Manual de Empacotamento Debian

Lucas Nussbaum
packaging-tutorial@packages.debian.org

Tradução para o português de
Tássia Camões Araújo, Leandro Luiz Pereira
e equipe de tradução para o português do Brasil

versão 0.24 – 2019-03-13

Sobre este manual

- Objetivo: **dizer o que é essencial saber sobre empacotamento Debian**
  - Modificar pacotes existentes
  - Criar os seus próprios pacotes
  - Interagir com a comunidade Debian
  - Tornar-se um usuário avançado do Debian
- Cobre os pontos mais importantes, mas não é completo
  - Você vai precisar ler mais documentação
- A maioria do conteúdo também se aplica a distribuições derivadas do Debian
  - Isso inclui Ubuntu

Sumário

Sumário

Debian

- **Distribuição GNU/Linux**
- 1ª grande distribuição desenvolvida “abertamente no espírito GNU”
- **Não-comercial**, construída via colaboração por mais de 1000 voluntários
- 3 funcionalidades principais:
  - **Qualidade** – cultura de excelência técnica
    Nós lançamos quando está pronto
  - **Liberdade** – desenvolvedores e usuários unidos pelo *Contrato Social*
    Promovendo a cultura do Software Livre desde 1993
  - **Independência** – nenhuma (única) companhia toma conta do Debian
    E processo de tomada de decisão aberto (*do-ocracy* + *democracy*)
- **Amador** no melhor sentido: feito com amor

Pacotes Debian

- arquivos **.deb** (pacotes binários)
- Uma maneira muito poderosa e conveniente de distribuir software aos usuários
- Um dos dois formatos de pacotes mais comuns (juntamente com o RPM)
- Universal:
  - 30000 pacotes binários no Debian
    $\rightarrow$ a maioria do software livre disponível é empacotado para Debian!
  - Para 12 portes (arquiteturas), incluindo 2 não-Linux (Hurd; KFreeBSD)
  - Também usado por 120 distribuições derivadas do Debian

O formato de pacotes Deb

- Arquivo `.deb`: um pacote `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`: versão do formato de arquivo deb, `"2.0\n"`
  - `control.tar.gz`: meta-dados sobre o pacote
    `control, md5sums, (pre|post)(rm|inst), triggers, shlibs,...`
  - `data.tar.gz`: arquivos de dados do pacote

- Você poderia criar os seus arquivos `.deb` manualmente
  `http://tldp.org/HOWTO/html_single/Debian-Binary-Package-Building-HOWTO/`

- Mas a maioria das pessoas não o faz dessa maneira

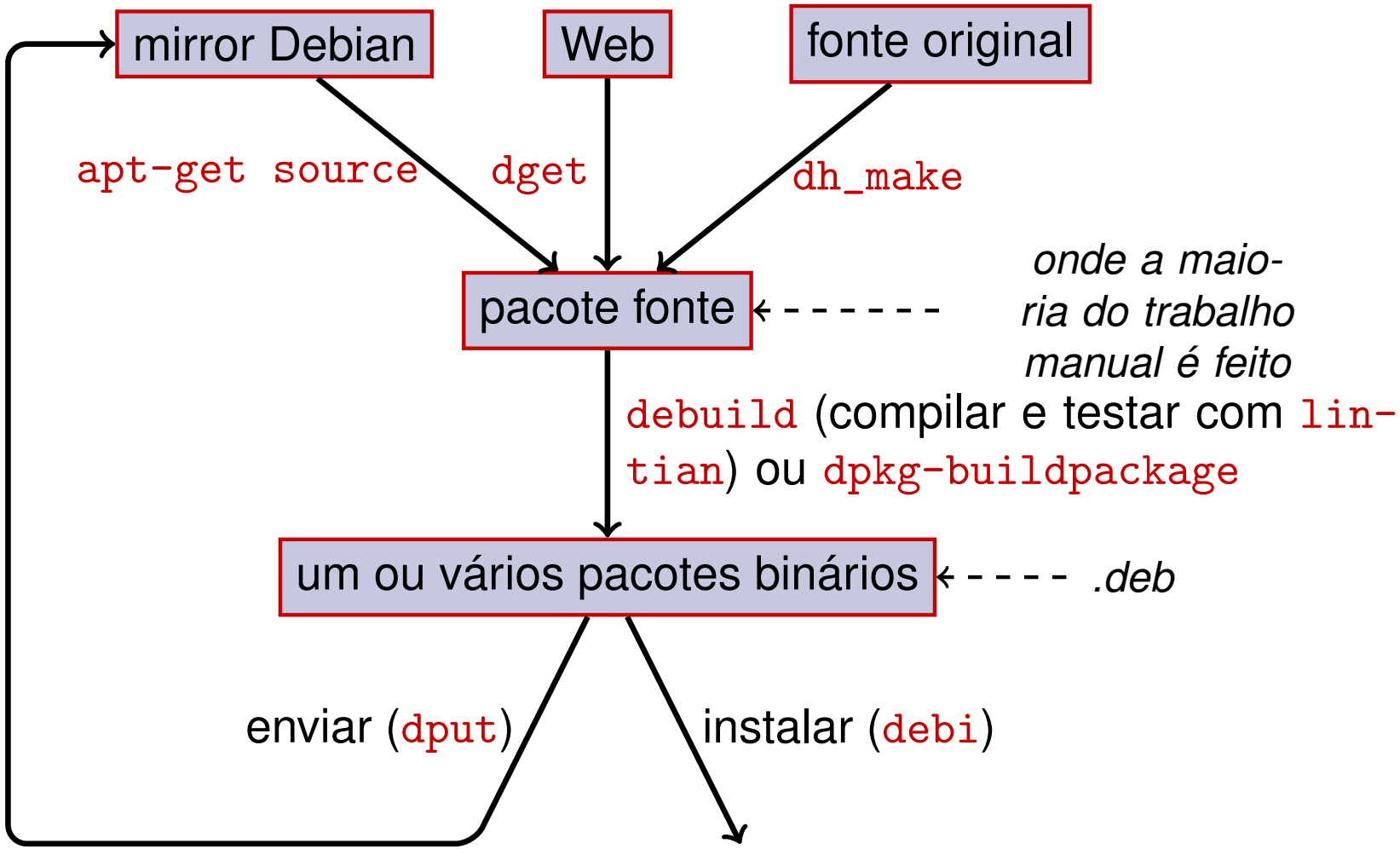
Neste tutorial: criar pacotes Debian, à maneira Debian

