

# Руководство по созданию пакетов Debian

Лукас Нуссбаум  
packaging-tutorial@packages.debian.org

version 0.24 – 2019-03-13



# Об этом руководстве

- ▶ Цель: сообщить вам то, что вам действительно нужно знать о создании пакетов Debian
  - ▶ Изменение существующих пакетов
  - ▶ Создание ваших собственных пакетов
  - ▶ Взаимодействие с сообществом Debian
  - ▶ Становимся продвинутым пользователем Debian
- ▶ Освещает большинство важных моментов, но не полно
  - ▶ Вам будет необходимо прочитать дополнительную документацию
- ▶ Большая часть изложения применима к производным от Debian дистрибутивам
  - ▶ Включая и Ubuntu



# План

- 1 Введение
- 2 Создание пакетов с исходным кодом
- 3 Сборка и тестирование пакетов
- 4 Практика 1: изменение пакета grep
- 5 Продвинутое темы в сборке пакетов
- 6 Сопровождение пакетов в Debian
- 7 Заключение
- 8 Дополнительные практические задания
- 9 Ответы на практические задания



# План

- 1 Введение
- 2 Создание пакетов с исходным кодом
- 3 Сборка и тестирование пакетов
- 4 Практика 1: изменение пакета `grep`
- 5 Продвинутые темы в сборке пакетов
- 6 Сопровождение пакетов в Debian
- 7 Заключение
- 8 Дополнительные практические задания
- 9 Ответы на практические задания



# Debian

- ▶ Дистрибутив GNU/Linux
- ▶ Первый крупный дистрибутив, разрабатываемый “открыто в духе GNU”
- ▶ Некоммерческий, создан совместно более чем 1,000 добровольцами
- ▶ 3 основных особенности
  - ▶ Качество – культура технического совершенства  
Мы выпускаем очередную версию, только когда она готова
  - ▶ Свобода – разработчики и пользователи связаны Общественным договором  
Продвигаем культуру Свободного ПО с 1993 года
  - ▶ Независимость – нет (ни одной) компаний, опекающих Debian  
Открытый процесс принятия решений (управление тех, кто делает + демократия)
- ▶ Любительский дистрибутив в лучшем смысле этого слова: создаётся ради него самого



# Пакеты Debian

- ▶ Файлы .deb (двоичные пакеты)
- ▶ Очень мощный и удобный способ распространения ПО пользователям
- ▶ Один из двух наиболее распространённых форматов пакетов (наряду с RPM)
- ▶ Универсален
  - ▶ 30,000 двоичных пакетов в Debian  
→ для большей части доступного свободного ПО создан пакет Debian!
  - ▶ 12 переносов (архитектур), включая переносы на отличное от Linux ядро (Hurd; kFreeBSD)
  - ▶ Используется 120 ответвлениями от дистрибутива Debian



# Формат пакетов deb

- ▶ Файл .deb: архив ar

```
$ ar tv wget_1.12-2.1_i386.deb
rw-r--r-- 0/0      4 Sep  5 15:43 2010 debian-binary
rw-r--r-- 0/0    2403 Sep  5 15:43 2010 control.tar.gz
rw-r--r-- 0/0  751613 Sep  5 15:43 2010 data.tar.gz
```

- ▶ debian-binary: версия формата файла deb, "2.0\n"
  - ▶ control.tar.gz: метаданные о пакете control, md5sums, (pre|post)(rm|inst), triggers, shlibs, ...
  - ▶ data.tar.gz: файлы данных пакета
- ▶ Можно создавать файлы .deb вручную  
[http://tldp.org/HOWTO/html\\_single/Debian-Binary-Package-Building-HOWTO/](http://tldp.org/HOWTO/html_single/Debian-Binary-Package-Building-HOWTO/)
- ▶ Но большинство пользователей этого не делают

Настоящее руководство: создание пакетов Debian способом Debian



## Инструменты, которые вам потребуются

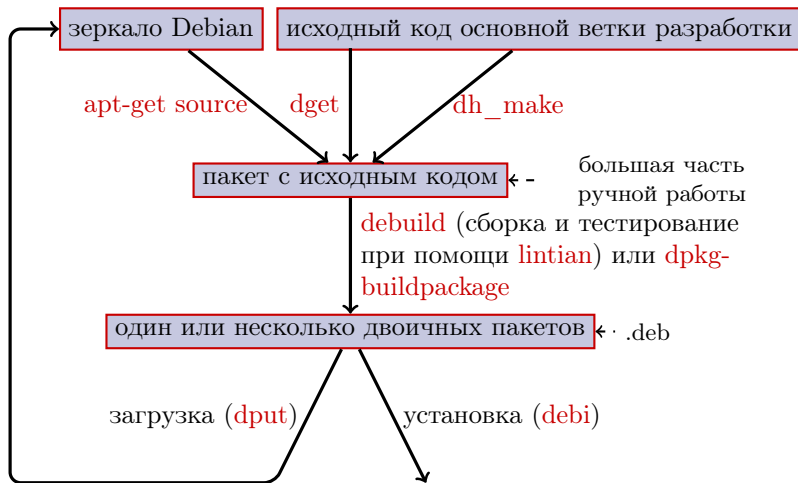
- ▶ Доступ к системе Debian (или Ubuntu) с правами суперпользователя
- ▶ Некоторые пакеты:
  - ▶ `build-essential`: имеет зависимости от пакетов, которые, как это предполагается, доступны на машине разработчика (не нужно указывать их в поле `Build-Depends`: управляющего файла вашего пакета)
    - ▶ включает зависимость от `dpkg-dev`, который содержит базовые инструменты Debian для создания пакетов
  - ▶ `devscripts`: содержит множество полезных сценариев для сопровождающих Debian

Множество других инструментов, которые также будут упомянуты в дальнейшем, такие как `debhelper`, `cdbs`, `quilt`, `pbuilder`, `sbuid`, `lintian`, `svn-buildpackage`, `git-buildpackage`, ...  
Установите их, если они вам нужны.





# Общая структура работы над созданием пакета



## Пример: пересборка dash

- 1 Установите пакеты, необходимые для сборки dash, а также devscripts  
`sudo apt-get build-dep dash`  
(требуются строки deb-src в /etc/apt/sources.list)  
`sudo apt-get install --no-install-recommends devscripts fakeroot`
- 2 Создайте рабочий каталог, и перейдите в него:  
`mkdir /tmp/debian-tutorial ; cd /tmp/debian-tutorial`
- 3 Загрузите пакет с исходным кодом dash  
`apt-get source dash`  
(Для этого требуются строки deb-src в вашем файле /etc/apt/sources.list)
- 4 Соберите пакет  
`cd dash-*`  
`debuild -us -uc` (-us -uc отключают подписывание пакета с помощью GPG)
- 5 Проверьте, что всё работает
  - ▶ В родительском каталоге присутствуют новые файлы .deb
- 6 Посмотрите каталог debian/
  - ▶ Здесь и осуществляется работа по созданию пакетов



# План

- 1 Введение
- 2 Создание пакетов с исходным кодом
- 3 Сборка и тестирование пакетов
- 4 Практика 1: изменение пакета grep
- 5 Продвинутое темы в сборке пакетов
- 6 Сопровождение пакетов в Debian
- 7 Заключение
- 8 Дополнительные практические задания
- 9 Ответы на практические задания



## Пакет с исходным кодом

- ▶ Один пакет с исходным кодом может создавать несколько двоичных пакетов  
напр., исходный код `libtar` создаёт двоичные пакеты `libtar0` и `libtar-dev`
- ▶ Два вида пакетов: (если не уверены, используйте `non-native`)
  - ▶ Пакеты `native`: обычно это ПО, специфичное для Debian (`dpkg`, `apt`)
  - ▶ Пакеты `non-native`: ПО, разрабатываемое за пределами Debian
- ▶ Основной файл: `.dsc` (метаданные)
- ▶ Другие файлы, зависящие от версии формата исходного кода
  - ▶ 1.0 или 3.0 (`native`): `package_version.tar.gz`
  - ▶ 1.0 (`non-native`):
    - ▶ `pkg_ver.orig.tar.gz`: исходный код основной ветки разработки
    - ▶ `pkg_debver.diff.gz`: заплатка для добавления специфичных для Debian изменений
  - ▶ 3.0 (`quilt`):
    - ▶ `pkg_ver.orig.tar.gz`: исходный код основной ветки разработки
    - ▶ `pkg_debver.debian.tar.gz`: tarball с изменениями Debian

