталоги, называть их названиями пакетов и использовать эти подкаталоги как префиксы. При таком подходе указанная выше проблема сваливания в кучу (если считать это проблемой) исчезает. Каждый пакет будет установлен в свой каталог. Приведённый выше пример с «foo» и «bar» будет выглядеть так (я бы посоветовал в названии подкаталогов в `/opt` указывать ещё и номер версии):

```
/opt/foo-1.0/include/foo.h
/opt/foo-1.0/lib/libfoo.so
/opt/bar-2.0/include/bar.h
/opt/bar-2.0/lib/libbar.so
```

Недостаток у такого решения тоже есть. Вам придётся самому добавлять все эти бесчисленные каталоги `/opt/foo-1.0/bin` (для каждого пакета) в PATH