Ferramentas que você vai precisar

- Um sistema Debian (ou Ubuntu) (com acesso root)
- Alguns pacotes:
  - **build-essential**: tem dependências nos pacotes que serão assumidos estar disponíveis na máquina do desenvolvedor (não é preciso especificá-las no campo `Build-Depends:` do arquivo control do seu pacote)
    - Inclui uma dependência de **dpkg-dev**, o qual contém ferramentas básicas específicas do Debian para criar pacotes
  - **devscripts**: contém muitos scripts úteis para mantenedores Debian

Muitas outras ferramentas serão também mencionadas mais tarde, tais como **debhelper**, **cdbs**, **quilt**, **pbuilder**, **sbuild**, **lintian**, **svn-buildpackage**, **git-buildpackage**, . . .
Instale-as quando precisar delas.

Fluxo de trabalho de empacotamento geral



Exemplo: recompilando o dash

1. Instale os pacotes necessários para compilar o dash e devscripts
   `sudo apt-get build-dep dash`
   (requer linhas `deb-src` em `/etc/apt/sources.list`)
   `sudo apt-get install --no-install-recommends devscripts fakeroot`
2. Crie um diretório de trabalho, e vá para ele :
   `mkdir /tmp/debian-tutorial ; cd /tmp/debian-tutorial`
3. Obtenha o pacote fonte do `dash`
   `apt-get source dash`
   (Para isto você precisa ter linhas `deb-src` no seu `/etc/apt/sources.list`)
4. Compile o pacote
   `cd dash-*`
   `debuild -us -uc` (`-us -uc` desativa a assinatura do pacote com GPG)
5. Verifique que funcionou
   - Existem alguns arquivos `.deb` novos no diretório anterior
6. Observe o diretório `debian/`
   - É aí que o trabalho de empacotamento é feito

Sumário

Pacote fonte

- Um pacote fonte pode gerar vários pacotes binários
  p. ex. o fonte `libtar` gera os pacotes binários `libtar0` e `libtar-dev`
- Dois tipos de pacotes: (em caso de dúvida, use não-nativo)
  - Pacotes nativos: geralmente para software específico do Debian (*dpkg*, *apt*)
  - Pacotes não-nativos: software desenvolvido fora do Debian
- Arquivo principal: `.dsc` (meta-dados)
- Outros arquivos dependendo da versão do formato do fonte
  - 1.0 ou 3.0 (nativo): `package_version.tar.gz`
  - 1.0 (não-nativo):
    - `pkg_ver.orig.tar.gz`: fonte do original (upstream)
    - `pkg_debver.diff.gz`: patch para adicionar alterações específicas do Debian
  - 3.0 (quilt):
    - `pkg_ver.orig.tar.gz`: fonte do original (upstream)
    - `pkg_debver.debian.tar.gz`: tarball com alterações do Debian

(Veja `dpkg-source(1)` para detalhes exatos)