(Детали см. в `dpkg-source(1)`)



## Пример пакета с исходным кодом (wget\_1.12-2.1.dsc)

Format: 3.0 (quilt)

Source: wget

Binary: wget

Architecture: any

Version: 1.12-2.1

Maintainer: Noel Kothe <noel@debian.org>

Homepage: <http://www.gnu.org/software/wget/>

Standards-Version: 3.8.4

Build-Depends: debhelper (>> 5.0.0), gettext, texinfo,  
libssl-dev (>= 0.9.8), dpatch, info2man

Checksums-Sha1:

50d4ed2441e67 [..] 1 ee0e94248 2464747 wget\_1.12.orig.tar.gz  
d4c1c8bbe431d [..] dd7cef3611 48308 wget\_1.12-2.1.debian.tar.gz

Checksums-Sha256:

7578ed0974e12 [..] dcba65b572 2464747 wget\_1.12.orig.tar.gz  
1e9b0c4c00eae [..] 89c402ad78 48308 wget\_1.12-2.1.debian.tar.gz

Files:

141461b9c04e4 [..] 9 d1f2abf83 2464747 wget\_1.12.orig.tar.gz  
e93123c934e3c [..] 2 f380278c2 48308 wget\_1.12-2.1.debian.tar.gz

## Получение существующего пакета с исходным кодом

---

- ▶ Из архива Debian:
  - ▶ `apt-get source пакет`
  - ▶ `apt-get source пакет=версия`
  - ▶ `apt-get source пакет/выпуск`(Требуются строки `deb-src` в `sources.list`)
- ▶ Из Интернет:
  - ▶ `dget адрес.dsc`
  - ▶ `dget http://snapshot.debian.org/archive/debian-archive/20090802T004153Z/debian/dists/bo/main/source/web/wget_1.4.4-6.dsc`  
(`snapshot.d.o` предоставляет все пакеты из Debian с 2005 года)
- ▶ Из (объявленной) системы контроля версий:
  - ▶ `debcheckout пакет`
- ▶ Когда пакет будет загружен, разверните его с помощью `dpkg-source -x файл.dsc`



## Создание простого пакета с исходным кодом

- ▶ Загрузите исходный код основной ветки разработки (исходный код основной ветки разработки = исходный код от исходных разработчиков ПО)
- ▶ Переименуйте его в `<пакет_с_исходным_кодом>_<версия_основной_ветки>.orig.tar.gz` (пример: `simgrid_3.6.orig.tar.gz`)
- ▶ Разверните его
- ▶ Переименуйте каталог в `<пакет_с_исходным_кодом>-<версия_основной_ветки>` (пример: `simgrid-3.6`)
- ▶ `cd <пакет_с_исходным_кодом>-<версия_основной_ветки> && dh_make` (из пакета `dh-make`)
- ▶ Для `dh_make` имеются некоторые альтернативы для конкретных наборов пакетов: `dh-make-perl`, `dh-make-php`, ...
- ▶ Будет создан каталог `debian/` с множеством файлов в нём



## Файлы в debian/

Вся работа по созданию пакетов должна осуществляться путём изменения файлов в debian/

- ▶ Основные файлы:
  - ▶ control – метаданные о пакете (зависимости и т.д.)
  - ▶ rules – определяет то, как собирать пакет
  - ▶ copyright – информация об авторских правах для данного пакета
  - ▶ changelog – история пакета Debian
- ▶ Другие файлы:
  - ▶ compat
  - ▶ watch
  - ▶ цели dh\_install\*  
\*.dirs, \*.docs, \*.manpages, ...
  - ▶ сценарии сопровождающего  
\*.postinst, \*.prerm, ...
  - ▶ source/format
  - ▶ patches/ – если вам нужно изменить исходный код основной ветки
- ▶ Некоторые файлы используют формат на основе RFC 822 (почтовые заголовки)



## debian/changelog

- ▶ Содержит список изменений пакета Debian
- ▶ Содержит текущую версию пакета

1.2.1.1-5  
└──────────┘└───┘  
Основная Debian-  
версия ревизия

- ▶ Редактируется вручную или с помощью **dch**
  - ▶ Создать запись об изменении для нового выпуска: **dch -i**
- ▶ Специальный формат для автоматического закрытия ошибок Debian или Ubuntu  
Debian: Closes: #595268; Ubuntu: LP: #616929
- ▶ Устанавливается как /usr/share/doc/package/changelog.Debian.gz

---

```
mpich2 (1.2.1.1 -5) unstable; urgency=low
```

- \* Use /usr/bin/python instead of /usr/bin/python2.5. Allow to drop dependency on python2.5. Closes: #595268
  - \* Make /usr/bin/mpdroot setuid. This is the default after the installation of mpich2 from source, too. LP: #616929
- + Add corresponding lintian override.

```
-- Lucas Nussbaum <lucas@debian.org> Wed, 15 Sep 2010 18:13:44 +0200
```

## debian/control

- ▶ Метаданные пакета
  - ▶ Для самого пакета с исходным кодом
  - ▶ Для каждого двоичного пакета, собираемого из исходного кода
- ▶ Имя, раздел, приоритет, сопровождающий, загружающие, сборочные зависимости, зависимости, описание, домашняя страница, ...
- ▶ Документация: Политика Debian, глава 5  
<https://www.debian.org/doc/debian-policy/ch-controlfields>

---

Source: wget  
Section: web  
Priority: important  
Maintainer: Noel Kothe <noel@debian.org>  
Build-Depends: debhelper (>> 5.0.0), gettext, texinfo,  
libssl-dev (>= 0.9.8), dpatch, info2man  
Standards-Version: 3.8.4  
Homepage: <http://www.gnu.org/software/wget/>

Package: wget  
Architecture: any  
Depends: \${shlibs:Depends}, \${misc:Depends}  
Description: retrieves files from the web  
Wget is a network utility to retrieve files from the Web



# Архитектура: все или какие-то

Два типа двоичных пакетов

- ▶ Пакеты с разным содержимым на каждой архитектуре Debian
  - ▶ Пример: программа на C
  - ▶ Architecture: any в debian/control
    - ▶ Либо, если она работает только на некоторых архитектурах:  
Architecture: amd64 i386 ia64 hurd-i386
  - ▶ [buildd.debian.org](http://buildd.debian.org): собирает для вас все остальные архитектуры при загрузке
  - ▶ Имеет имя пакет\_версия\_архитектура.deb
- ▶ Пакеты с одним и тем же содержимым на всех архитектурах
  - ▶ Пример: библиотека Perl
  - ▶ Architecture: all в debian/control
  - ▶ Имеет имя пакет\_версия\_all.deb

Пакет с исходным кодом может создавать двоичные пакеты и с Architecture: какая-то, и с Architecture: all



- ▶ Makefile
- ▶ Интерфейс, используемый для сборки пакетов Debian
- ▶ Документирован в Политике Debian, глава 4.8  
<https://www.debian.org/doc/debian-policy/ch-source#s-debianrules>
- ▶ Требуемые цели:
  - ▶ `build`, `build-arch`, `build-indep`: должны выполнить все настройки и компиляцию
  - ▶ `binary`, `binary-arch`, `binary-indep`: сборка двоичных пакетов
    - ▶ `dpkg-buildpackage` вызовет `binary` для сборки всех пакетов, либо `binary-arch` для сборки Architecture: какая-то пакетов
  - ▶ `clean`: очищает каталог с исходным кодом



# Утилиты, упрощающие создание пакетов – debhelper

---

- ▶ Вы можете добавить код оболочки напрямую в `debian/rules`
  - ▶ Пример см. в пакете `rsync`
- ▶ Лучшее всего (используется большинством пакетов): использовать утилиту, упрощающую создание пакетов
- ▶ Наиболее популярен: `debhelper` (используется в 98% пакетов)
- ▶ Цели:
  - ▶ Решить общие задачи стандартными средствами, для всех пакетов
  - ▶ Исправить ряд ошибок при создании пакетов, для всех пакетов

`dh_installdirs`, `dh_installchangelogs`, `dh_installdocs`, `dh_installexamples`, `dh_install`, `dh_installdebconf`, `dh_installinit`, `dh_link`, `dh_strip`, `dh_compress`, `dh_fixperms`, `dh_perl`, `dh_makeshlibs`, `dh_installdeb`, `dh_shlibdeps`, `dh_gencontrol`, `dh_md5sums`, `dh_builddeb`, ...

  - ▶ Вызывается из `debian/rules`
  - ▶ Настройка через параметры команды или файлы в `debian/`  
`пакет.docs`, `пакет.examples`, `пакет.install`, `пакет.manpages`, ...
- ▶ Сторонние утилиты для ряда пакетов: `python-support`, `dh_ocaml`, ...
- ▶ Попался: `debian/compat`: версия совместимости `debhelper` (используйте "7")

## debian/rules, использующий debhelper (1/2)

```
#!/usr/bin/make -f
```

```
# Uncomment this to turn on verbose mode.
```

```
#export DH_VERBOSE=1
```

```
build:
```

```
$(MAKE)
```

```
#docbook-to-man debian/packagename.sgml > packagename.1
```

```
clean:
```

```
dh_testdir
```

```
dh_testroot
```

```
rm -f build-stamp configure-stamp
```

```
$(MAKE) clean
```

```
dh_clean
```

```
install: build
```

```
dh_testdir
```

```
dh_testroot
```

```
dh_clean -k
```

```
dh_installdirs
```

```
# Add here commands to install the package into debian/package
```

```
$(MAKE) DESTDIR=$(CURDIR)/debian/packagename install
```

## debian/rules, использующий debhelper (2/2)

```
# Build architecture-independent files here.  
binary-indep: build install
```

```
# Build architecture-dependent files here.  
binary-arch: build install
```

```
dh_testdir  
dh_testroot  
dh_installchangelogs  
dh_installdocs  
dh_installexamples  
dh_install  
dh_installman  
dh_link  
dh_strip  
dh_compress  
dh_fixperms  
dh_installdeb  
dh_shlibdeps  
dh_gencontrol  
dh_md5sums  
dh_builddeb
```

```
binary: binary-indep binary-arch
```

```
.PHONY: build clean binary-indep binary-arch binary install configure
```



# CDBS

- ▶ С debhelper всё равно остаётся много лишней работы
- ▶ Второпорядковые утилиты с большей функциональностью
  - ▶ Напр., сборка с `./configure && make && make install` или CMake
- ▶ CDBS:
  - ▶ Представлен в 2005 году, основан на продвинутой магии GNU make
  - ▶ Документация: `/usr/share/doc/cdb/`
  - ▶ Поддержка Perl, Python, Ruby, GNOME, KDE, Java, Haskell, ...
  - ▶ Но некоторым она не нравится:
    - ▶ Иногда трудно настраивать сборку пакетов:  
"лабиринт make-файлов и переменных окружения"
    - ▶ Медленнее, чем обычный debhelper (множество бесполезных вызовов `dh_*`)

---

```
#!/usr/bin/make -f
include /usr/share/cdb/1/rules/debhelper.mk
include /usr/share/cdb/1/class/autotools.mk
```

```
# add an action after the build
build/mypackage::
    /bin/bash debian/scripts/foo.sh
```





## Dh (также известный как Debhelper 7, либо dh7)

- ▶ Представлен в 2008 году как убийца CDBS
- ▶ Команда dh, вызывающая dh\_\*
- ▶ Простой файл debian/rules, содержащий только список отклонений
- ▶ Проще настраивать, чем CDBS
- ▶ Документация: справочные страницы (debhelper(7), dh(1)) + слайды с выступления на DebConf9  
<http://kitenet.net/~joey/talks/debhelper/debhelper-slides.pdf>

---

```
#!/usr/bin/make -f
%:
    dh $@
```

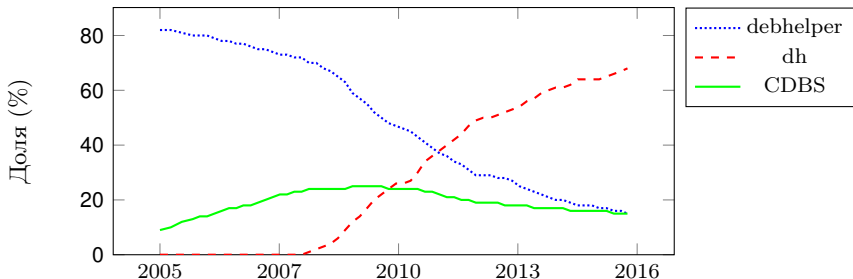
```
override_dh_auto_configure:
    dh_auto_configure -- --with-kitchen-sink
```

```
override_dh_auto_build:
    make world
```



# Классический debhelper против CDBS и dh

- ▶ Доля использования:  
Классический debhelper: 15%    CDBS: 15%    dh: 68%
- ▶ Что мне изучать?
  - ▶ Вероятно, всё по чуть-чуть
  - ▶ Вам нужно знать debhelper для использования dh и CDBS
  - ▶ Вам может потребоваться изменить пакеты CDBS
- ▶ Что использовать для нового пакета?
  - ▶ dh (единственное решение с увеличивающейся долей использования)



# План

- 1 Введение
- 2 Создание пакетов с исходным кодом
- 3 Сборка и тестирование пакетов
- 4 Практика 1: изменение пакета grep
- 5 Продвинутое темы в сборке пакетов
- 6 Сопровождение пакетов в Debian
- 7 Заключение
- 8 Дополнительные практические задания
- 9 Ответы на практические задания



## Сборка пакетов

- ▶ **apt-get build-dep мой\_пакет**

Устанавливает зависимости для сборки (build-dependencies) (для пакета, уже включённого в Debian)

Либо **mk-build-deps -ir** (для пакета, который пока не загружен)

- ▶ **debuild**: сборка, тестирование с помощью lintian, подпись GPG

- ▶ Также можно вызвать **dpkg-buildpackage** напрямую

- ▶ Обычно с помощью **dpkg-buildpackage -us -uc**

- ▶ Лучше собирать пакеты в чистом минимальном окружении

- ▶ **pbuilder** – утилита, облегчающая сборку пакетов в chroot

- Хорошая документация: <https://wiki.ubuntu.com/PbuilderHowto>  
(оптимизация: **cowbuilder ccache distcc**)

- ▶ **schroot** и **sbuild**: используются в службах сборки Debian

- (не так просты как pbuilder, но позволяют делать снимки LVM см.: <https://help.ubuntu.com/community/SbuildLVMHowto>)

- ▶ Создаёт файлы .deb и файл .changes

- ▶ .changes: описывает, что было собрано; для загрузки пакета



## Установка и тестирование пакетов

- ▶ Установить пакет локально: **debi** (будет использовать `.changes` для того, чтобы определить, что устанавливать)
- ▶ Список содержимого пакета: **debc** `../мой_пакет<ТАВ>.changes`
- ▶ Сравнить пакет с предыдущей версией:  
**debdiff** `../мой_пакет_1_*.changes ../мой_пакет_2_*.changes`  
или сравнить с исходным кодом:  
**debdiff** `../мой_пакет_1_*.dsc ../мой_пакет_2_*.dsc`
- ▶ Проверить пакет с помощью **lintian** (статический анализатор):  
**lintian** `../мой_пакет<ТАВ>.changes`  
`lintian -i`: выдаёт дополнительную информацию об ошибках  
`lintian -EviLL +pedantic`: показывает больше проблем
- ▶ Загрузить пакет в Debian (**dput**) (требуется настройка)
- ▶ Управлять частным архивом Debian с помощью **reprepro** или **textttcaptly**  
Документация: <https://wiki.debian.org/HowToSetupADebianRepository>

# План

- 1 Введение
- 2 Создание пакетов с исходным кодом
- 3 Сборка и тестирование пакетов
- 4 Практика 1: изменение пакета `grep`
- 5 Продвинутые темы в сборке пакетов
- 6 Сопровождение пакетов в Debian
- 7 Заключение
- 8 Дополнительные практические задания
- 9 Ответы на практические задания



## Практика 1: изменение пакета grep

- 1 Перейдите по адресу `http://ftp.debian.org/debian/pool/main/g/grep/` и загрузите версию 2.12-2 пакета
  - ▶ Если пакет с исходным кодом не распаковывается автоматически, распакуйте его с помощью `dpkg-source -x grep_*.dsc`
- 2 Посмотрите файлы в `debian/`.
  - ▶ Сколько двоичных пакетов создаётся этим пакетом?
  - ▶ Какая утилита для создания пакетов используется?
- 3 Соберите пакет
- 4 Мы собираемся изменить пакет. Добавьте запись в журнал изменений и увеличьте номер версии.
- 5 Теперь отключите поддержку регулярных выражений Perl (`perl-regex`) (это опция `./configure`)
- 6 Соберите пакет заново
- 7 Сравните оригинальный и новый пакеты с помощью `debdiff`
- 8 Установите собранный заново пакет