Exemplo de pacote fonte (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[..]1ee0e94248 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[..]9d1f2abf83 2464747 wget_1.12.orig.tar.gz
 e93123c934e3c[..]2f380278c2 48308 wget_1.12-2.1.debian.tar.gz
```

Obtendo um pacote fonte existente

- Do arquivo Debian:
  - `apt-get source` *`pacote`*
  - `apt-get source` *`pacote=versão`*
  - `apt-get source` *`pacote/lançamento`*

  (Você precisa de linhas `deb-src` em `sources.list`)

- Da Internet:
  - `dget` *`url-to.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` disponibiliza todos os pacotes Debian desde 2005)

- Do sistema de controle de versão (declarado):
  - `debcheckout` *`pacote`*

- Após baixado, extraia com `dpkg-source -x` *`file.dsc`*

Criar um pacote fonte básico

- Baixe o fonte original (upstream)
  (*fonte upstream* = aquele dos desenvolvedores originais do software)
- Renomeie para `<pacote_fonte>_<versao_original>.orig.tar.gz`
  (exemplo: `simgrid_3.6.orig.tar.gz`)
- Descompacte-o
- Renomeie o diretório para `<pacote_fonte>-<versao_original>`
  (exemplo: `simgrid-3.6`)
- `cd <pacote_fonte>-<versao_original> && dh_make`
  (do pacote **dh-make**)
- Existem algumas alternativas ao `dh_make` para conjuntos de pacotes específicos: **dh-make-perl**, **dh-make-php**, ...
- Diretório `debian/` criado, com muitos arquivos dentro dele

Arquivos em debian/

Todo o empacotamento deve ser feito modificando-se arquivos em `debian/`

- Arquivos principais:
  - **control** – meta-dados sobre o pacote (dependências, etc)
  - **rules** – especifica como compilar o pacote
  - **copyright** – informação de copyright para o pacote
  - **changelog** – história do pacote Debian
- Outros arquivos:
  - compat
  - watch
  - dh_install* targets
    *.dirs, *.docs, *.manpages, . . .
  - scripts do mantenedor
    *.postinst, *.prerm, . . .
  - source/format
  - patches/ – se você precisar modificar os fontes do autor original
- Vários arquivos usam formato baseado em RFC 822 (cabeçalhos de email)

debian/changelog

- Lista as alterações no empacotamento Debian
- Determina a versão atual do pacote

$$\underbrace{1.2.1.1}_{\text{Versão original}}\text{-}\underbrace{5}_{\text{Revisão Debian}}$$

- Editado manualmente ou com `dch`
  - Crie uma entrada no changelog para um novo lançamento: `dch -i`
- Formato especial para fechar automaticamente bugs do Debian ou Ubuntu. Debian: `Closes: #595268`; Ubuntu: `LP: #616929`
- Instalado como `/usr/share/doc/`*`pacote`*`/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

- Meta-dados do pacote
  - Para o próprio pacote fonte
  - Para cada pacote binário compilado deste fonte
- Nome do pacote, seção, prioridade, mantenedor, desenvolvedor que faz os uploads, dependências de compilação, dependências, descrição, página do projeto, ...
- Documentação: Debian Policy capítulo 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
```

Arquitetura: todas ou alguma

Dois tipos de pacotes binários:

- Pacotes com conteúdos diferentes para cada arquitectura Debian
  - Exemplo: programa C
  - `Architecture: any` em `debian/control`
    - Ou, se apenas funcionar num sub-conjunto de arquiteturas: `Architecture: amd64 i386 ia64 hurd-i386`
  - buildd.debian.org: compila para todas as outras arquiteturas no upload
  - Chamado *pacote*`_`*versão*`_`*arquitetura*`.deb`
- Pacotes com o mesmo conteúdo para todas as arquiteturas
  - Exemplo: biblioteca Perl
  - `Architecture: all` em `debian/control`
  - Chamado *pacote*`_`*versão*`_all.deb`

Um pacote fonte pode gerar uma mistura de pacotes binários de `Arquitecture: any` e `Arquitecture: all`

debian/rules

- Makefile
- Interface usada para compilar pacotes Debian
- Documentado na Debian Policy, capitulo 4.8
  `https://www.debian.org/doc/debian-policy/ch-source#s-debianrules`
- Alvos necessários:
  - `build, build-arch, build-indep`: deve executar toda a configuração e compilação
  - `binary, binary-arch, binary-indep`: compila os pacotes binários
    - `dpkg-buildpackage` vai chamar `binary` para compilar todos os pacotes, ou `binary-arch` para compilar apenas os pacotes de `Arquitecture: any`
  - `clean`: limpa o diretório fonte

Ajudantes de empacotamento – debhelper

- Você poderia escrever código de shell diretamente em `debian/rules`
  - Veja o pacote `rsync` como exemplo
- Melhor prática (mais popular): use um *Ajudante de empacotamento*
- O mais popular deles: **debhelper** (usado por 98% dos pacotes)
- Objetivos:
  - Divide tarefas comuns em ferramentas padrão usadas por todos os pacotes
  - Corrige bugs de empacotamento de uma vez para todos os pacotes

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, ...

  - Chamado a partir de `debian/rules`
  - Configurável usando parâmetros de comandos ou arquivos em `debian/`

package`.docs`, *package*`.exemplos`, *package*`.install`, *package*`.manpages`, ...