# План

- 1 Введение
- 2 Создание пакетов с исходным кодом
- 3 Сборка и тестирование пакетов
- 4 Практика 1: изменение пакета grep
- 5 Продвинутое темы в сборке пакетов
- 6 Сопровождение пакетов в Debian
- 7 Заключение
- 8 Дополнительные практические задания
- 9 Ответы на практические задания





## debian/copyright

- ▶ Авторское право и лицензионная информация для исходного кода и создания пакета
- ▶ Традиционно записывается в виде текстового файла
- ▶ Новый машиночитаемый формат:  
<https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/>

---

```
Format: https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
Upstream-Name: X Solitaire
Source: ftp://ftp.example.com/pub/games
```

```
Files: *
Copyright: Copyright 1998 John Doe <jdoe@example.com>
License: GPL-2+
This program is free software; you can redistribute it
[...]
```

```
.
On Debian systems, the full text of the GNU General Public
License version 2 can be found in the file
‘/usr/share/common-licenses/GPL-2’.
```

```
Files: debian/*
Copyright: Copyright 1998 Jane Smith <jsmith@example.net>
License:
[LICENSE TEXT]
```



# Изменение исходного кода основной ветки разработки

Зачастую требуется:

- ▶ Исправить ошибки или добавить изменения, специфичные для Debian
- ▶ Перенести исправления из более нового выпуска основной ветки

Несколько методов, чтобы сделать это:

- ▶ Изменение файлов напрямую
  - ▶ Просто
  - ▶ Нет возможность отследить и задокументировать изменения
- ▶ Использование систем заплат
  - ▶ Упрощает внесение ваших изменений в основную ветку
  - ▶ Помогает делиться исправлениями с производными дистрибутивами
  - ▶ Больше внимания изменениям  
<http://patch-tracker.debian.org/> (в настоящее время выключен)



## Системы заплат

- ▶ Принцип: изменения сохраняются в виде заплат в `debian/patches/`
- ▶ Применяются или не применяются во время сборки
- ▶ В прошлом: несколько реализаций – `simple-patchsys` (`cdbs`), `dpatch`, `quilt`
  - ▶ Каждая поддерживает две цели `debian/rules`:
    - ▶ `debian/rules patch`: применяет все заплаты
    - ▶ `debian/rules unpatch`: отменяет все заплаты
  - ▶ Дополнительная документация:  
<https://wiki.debian.org/debian/patches>
- ▶ Новый формат пакетов с исходным кодом со встроенной системой заплат: 3.0 (`quilt`)
  - ▶ Рекомендуемое решение
  - ▶ You need to learn quilt  
<https://perl-team.pages.debian.net/howto/quilt.html>
  - ▶ Независимый от системы заплат инструмент в `devscripts`:  
`edit-patch`



# Документирование заплат

- ▶ Стандартные заголовки в начале заплаты
- ▶ Документированы в DEP-3 - Руководство по тегам заплата  
<http://dep.debian.net/deps/dep3/>

---

```
Description: Fix widget frobnication speeds
Frobnicating widgets too quickly tended to cause explosions.
Forwarded: http://lists.example.com/2010/03/1234.html
Author: John Doe <johndoe-guest@users.alioth.debian.org>
Applied-Upstream: 1.2, http://bZR.foo.com/frobnicator/revision/123
Last-Update: 2010-03-29
```

```
--- a/src/widgets.c
+++ b/src/widgets.c
@@ -101,9 +101,6 @@ struct {
```



## Сделать что-то во время установки или удаления

- ▶ Иногда недостаточно просто распаковать пакет
- ▶ Создать/удалить системных пользователей, запустить/остановить службы, обработать альтернативы
- ▶ Осуществляется в сценариях сопровождающего preinst, postinst, prerm, postrm
  - ▶ Части общих действий могут быть созданы при помощи debhelper
- ▶ Документация:
  - ▶ Руководство по политике Debian, глава 6  
<https://www.debian.org/doc/debian-policy/ch-maintainerscripts>
  - ▶ Справочник разработчика Debian, глава 6.4  
<https://www.debian.org/doc/developers-reference/best-pkging-practices.html>
  - ▶ <https://people.debian.org/~srivasta/MaintainerScripts.html>
- ▶ Приглашение пользователя
  - ▶ Должно осуществляться с помощью debconf
  - ▶ Документация: debconf-devel(7) (пакет debconf-doc)



## Отслеживание версий основной ветки

- ▶ В файле `debian/watch` укажите где искать (см. `uscan(1)`)

```
version=3
```

```
http://tmrc.mit.edu/mirror/twisted/Twisted/(\d\.\d)/ \
Twisted - ([\d\.]*)\.tar\.bz2
```

- ▶ There are automated trackers of new upstream versions, that notify the maintainer on various dashboards including <https://tracker.debian.org/> and <https://udd.debian.org/dmd/>
- ▶ `uscan`: запустить ручную проверку
- ▶ `uupdate`: попытаться обновить ваш пакет до самой последней версии основной ветки разработки



## Создание пакетов в системе контроля версий

- ▶ Инструменты, помогающие управлять ветками и тегами вашей работы по созданию пакетов:  
svn-buildpackage, git-buildpackage
- ▶ Пример: git-buildpackage
  - ▶ Ветка upstream для отслеживания основной ветки с тегами upstream/version
  - ▶ Ветка master отслеживает пакет Debian
  - ▶ Теги debian/version для каждой загрузки
  - ▶ Ветка pristine-tar для сборки tarball из основной ветки

Дос:

<http://honk.sigxcpu.org/projects/git-buildpackage/manual-html/gbp.html>

- ▶ Поля Vcs-\* в debian/control для определения репозитория
  - ▶ <https://wiki.debian.org/Alioth/Git>
  - ▶ <https://wiki.debian.org/Alioth/Svn>

Vcs-Browser: <http://anonscm.debian.org/gitweb/?p=collab-maint/devscripts.git>

Vcs-Git: <git://anonscm.debian.org/collab-maint/devscripts.git>

Vcs-Browser: <http://svn.debian.org/viewsvn/pkg-perl/trunk/libwww-perl/>

Vcs-Svn: <svn://svn.debian.org/pkg-perl/trunk/libwww-perl>

- ▶ Интерфейс, независящий от системы контроля версий: debcheckout,



# Обратный перенос пакетов

- ▶ Цель: использовать более новую версию пакета на старой системе напр., использовать mutt из Debian unstable на Debian stable
- ▶ Общая идея:
  - ▶ Взять пакет с исходным кодом из Debian unstable
  - ▶ Изменить его так, чтобы он собирался и нормально работал в Debian stable
    - ▶ Иногда тривиально (изменения не требуются)
    - ▶ Иногда сложно
    - ▶ Иногда невозможно (много недоступных зависимостей)
- ▶ Некоторые обратные переносы предоставляются и поддерживаются Проектом Debian  
<http://backports.debian.org/>



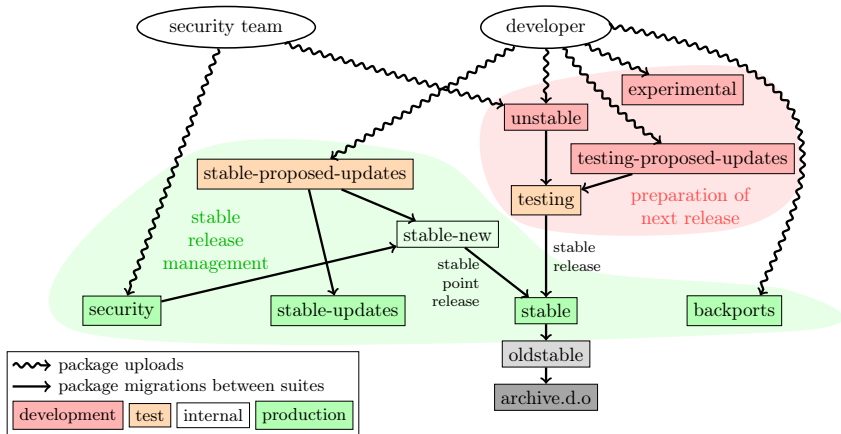


# План

- 1 Введение
- 2 Создание пакетов с исходным кодом
- 3 Сборка и тестирование пакетов
- 4 Практика 1: изменение пакета grep
- 5 Продвинутые темы в сборке пакетов
- 6 Сопровождение пакетов в Debian
- 7 Заключение
- 8 Дополнительные практические задания
- 9 Ответы на практические задания