- Ajudantes de terceiros para conjuntos de pacotes: **python-support**, **dh_ocaml**, ...
- `debian/compat`: Versão de compatibilidade do Debhelper (use "7")

debian/rules usando 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 usando 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

- Com o debhelper, ainda bastante redundância entre pacotes
- Ajudantes de segundo-nível que dividem funcionalidades comuns
  - P. ex. compilando com `./configure && make && make install` ou CMake
- CDBS:
  - Introduzido em 2005, baseado na magia avançada do *GNU make*
  - Documentação: `/usr/share/doc/cdbs/`
  - Suporte para Perl, Python, Ruby, GNOME, KDE, Java, Haskell, . . .
  - Porém existem pessoas que o detestam:
    - Às vezes é difícil personalizar compilações de pacotes: "*labirinto enrolado de makefiles e variáveis de ambiente*"
    - Mais lento que o debhelper puro (muitas chamadas desnecessárias a `dh_*`)

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

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

Dh (aka Debhelper 7, ou dh7)

- Introduzido em 2008 como um *matador do CDBS*
- comando **dh** que chama `dh_*`
- *debian/rules* simples, listando apenas as sobreposições
- Mais fácil de personalizar que o CDBS
- Documentos: manpages (`debhelper(7)`, `dh(1)`) + slides da palestra na DebConf9
  `http://kitenet.net/~joey/talks/debhelper/debhelper-slides.pdf`

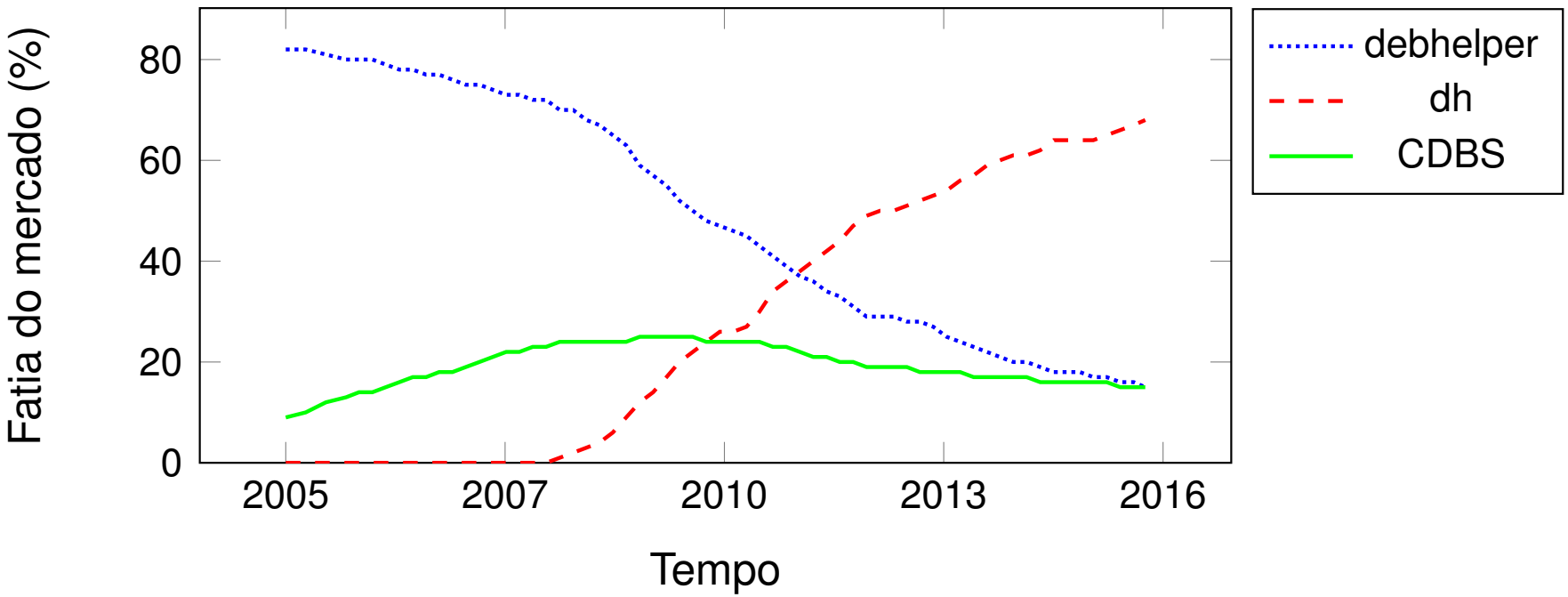
```
#!/usr/bin/make -f
%:
	dh $@

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

override_dh_auto_build:
	make world
```

debhelper clássico versus CDBS versus dh

- Popularidade:
  debhelper clássico: 15% CDBS: 15% dh: 68%
- Qual deles devo aprender?
  - Provavelmente um pouco de todos eles
  - Você precisa conhecer o debhelper para usar o dh e o CDBS
  - Você pode ter que modificar pacotes CDBS
- Qual deles devo usar para um pacote novo?
  - **dh** (única solução com um aumento de popularidade)