# Debian archive and suites



Based on graph by Antoine Beaupré. <https://salsa.debian.org/debian/package-cycle>

## Suites for development

- ▶ New versions of packages are uploaded to unstable (sid)
- ▶ Packages migrate from unstable to testing based on several criterias (e.g. has been in unstable for 10 days, and no regressions)
- ▶ New packages can also be uploaded to:
  - ▶ experimental (for more experimental packages, such as when the new version is not ready to replace the one currently in unstable)
  - ▶ testing-proposed-updates, to update the version in testing without going through unstable (this is rarely used)



## Freezing and releasing

- ▶ At some point during the release cycle, the release team decides to freeze testing: automatic migrations from unstable to testing are stopped, and replaced by manual review
- ▶ When the release team considers testing to be ready for release:
  - ▶ The testing suite becomes the new stable suite
  - ▶ Similarly, the old stable becomes oldstable
  - ▶ Unsupported releases are moved to [archive.debian.org](https://archive.debian.org)
- ▶ See <https://release.debian.org/>



# Stable release management

- ▶ Several suites are used to provide stable release packages:
  - ▶ stable: the main suite
  - ▶ security updates suite provided on security.debian.org, used by the security team. Updates are announced on the debian-security-announce mailing list
  - ▶ stable-updates: updates that are not security related, but that should urgently be installed (without waiting for the next point release): antivirus databases, timezone-related packages, etc. Announced on the debian-stable-announce mailing list
  - ▶ backports: new upstream versions, based on the version in testing
- ▶ The stable suite is updated every few months by stable point releases (that include only bug fixes)
  - ▶ Packages targetting the next stable point release are uploaded to stable-proposed-updates and reviewed by the release team
- ▶ The oldstable release has the same set of suites



# Несколько способов внести вклад в Debian

## ▶ Худший способ участия:

- ❶ Создать пакет для вашего собственного приложения
- ❷ Добавить его в Debian
- ❸ Исчезнуть

## ▶ Лучшие способы участия:

- ▶ Подключиться к работе команд по созданию пакетов
  - ▶ Многие команды концентрируются на наборах пакетов, им нужна помощь
  - ▶ Список доступен по адресу <https://wiki.debian.org/Teams>
  - ▶ Отличный способ научиться у более опытных участников
- ▶ Усыновление существующих несопровождаемых пакетов (осиротевших пакетов)
- ▶ Привнести в Debian новое ПО
  - ▶ Пожалуйста, только если оно достаточно интересно/полезно
  - ▶ Существуют ли какие-либо альтернативы для которых уже созданы пакеты Debian?



# Усыновление осиротевших пакетов

- ▶ В Debian много несопровождаемых пакетов
- ▶ Полный список + процесс: <https://www.debian.org/devel/wnpp/>
- ▶ Установлены на вашей машине: `wnpp-alert`  
Или лучше: `how-can-i-help`
- ▶ Различные состояния:
  - ▶ Orphaned (осиротевший): пакет не сопровождается  
Усыновите его
  - ▶ RFA: Request For Adopter (Требуется усыновитель)  
Сопровождающий ищет усыновителя, но пока продолжает работать  
Усыновите его. Обратитесь к текущему сопровождающему
  - ▶ ITA: Intent To Adopt (Собираюсь усыновить)  
Кто-то намерен усыновить пакет  
Помогите ему!
  - ▶ RFH: Request For Help (Запрос о помощи)  
Сопровождающий ищет помощь
- ▶ Ряд несопровождаемых пакетов ещё не обнаружен → не осиротели
- ▶ Спрашивайте на [debian-qa@lists.debian.org](mailto:debian-qa@lists.debian.org)



## Усыновление пакета: пример

```
From: You <you@yourdomain>  
To: 640454@bugs.debian.org, control@bugs.debian.org  
Cc: Francois Marier <francois@debian.org>  
Subject: ITA: verbiste -- French conjugator
```

```
retitle 640454 ITA: verbiste -- French conjugator  
owner 640454 !  
thanks
```

Hi,

I am using verbiste and I am willing to take care of the package.

Cheers,

You

- ▶ Считается вежливым, если вы свяжитесь с предыдущим сопровождающим (особенно, если пакет имеет статус RFA и пока не осиротел)
- ▶ Хорошо бы связаться с проектом основной ветки разработки





## Добавление в Debian вашего пакета

- ▶ Для добавления вашего пакета в Debian вам не нужно иметь официальный статус
  - ➊ Отправьте ИТР отчет (Intent To Package (Намерен создать пакет)) при помощи `reportbug wnpp`
  - ➋ Подготовьте пакет с исходным кодом
  - ➌ Найдите разработчика Debian, который будет спонсировать ваш пакет
- ▶ Официальный статус (когда вы уже являетесь опытным сопровождающим пакетов):
  - ▶ Сопровождающий Debian (DM):  
Право на загрузку собственных пакетов  
См. <https://wiki.debian.org/DebianMaintainer>
  - ▶ Разработчик Debian (DD):  
Член Проекта Debian; может голосовать и загружать любой пакет



## Что проверить прежде чем просить о спонсорстве

- ▶ Debian обращает большое внимание на качество
- ▶ Обычно, спонсоров сложно найти, и они заняты
  - ▶ До того, как просить о спонсорстве, убедитесь, что ваш пакет готов
- ▶ Что проверить:
  - ▶ Избегайте отсутствующих зависимостей для сборки: убедитесь, что ваш пакет нормально собирается в чистом sid chroot
    - ▶ Рекомендуется использовать pbuilder
  - ▶ Проверьте ваш пакет с помощью lintian -EviIL +pedantic
    - ▶ Ошибки должны быть исправлены, все остальные проблемы тоже следует устранить
  - ▶ Конечно, хорошенько протестируйте ваш пакет
- ▶ Если у вас возникли сомнения, попросите помощи



## Где искать помощь?

Помощь, которая вам потребуется:

- ▶ Советы и ответы на вопросы, отзывы о коде
- ▶ Спонсирование ваших загрузок, когда ваш пакет будет готов

Вы можете получить помощь от:

- ▶ Других членов команды по созданию пакетов
  - ▶ Список команд: <https://wiki.debian.org/Teams>
- ▶ Группы наставников Debian (если ваш пакет не подходит ни для одной группы)
  - ▶ <https://wiki.debian.org/DebianMentorsFaq>
  - ▶ Список рассылки: [debian-mentors@lists.debian.org](mailto:debian-mentors@lists.debian.org)  
(так же хороший способ случайно чему-то научиться)
  - ▶ IRC: #debian-mentors на [irc.debian.org](http://irc.debian.org)
  - ▶ <http://mentors.debian.net/>
  - ▶ Документация: <http://mentors.debian.net/intro-maintainers>
- ▶ Локализованные списки рассылки (получите помощь на вашем языке)
  - ▶ [debian-devel-{french,italian,portuguese,spanish}@lists.d.o](mailto:debian-devel-{french,italian,portuguese,spanish}@lists.d.o)
  - ▶ Полный список: <https://lists.debian.org/devel.html>
  - ▶ Или пользовательские списки: <https://lists.debian.org/users.html>



## Дополнительная документация

- ▶ Уголок разработчика Debian  
<https://www.debian.org/devel/>  
Ссылки на множество ресурсов о разработке Debian
- ▶ Guide for Debian Maintainers  
<https://www.debian.org/doc/manuals/debmake-doc/>
- ▶ Справочник разработчика Debian  
<https://www.debian.org/doc/developers-reference/>  
По большей части о процедурах Debian, но также о некоторых лучших практиках создания пакетов (часть 6)
- ▶ Политика Debian  
<https://www.debian.org/doc/debian-policy/>
  - ▶ Все требования, которые должен выполнять всякий пакет
  - ▶ Определяет политики для Perl, Java, Python, ...
- ▶ Руководство по созданию пакетов Ubuntu  
<http://developer.ubuntu.com/resources/tools/packaging/>