Sumário

Compilando pacotes

- `apt-get build-dep mypackage`
  Instala as *build-dependencies* (para um pacote já no Debian)
  Ou `mk-build-deps -ir` (para um pacote ainda não submetido)

- `debuild`: compila, testa com `lintian`, assina com GPG

- Também é possível chamar diretamente `dpkg-buildpackage`
  - Normalmente com `dpkg-buildpackage -us -uc`

- É melhor compilar os pacotes num ambiente limpo & mínimo
  - `pbuilder` – ajudante para compilar pacotes num *chroot*
    Boa documentação: `https://wiki.ubuntu.com/PbuilderHowto`
    (otimização: `cowbuilder ccache distcc`)
  - `schroot` e `sbuild`: usados nos daemons de compilação do Debian
    (não tão simples quanto `pbuilder`, mas permite snapshots LVM
    veja: `https://help.ubuntu.com/community/SbuildLVMHowto`)

- Gera arquivos `.deb` e um arquivo `.changes`
  - `.changes`: descreve o que foi compilado; usado para fazer o upload do pacote

Instalando e testando pacotes

- Instale o pacote localmente: `debi` (`.changes` vai dizer o que instalar)
- Liste o conteúdo do pacote: `debc ../mypackage<TAB>.changes`
- Compare o pacote com a versão anterior:
  `debdiff ../mypackage_1_*.changes ../mypackage_2_*.changes`
  ou para comparar os fontes:
  `debdiff ../mypackage_1_*.dsc ../mypackage_2_*.dsc`
- Verifique o pacote com `lintian` (analisador estático):
  `lintian ../mypackage<TAB>.changes`
  `lintian -i`: fornece mais informação sobre os erros
  `lintian -EviIL +pedantic`: mostra mais problemas
- Faça o upload do pacote para o Debian (`dput`) (precisa de configuração)
- Gerencie um repositório Debian privado com `reprepro` ou `aptly`
  Documentação:
  `https://wiki.debian.org/HowToSetupADebianRepository`

Sumário

Sessão prática 1: modificando o pacote grep

1. Visite `http://ftp.debian.org/debian/pool/main/g/grep/` e baixe a versão 2.12-2 do pacote
   - Se o pacote fonte não descompactar automaticamente, descompacte-o com `dpkg-source -x grep_*.dsc`
2. Observe os arquivos em `debian/`
   - Quantos pacotes binários são gerados por este pacote fonte?
   - Qual ajudante de empacotamento este pacote usa?
3. Compile o pacote
4. Agora vamos modificar o pacote. Adicione uma entrada changelog e incremente o número da versão.
5. Agora desative o suporte a perl-regexp (uma opção do `./configure`)
6. Re-compile o pacote
7. Compare os pacotes original e novo com o debdiff
8. instale o pacote que acaba de ser compilado

Sumário

debian/copyright

- Informação de copyright e licença para a fonte e o empacotamento
- Tradicionalmente escrito num arquivo de texto
- Novo formato legível por máquina:
  `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]
```

Modificando a fonte do original

Muitas vezes necessário:

- Corrigir bugs ou adicionar personalizações que são específicas do Debian
- Correções a versões anteriores (backport fixes) a partir de lançamento mais recente do autor original

Vários métodos para fazer isso:

- Modificando os arquivos diretamente
  - Simples
  - Mas não permite acompanhar e documentar as alterações
- Utilizando sistemas de patch
  - Facilita a contribuição das suas alterações para o autor original (upstream)
  - Ajuda a compartilhar as correções com os derivados do Debian
  - Dá mais visibilidade às alterações
    `http://patch-tracker.debian.org/` (no momento fora de serviço)

Sistemas de patch

- Princípio: as alterações são guardadas como patches em `debian/patches/`
- Aplicado e "des-aplicado" durante a compilação
- Passado: várias implementações – *simple-patchsys* (*cdbs*), *dpatch*, ***quilt***
  - Cada um suporta dois alvos `debian/rules`:
    - `debian/rules patch`: aplica todos os patches
    - `debian/rules unpatch`: retira as alterações de todos os patches
  - Mais documentação: `https://wiki.debian.org/debian/patches`
- **Novo formato de pacote fonte com sistema de patch integrado: 3.0 (quilt)**
  - Solução recomendada
  - Você precisa aprender *quilt* `http://perl-team.pages.debian.net/howto/quilt.html`
  - Ferramenta em `devscripts` independente do sistema de patch usado: `edit-patch`

Documentação de patches

- Cabeçalhos padrão no inicio do patch
- Documentado em DEP-3 - Patch Tagging Guidelines
  `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 {
```

Tomando ações durante instalação e remoção

- Descompactar o pacote às vezes não é suficiente
- Criar/remover usuários do sistema, iniciar/parar serviços, gerenciar *alternatives*
- Feito nos *scripts do mantenedor*
  `preinst, postinst, prerm, postrm`
  - Trechos de código para ações comuns podem ser gerados pelo debhelper
- Documentação:
  - Manual de Políticas Debian (Debian Policy), capítulo 6
    `https://www.debian.org/doc/debian-policy/ch-maintainerscripts`
  - Referência dos Desenvolvedores Debian, capítulo 6.4
    `https://www.debian.org/doc/developers-reference/best-pkging-practices.html`
  - `https://people.debian.org/~srivasta/MaintainerScripts.html`
- Questionando o usuário
  - Precisa ser feito com **debconf**
  - Documentação: `debconf-devel(7)` (pacote `debconf-doc`)

Monitorando versões do autor original (upstream)

- Especifique onde procurar em `debian/watch` (veja `uscan(1)`)

```
version=3

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

- Existem seguidores automáticos de novas versões do autor original, que notificam o mantenedor em vários painéis de controle incluindo `https://tracker.debian.org/` e `https://udd.debian.org/dmd/`
- `uscan`: executa uma verificação manual
- `uupdate`: tenta atualizar o seu pacote para a versão mais recente do autor original

Empacotando com Sistema de Controle de Versão

- Várias ferramentas para ajudar a gerenciar branches e tags: `svn-buildpackage`, `git-buildpackage`
- Exemplo: `git-buildpackage`
  - `upstream` branch que acompanha upstream com tags `upstream/`*`version`*
  - `master` branch que acompanha o pacote Debian
  - `debian/`*`version`* tags para cada envio (upload)
  - `pristine-tar` branch para a recompilação do tarball original

  Documento: `http://honk.sigxcpu.org/projects/git-buildpackage/manual-html/gbp.html`
- Campos `Vcs-*` em `debian/control` para localizar o repositório
  - `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
```

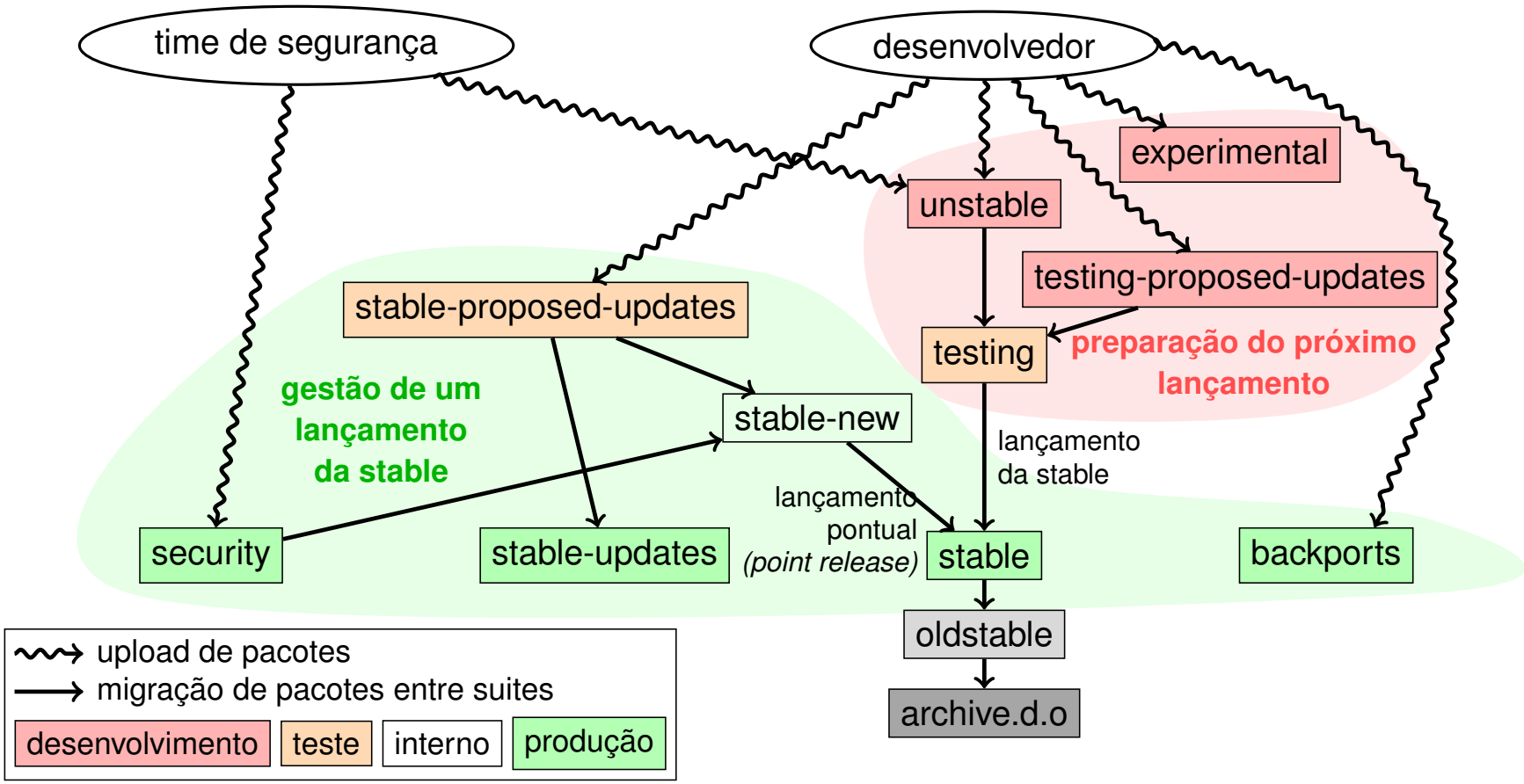
- Interface independente de VCS: `debcheckout`, `debcommit`, `debrelease`
  - `debcheckout grep` → obtém o pacote fonte do Git