## Панели Debian для сопровождающих

- ▶ Фокус на пакетах с исходным кодом: <https://tracker.debian.org/dpkg>
- ▶ Фокус на сопровождающем/команде: Обзор пакетов разработчика (DDPO)  
<https://qa.debian.org/developer.php?login=pkg-ruby-extras-maintainers@lists.alioth.debian.org>
- ▶ Фокус на списке задач: Панель сопровождающего Debian (DMD)  
<https://udd.debian.org/dmd/>



# Использование системы отслеживания ошибок (BTS)

---

- ▶ Уникальный способ управления ошибками
  - ▶ Веб-интерфейс для просмотра ошибок
  - ▶ Интерфейс электронной почты для изменения ошибок
- ▶ Добавление информации к ошибкам:
  - ▶ Напишите на `123456@bugs.debian.org` (не включает отправителя ошибки, вам нужно добавить `123456-submitter@bugs.debian.org`)
- ▶ Изменение статуса ошибки:
  - ▶ Отправьте команды на `control@bugs.debian.org`
  - ▶ Интерфейс командной строки: команда `bts` в `devscripts`
  - ▶ Документация: <https://www.debian.org/Bugs/server-control>
- ▶ Отправка отчётов об ошибках: используйте `reportbug`
  - ▶ Обычно используется с локальным почтовым сервером: `install ssmtp` или `nullmailer`
  - ▶ Либо используйте `reportbug --template`, затем отправьте (вручную) по адресу `submit@bugs.debian.org`



## Использование BTS: примеры

- ▶ Отправка сообщения ошибке и приславшему эту ошибку:  
<https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680822#10>
- ▶ Отметка тегом и изменение серьёзности:  
<https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680227#10>
- ▶ Переназначение, изменение серьёзности, изменение заголовка . . . :  
<https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680822#93>
  - ▶ notfound, found, notfixed, fixed для version-tracking  
См. [https://wiki.debian.org/HowtoUseBTS#Version\\_tracking](https://wiki.debian.org/HowtoUseBTS#Version_tracking)
- ▶ Использование пользовательских тегов:  
<https://bugs.debian.org/cgi-bin/bugreport.cgi?msg=42;bug=642267>  
См. <https://wiki.debian.org/bugs.debian.org/usertags>
- ▶ Документация по BTS:
  - ▶ <https://www.debian.org/Bugs/>
  - ▶ <https://wiki.debian.org/HowtoUseBTS>



## Больше заинтересованы в Ubuntu?

- ▶ В случае Ubuntu в основном обрабатываются расхождения с Debian
- ▶ Не фокусирует внимание на конкретных пакетах  
Сотрудничает с командами Debian
- ▶ Обычно рекомендуется загружать новые пакеты сначала в Debian  
<https://wiki.ubuntu.com/UbuntuDevelopment/NewPackages>
- ▶ Вероятно, лучше:
  - ▶ Принять участие в команде Debian и действовать в качестве моста с Ubuntu
  - ▶ Помогать уменьшить расхождения, сортировать ошибки в Launchpad
  - ▶ Вам могут помочь многие инструменты Debian:
    - ▶ Колонка Ubuntu в обзоре пакетов для разработчика
    - ▶ Бокс Ubuntu в системе отслеживания пакетов
    - ▶ Получайте почту об ошибках от Launchpad через PTS





# План

- 1 Введение
- 2 Создание пакетов с исходным кодом
- 3 Сборка и тестирование пакетов
- 4 Практика 1: изменение пакета `grep`
- 5 Продвинутое темы в сборке пакетов
- 6 Сопровождение пакетов в Debian
- 7 Заключение
- 8 Дополнительные практические задания
- 9 Ответы на практические задания



## Заключение

- ▶ Теперь у вас имеется полный обзор процесса создания пакетов Debian
- ▶ Но вам нужно ознакомиться с дополнительной документацией
- ▶ Лучшие практики эволюционировали на протяжении нескольких лет
  - ▶ Если вы не уверены, используйте утилиту для создания пакетов `dh`, а также формат 3.0 (`quilt`)

Обратная связь: [packaging-tutorial@packages.debian.org](mailto:packaging-tutorial@packages.debian.org)



# ПРАВОВЫЕ ВОПРОСЫ

Copyright ©2011–2019 Лукас Нуссбаум – lucas@debian.org

This document is free software: you can redistribute it and/or modify it under either (at your option):

- ▶ The terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.  
<http://www.gnu.org/licenses/gpl.html>
- ▶ The terms of the Creative Commons Attribution-ShareAlike 3.0 Unported License.  
<http://creativecommons.org/licenses/by-sa/3.0/>



# Принять участие в развитии этого руководства

## ▶ Принять участие:

- ▶ `apt-get source packaging-tutorial`
- ▶ `debcheckout packaging-tutorial`
- ▶ `git clone`  
`git://git.debian.org/collab-maint/packaging-tutorial.git`
- ▶ `http://git.debian.org/?p=collab-maint/packaging-tutorial.git`
- ▶ Открытые ошибки: `bugs.debian.org/src:packaging-tutorial`

## ▶ Обратная связь:

- ▶ `mailto:packaging-tutorial@packages.debian.org`
  - ▶ Что следует добавить в это руководство?
  - ▶ Что следует улучшить?
- ▶ `reportbug packaging-tutorial`



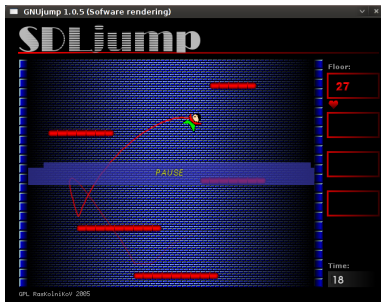
# План

- 1 Введение
- 2 Создание пакетов с исходным кодом
- 3 Сборка и тестирование пакетов
- 4 Практика 1: изменение пакета `grep`
- 5 Продвинутые темы в сборке пакетов
- 6 Сопровождение пакетов в Debian
- 7 Заключение
- 8 Дополнительные практические задания
- 9 Ответы на практические задания



## Практика 2: создание пакета GNUjump

- 1 Загрузите GNUjump 1.0.8 по адресу <http://ftp.gnu.org/gnu/gnujump/gnujump-1.0.8.tar.gz>
- 2 Создайте для него пакет Debian
  - ▶ Установите зависимости для сборки, чтобы вы смогли собрать пакет
  - ▶ Исправьте ошибки
  - ▶ Получите простой работающий пакет
  - ▶ Закончите заполнение `debian/control` и других файлов
- 3 Наслаждайтесь



## Практика 2: создание пакета GNUjump (подсказки)

- ▶ Для получения базового работающего пакета используйте команду:  
`dh_make`
- ▶ Создать пакет с исходным кодом формата 1.0 проще, чем формат 3.0 (`quilt`) (измените формат в `debian/source/format`)
- ▶ Для поиска отсутствующих сборочных зависимостей, определите отсутствующий файл, а затем используйте `art-file` для того, чтобы найти отсутствующий пакет
- ▶ Если вы столкнулись со следующей ошибкой:

```
/usr/bin/ld: SDL_rotozoom.o: undefined reference to symbol 'ceil@@GLIBC_2.2.5'  
//lib/x86_64-linux-gnu/libm.so.6: error adding symbols: DSO missing from command line  
collect2: error: ld returned 1 exit status  
Makefile:376: recipe for target 'gnujump' failed
```

вам следует добавить `-lm` в строку с командой вызова компоновщика:  
Отредактируйте `src/Makefile.am` и замените

```
gnujump_LDFLAGS = $(all_libraries)
```

на

```
gnujump_LDFLAGS = -Wl,--as-needed  
gnujump_LDADD = $(all_libraries) -lm
```

Затем запустите `autoreconf -i`



## Практика 3: создание пакета библиотеки Java

- 1 Просмотрите документацию о создании пакетов Java:
  - ▶ <https://wiki.debian.org/Java>
  - ▶ <https://wiki.debian.org/Java/Packaging>
  - ▶ <https://www.debian.org/doc/packaging-manuals/java-policy/>
  - ▶ [/usr/share/doc/javahelper/tutorial.txt.gz](https://usr/share/doc/javahelper/tutorial.txt.gz)
- 2 Загрузите IRClib по адресу <http://moepii.sourceforge.net/>
- 3 Создайте пакет