Portando pacotes para trás (backporting)

- Objetivo: usar uma nova versão de um pacote num sistema mais antigo p.ex. usar *mutt* do Debian *unstable* no Debian *stable*
- Ideia geral:
  - Obtenha o pacote fonte do Debian unstable
  - Modifique-o para que compile e funcione bem no Debian stable
    - Às vezes isso é trivial (sem alterações necessárias)
    - Às vezes é difícil
    - Às vezes é impossível (muitas dependências não disponíveis)
- Alguns "backports" são disponibilizados e suportados pelo projeto Debian `http://backports.debian.org/`

Sumário

Repositório e suites Debian



Baseado no grafo de Antoine Beaupré. `https://salsa.debian.org/debian/package-cycle`

Suites de desenvolvimento

- Novas versões de pacotes são enviadas para **unstable** (**sid**)
- Pacotes migram da **unstable** para **testing** com base em diversos critérios (p.ex. estar na unstable por 10 dias, e sem regressão)
- Novos pacotes também podem ser submetidos para:
  - **experimental** (para pacotes mais *experimentais*, como quando a nova versão não está pronta para substituir a que está na unstable)
  - **testing-proposed-updates**, para atualizar a versão na **testing** sem passar pela **unstable** (raramente usado)

Congelando e lançando

- Em algum momento do ciclo, o time de lançamento (*release* team) decide congelar (*freeze*) a testing: migrações automáticas da **unstable** para **testing** são paradas e substituídas por revisão manual
- Quando o time de release considera a **testing** pronta para lançamento:
  - A suite **testing** torna-se a nova suite **stable**
  - Similarmente, a que era **stable** torna-se **oldstable**
  - Lançamentos não mais mantidos são movidos para `archive.debian.org`
- Veja `https://release.debian.org/`

Gestão de lançamento da stable

- Diversas suites proveem pacotes para o lançamento da stable:
  - **stable**: a suite principal
  - suite de atualizações de **segurança** disponibilizada em `security.debian.org`, usada pelo time de segurança. Atualizações são anunciadas na lista de discussão `debian-security-announce`
  - **stable-updates**: atualizações que não de segurança, mas que deveriam ser instaladas urgentemente (sem esperar pelo próximo lançamento pontual (*stable point release*): banco de dados de anti-virus, pacotes relacionados a fuso horário, etc. Anunciadas na lista de discussão `debian-stable-announce`
  - **backports**: novas versões do original, baseadas na versão em **testing**
- A suite **stable** é atualizada a cada poucos meses por lançamentos pontuais (*point releases*) que incluem apenas correções de bugs
  - Pacotes cujo alvo é o próximo lançamento pontual são enviados para **stable-proposed-updates** e revisados pelo time de lançamento
- A versão **oldstable** tem o mesmo conjunto de suites

Várias maneiras de contribuir para Debian

- **Pior** maneira de contribuir:
  1. Empacotar a sua própria aplicação
  2. Colocar ela no Debian
  3. Desaparecer

- **Melhores** maneiras de contribuir:
  - Envolver-se com as equipes de empacotamento
    - Muitas equipes focam em conjuntos de pacotes, e precisam de ajuda
    - Lista disponível em `https://wiki.debian.org/Teams`
    - Uma excelente maneira de aprender de contribuidores mais experientes
  - Adotar pacotes existentes não mantidos (*pacotes órfãos*)
  - Trazer novo software para o Debian
    - Apenas se for suficientemente interessante/útil, por favor
    - Existem alternativas já empacotadas no Debian?