## Практика 4: создание пакета с gem-пакетом Ruby

- 1 Просмотрите документацию о создании пакетов Ruby:
  - ▶ <https://wiki.debian.org/Ruby>
  - ▶ <https://wiki.debian.org/Teams/Ruby>
  - ▶ <https://wiki.debian.org/Teams/Ruby/Packaging>
  - ▶ `gem2deb(1)`, `dh_ruby(1)` (in the `gem2deb` package)
- 2 Создайте простой Debian пакет с исходным кодом из `peach gem`:  
`gem2deb peach`
- 3 Улучшите его так, чтобы он стал правильным пакетом Debian



## Практика 5: создание пакета с модулем Perl

- 1 Просмотрите документацию о создании пакетов Perl:
  - ▶ <https://perl-team.pages.debian.net>
  - ▶ <https://wiki.debian.org/Teams/DebianPerlGroup>
  - ▶ `dh-make-perl(1)`, `dpt(1)` (in the `pkg-perl-tools` package)
- 2 Создайте простой Debian пакет с исходным кодом из Асме CPAN:  
`dh-make-perl --cpan Асме`
- 3 Улучшите его так, чтобы он стал правильным пакетом Debian



# План

- 1 Введение
- 2 Создание пакетов с исходным кодом
- 3 Сборка и тестирование пакетов
- 4 Практика 1: изменение пакета `grep`
- 5 Продвинутые темы в сборке пакетов
- 6 Сопровождение пакетов в Debian
- 7 Заключение
- 8 Дополнительные практические задания
- 9 Ответы на практические задания



# Ответы на практические задания



## Практика 1: изменение пакета grep

- 1 Перейдите по адресу `http://ftp.debian.org/debian/pool/main/g/grep/` и загрузите версию 2.12-2 пакета
- 2 Посмотрите файлы в `debian/`.
  - ▶ Сколько двоичных пакетов создаётся этим пакетом?
  - ▶ Какая утилита для создания пакетов используется?
- 3 Соберите пакет
- 4 Мы собираемся изменить пакет. Добавьте запись в журнал изменений и увеличьте номер версии.
- 5 Теперь отключите поддержку регулярных выражений Perl (`perl-regex`) (это опция `./configure`)
- 6 Соберите пакет заново
- 7 Сравните оригинальный и новый пакеты с помощью `debdiff`
- 8 Установите собранный заново пакет



## Загрузка исходного кода

- ➊ Перейдите по адресу `http://ftp.debian.org/debian/pool/main/g/grep/` и загрузите версию 2.12-2 пакета
  - ▶ Используйте `dget` для загрузки файла `.dsc`:  
`dget http://cdn.debian.net/debian/pool/main/g/grep/grep_2.12-2.dsc`
  - ▶ If you have deb-src for a Debian release that has grep version 2.12-2 (find out on <https://tracker.debian.org/grep>), you can use: `apt-get source grep=2.12-2`  
or `apt-get source grep/release` (e.g. `grep/stable`)  
or, if you feel lucky: `apt-get source grep`
  - ▶ Пакет с исходным кодом `grep` содержит три файла:
    - ▶ `grep_2.12-2.dsc`
    - ▶ `grep_2.12-2.debian.tar.bz2`
    - ▶ `grep_2.12.orig.tar.bz2`

Это обычный формат "3.0 (quilt)".

- ▶ Если это необходимо, распакуйте исходный код при помощи `dpkg-source -x grep_2.12-2.dsc`



## Осматриваем и собираем пакет

### 2 Look at the files in debian/

- ▶ Сколько двоичных пакетов создаётся этим пакетом?
  - ▶ Какая утилита для создания пакетов используется?
- ▶ Согласно `debian/control`, этот пакет создаёт только один двоичный пакет, а именно `grep`.
- ▶ Согласно `debian/rules`, этот пакет представляет собой типичный пакет для классического `debhelper`, без использования `CDBS` или `dh`. Можно видеть различные вызовы команд `dh_*` в `debian/rules`.

### 3 Соберите пакет

- ▶ Используйте `apt-get build-dep grep` для загрузки сборочных зависимостей
- ▶ Далее, `debuild` или `dpkg-buildpackage -us -uc` (Занимает 1 минуту)



## Редактирование журнала изменений

- ④ Мы собираемся изменить пакет. Добавьте запись в журнал изменений и увеличьте номер версии.
  - ▶ `debian/changelog` является текстовым файлом. Можно отредактировать его и добавить новую запись вручную.
  - ▶ Либо можно использовать `dch -i`, который добавит запись и откроет редактор
  - ▶ Имя и адрес электронной почты могут быть определены при помощи переменных окружения `DEBFULLNAME` и `DEBEMAIL`
  - ▶ После этого соберите пакет заново: будет собрана новая версия пакета
  - ▶ Присвоение версий пакетам описывается в разделе 5.6.12 Политики Debian  
<https://www.debian.org/doc/debian-policy/ch-controlfields>





## Отключение регулярных выражений Perl

- 5 Теперь отключите поддержку регулярных выражений Perl (perl-regex) (это опция ./configure)
  - 6 Соберите пакет заново
- ▶ Посмотрите ./configure --help: опция для отключения регулярных выражений Perl – --disable-perl-regex
  - ▶ Откройте для редактирования debian/rules и найдите строку ./configure
  - ▶ Добавьте --disable-perl-regex
  - ▶ Соберите заново при помощи debuild или dpkg-buildpackage -us -uc



# Сравнение и тестирование пакетов

- 7 Сравните оригинальный и новый пакеты с помощью `debdiff`
- 8 Установите собранный заново пакет
  - ▶ Сравните двоичные пакеты: `debdiff ../changes`
  - ▶ Сравните пакеты с исходным кодом: `debdiff ../dsc`
  - ▶ Установите заново собранный пакет: `debi`  
Или `dpkg -i ../grep_<ТАВ>`
  - ▶ `grep -P foo` больше не работает!

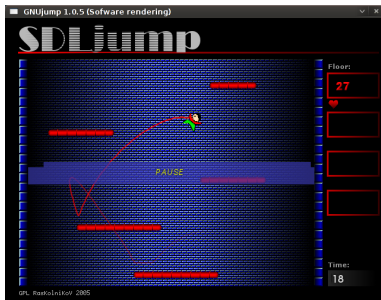
Переустановите предыдущую версию пакета:

- ▶ `apt-get install --reinstall grep=2.6.3-3` (= предыдущая версия)



## Практика 2: создание пакета GNUjump

- 1 Загрузите GNUjump 1.0.8 по адресу <http://ftp.gnu.org/gnu/gnujump/gnujump-1.0.8.tar.gz>
- 2 Создайте для него пакет Debian
  - ▶ Установите зависимости для сборки, чтобы вы смогли собрать пакет
  - ▶ Получите простой работающий пакет
  - ▶ Закончите заполнение `debian/control` и других файлов
- 3 Наслаждайтесь



## Шаг за шагом...

- ▶ `wget http://ftp.gnu.org/gnu/gnujump/gnujump-1.0.8.tar.gz`
- ▶ `mv gnujump-1.0.8.tar.gz gnujump_1.0.8.orig.tar.gz`
- ▶ `tar xf gnujump_1.0.8.orig.tar.gz`
- ▶ `cd gnujump-1.0.8/`
- ▶ `dh_make -f ../gnujump-1.0.8.tar.gz`
  - ▶ Тип пакета: один двоичный (сейчас)

```
gnujump-1.0.8$ ls debian/
changelog          gnujump.default.ex  preinst.ex
compat            gnujump.doc-base.EX  prerm.ex
control           init.d.ex           README.Debian
copyright         manpage.1.ex       README.source
docs             manpage.sgml.ex    rules
emacsen-install.ex  manpage.xml.ex     source
emacsen-remove.ex  menu.ex            watch.ex
emacsen-startup.ex  postinst.ex
gnujump.cron.d.ex  postrm.ex
```



## Шаг за шагом... (2)

- ▶ Посмотрите `debian/changelog`, `debian/rules`, `debian/control` (заполняются автоматически при помощи `dh_make`)
- ▶ В `debian/control`:  
Build-Depends: `debhelper (>= 7.0.50)`, `autotools-dev`  
Приведён список сборочных зависимостей = пакетов, необходимых для сборки пакета
- ▶ Попробуйте собрать пакет как есть с помощью `debuild` (используя магию `dh`)
  - ▶ Добавляйте сборочные зависимости, пока он не будет собираться
  - ▶ Подсказка: используйте `apt-cache search` и `apt-file`, чтобы найти пакеты
  - ▶ Пример:

```
checking for sdl-config... no
checking for SDL - version >= 1.2.0... no
[...]
```

`configure: error: *** SDL version 1.2.0 not found!`  
→ Добавьте `libsdl1.2-dev` в поле Build-Depends и установите этот пакет.
- ▶ Лучше: используйте `rbuilder` для сборки в чистом окружении



## Шаг за шагом... (3)

- ▶ Требуются следующие сборочные зависимости: `libsdl1.2-dev`, `libsdl-image1.2-dev`, `libsdl-mixer1.2-dev`
- ▶ Далее, скорее всего вы столкнётесь с другой ошибкой:

```
/usr/bin/ld: SDL_rotozoom.o: undefined reference to symbol 'ceil@@GLIBC_2.2.5'  
//lib/x86_64-linux-gnu/libm.so.6: error adding symbols: DSO missing from command  
collect2: error: ld returned 1 exit status  
Makefile:376: recipe for target 'gnujump' failed
```

- ▶ Эта проблема вызвана `bitrot`: `gnujump` не был настроен в соответствии с изменениями компоновщика.
- ▶ Если вы используете формат исходного кода версии 1.0, вы можете напрямую изменять исходный код основной ветки разработки.

- ▶ Отредактируйте `src/Makefile.am` и замените

```
gnujump_LDFLAGS = $(all_libraries)
```

на

```
gnujump_LDFLAGS = -Wl,--as-needed  
gnujump_LDADD = $(all_libraries) -lm
```

- ▶ Затем запустите `autoreconf -i`



## Шаг за шагом... (4)

- ▶ Если вы используете формат исходного кода версии 3.0 (quilt), то используйте quilt для создания заплата. (См. <https://wiki.debian.org/UsingQuilt>)
  - ▶ `export QUILT_PATCHES=debian/patches`
  - ▶ `mkdir debian/patches`  
`quilt new linker-fixes.patch`  
`quilt add src/Makefile.am`
  - ▶ Отредактируйте `src/Makefile.am` и замените `gnujump_LDFLAGS = $(all_libraries)` на  
`gnujump_LDFLAGS = -Wl,--as-needed`  
`gnujump_LDADD = $(all_libraries) -lm`
  - ▶ `quilt refresh`
  - ▶ Поскольку `src/Makefile.am` был изменён, во время сборки следует вызвать `autoreconf`. Чтобы сделать это автоматически с помощью `dh`, измените вызов `dh` в `debian/rules` с `dh $ --with autotools-dev` на `dh $ --with autotools-dev --with autoreconf`



## Шаг за шагом... (5)

- ▶ Теперь сборка пакета должна завершаться корректно.
- ▶ Используйте `debcs` для получения списка содержимого созданного пакета и `debi` для его установки и тестирования.
- ▶ Протестируйте пакет при помощи `lintian`
  - ▶ Хотя это не является строгим требованием, рекомендуется, чтобы пакеты, загружаемые в Debian были `lintian-clean`
  - ▶ Дополнительные проблемы могут быть просмотрены при помощи `lintian -EviLL +pedantic`
  - ▶ Несколько подсказок:
    - ▶ Удалите ненужные вам файлы в `debian/`
    - ▶ Заполните `debian/control`
    - ▶ Установите выполняемые файлы в `/usr/games`, изменив `dh_auto_configure`
    - ▶ Используйте флаги `hardening` компилятора для увеличения безопасности.  
См. <https://wiki.debian.org/Hardening>





## Шаг за шагом... (6)

- ▶ Сравните ваш пакет с пакетом в Debian:
  - ▶ Это выделит файлы с данными во второй пакет, который одинаков для всех архитектур (→ сохраняет место в архиве Debian)
  - ▶ Это установит файл `.desktop` (для меню GNOME/KDE), а также интегрирует в меню Debian
  - ▶ Это исправит несколько небольших проблем при использовании заплат



## Практика 3: создание пакета библиотеки Java

- 1 Просмотрите документацию о создании пакетов Java:
  - ▶ <https://wiki.debian.org/Java>
  - ▶ <https://wiki.debian.org/Java/Packaging>
  - ▶ <https://www.debian.org/doc/packaging-manuals/java-policy/>
  - ▶ [/usr/share/doc/javahelper/tutorial.txt.gz](https://usr/share/doc/javahelper/tutorial.txt.gz)
- 2 Загрузите IRClib по адресу <http://moepii.sourceforge.net/>
- 3 Создайте пакет



## Шаг за шагом...

- ▶ `apt-get install javahelper`
- ▶ Создайте простой пакет с исходным кодом: `jh_makerkg`
  - ▶ Библиотека
  - ▶ Нет
  - ▶ Свободный компилятор/время выполнения по умолчанию
- ▶ Посмотрите и исправьте `debian/*`
- ▶ `dpkg-buildpackage -us -uc` или `debuild`
- ▶ `lintian`, `debc` и т.д.
- ▶ Сравните ваш результат с пакетом с исходным кодом `libirclib-java`



## Практика 4: создание пакета с gem-пакетом Ruby

- 1 Просмотрите документацию о создании пакетов Ruby:
  - ▶ <https://wiki.debian.org/Ruby>
  - ▶ <https://wiki.debian.org/Teams/Ruby>
  - ▶ <https://wiki.debian.org/Teams/Ruby/Packaging>
  - ▶ `gem2deb(1)`, `dh_ruby(1)` (in the `gem2deb` package)
- 2 Создайте простой Debian пакет с исходным кодом из `peach gem`:  
`gem2deb peach`
- 3 Улучшите его так, чтобы он стал правильным пакетом Debian



## Шаг за шагом...

gem2deb peach:

- ▶ Загрузите gem-пакет с [rubygems.org](http://rubygems.org)
- ▶ Создаёт подходящий архив `.orig.tar.gz`, и распаковывает его
- ▶ Инициализирует пакет Debian с исходным кодом на основе метаданных gem-пакета
  - ▶ Имеет имя `ruby-имя_gem`
- ▶ Пытается собрать двоичный пакет Debian (это может не сработать)

`dh_ruby` (включён в `gem2deb`) выполняет специфичные для Ruby задачи:

- ▶ Собирает расширения C для каждой версии Ruby
- ▶ Копирует файлы в их каталоги назначения
- ▶ Обновляет `shebang`-и в исполняемых сценариях
- ▶ Запускает тесты, определённые в `debian/ruby-tests.rb`, `debian/ruby-tests.rake` или `debian/ruby-test-files.yaml`, а также другие проверки



## Шаг за шагом... (2)

Улучшить созданный пакет:

- ▶ Запустите `debclean` для очистки дерева исходного кода. Посмотрите `debian/`.
- ▶ `changelog` и `compat` должны быть верны
- ▶ Отредактируйте `debian/control`: улучшите `Description`
- ▶ Напишите правильный файл `copyright` на основе файлов из основной ветки разработки
- ▶ Соберите пакет
- ▶ Сравните ваш пакет с пакетом `ruby-peach` в архиве Debian



## Практика 5: создание пакета с модулем Perl

- 1 Просмотрите документацию о создании пакетов Perl:
  - ▶ <https://perl-team.pages.debian.net>
  - ▶ <https://wiki.debian.org/Teams/DebianPerlGroup>
  - ▶ `dh-make-perl(1)`, `dpt(1)` (in the `pkg-perl-tools` package)
- 2 Создайте простой Debian пакет с исходным кодом из Асме CPAN:  
`dh-make-perl --cpан Асме`
- 3 Улучшите его так, чтобы он стал правильным пакетом Debian



## Шаг за шагом...

dh-make-perl --сran Асме:

- ▶ Загрузите tarball с CPAN
- ▶ Создаёт подходящий архив .orig.tar.gz и распаковывает его
- ▶ Инициализирует пакет Debian с исходным кодом на основе метаданных дистрибутива
  - ▶ Имеет имя libdistname-perl





## Шаг за шагом... (2)

Улучшить созданный пакет:

- ▶ `debian/changelog`, `debian/compat`, `debian/libasme-perl.docs` и `debian/watch` должны быть правильными
- ▶ Отредактируйте `debian/control`: улучшите `Description` и удалите шаблон снизу
- ▶ Отредактируйте `debian/copyright`: удалите шаблон сверху, добавьте годы действия авторского права в строку `Files: *`