Adotando pacotes órfãos

- Muitos pacotes não mantidos no Debian
- Lista completa + processo: `https://www.debian.org/devel/wnpp/`
- Instalados na sua máquina: `wnpp-alert`
  Ou melhor: `how-can-i-help`
- Estados diferentes:
  - **O**rphaned: o pacote não é mantido. Sinta-se livre para o adotar.
  - **RFA**: **R**equest **F**or **A**dopter
    O mantenedor procura quem adote, mas continua a trabalhar enquanto isso
    Sinta-se livre para adotar. É cordial enviar um mail ao mantenedor atual.
  - **ITA**: **I**ntent **T**o **A**dopt
    Alguém tem a intenção de adotar o pacote. Você pode oferecer ajuda!
  - **RFH**: **R**equest **F**or **H**elp
    O mantenedor procura ajuda.
- Alguns pacotes não mantidos e não detectados $\rightarrow$ ainda não estão órfãos
- Quando em dúvidas, pergunte a `debian-qa@lists.debian.org`
  ou `#debian-qa` em `irc.debian.org`

Adoptando um pacote: exemplo

```
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
```

- Seja cortês ao contactar o mantenedor anterior (especialmente se o pacote estava em RFA, não órfão)
- É uma boa ideia contactar o projeto original

Colocando o seu pacote no Debian

- Você não precisa de nenhum status oficial para ter o seu pacote no Debian
  1. Submeta um bug **ITP** (**I**ntent **T**o **P**ackage) usando `reportbug wnpp`
  2. Prepare um pacote fonte
  3. Encontre um Desenvolvedor Debian que apadrinhe o seu pacote (*sponsor*)

- Status oficial (quando você é um mantenedor de pacotes experiente)
  - **Mantenedor Debian (DM):**
    Permissão para submeter os seus próprios pacotes
    Veja `https://wiki.debian.org/DebianMaintainer`
  - **Desenvolvedor Debian (DD):**
    Membro do projeto Debian; pode votar e enviar (upload) qualquer pacote

A verificar antes de pedir apadrinhamento

- Debian tem **muita atenção à qualidade**
- Geralmente, os **padrinhos são difíceis de encontrar e ocupados**
  - Certifique-se de que seu pacote está pronto antes de pedir apadrinhamento
- Coisas a verificar:
  - Evite a falta de dependências de compilação: certifique-se de que seu pacote compila bem num *chroot sid* limpo
    - É recomendado usar o `pbuilder`
  - Rode `lintian -EviIL +pedantic` no seu pacote
    - Os erros têm de ser corrigidos, todos os outros problemas devem ser corrigidos
  - E claro, faça testes abrangentes no seu pacote
- Em dúvida, peça ajuda

Onde encontrar ajuda?

Ajuda que você vai precisar:

- Conselhos e respostas para as suas perguntas, revisões de código
- Padrinhos para os seus uploads, quando o seu pacote estiver pronto

Você pode obter ajuda de:

- **Outros membros de uma equipe de empacotamento**
  - Lista de equipes: `https://wiki.debian.org/Teams`
- O grupo **Debian Mentors** (se o seu pacote não se encaixar numa equipe)
  - `https://wiki.debian.org/DebianMentorsFaq`
  - Lista de email: `debian-mentors@lists.debian.org` (também uma boa maneira de aprender por acaso)
  - IRC: `#debian-mentors` em `irc.debian.org`
  - `http://mentors.debian.net/`
  - Documentação: `http://mentors.debian.net/intro-maintainers`
- **Listas de email localizadas** (obtenha ajuda no seu idioma)
  - `debian-devel-{french,italian,portuguese,spanish}@lists.d.o`
  - Lista completa: `https://lists.debian.org/devel.html`
  - Ou listas de usuários: `https://lists.debian.org/users.html`

Mais documentação

- O Canto dos Desenvolvedores Debian
  `https://www.debian.org/devel/`
  Links para muitos recursos sobre o desenvolvimento do Debian
- Guia para Mantenedores Debian
  `https://www.debian.org/doc/manuals/debmake-doc/`
- Referência dos Desenvolvedores Debian
  `https://www.debian.org/doc/developers-reference/`
  Majoritariamente sobre procedimentos no Debian, mas também algumas melhores práticas de empacotamento (parte 6)
- Política Debian
  `https://www.debian.org/doc/debian-policy/`
  - Todos os requerimentos que cada pacote deve satisfazer
  - Políticas específicas para Perl, Java, Python, ...
- Guia de Empacotamento Ubuntu
  `http://developer.ubuntu.com/resources/tools/packaging/`

Paineis de controle do Debian para mantenedores

- **Centrado no pacote fonte**:
  `https://tracker.debian.org/dpkg`
- **Centrado no mantenedor/equipe**: Visão Geral de Pacotes de Desenvolvedores (DDPO)
  `https://qa.debian.org/developer.php?login=pkg-ruby-extras-maintainers@lists.alioth.debian.org`
- **Orientado a Lista-A-FAZER**: Painel de Controle do Mantenedor Debian (DMD)
  `https://udd.debian.org/dmd/`

Usando o Debian Bug Tracking System (BTS)

- Uma maneira única de gerenciar bugs
  - Interface web para ver os bugs
  - Interface de email para fazer alterações nos bugs
- Adicionando informação aos bugs:
  - Escreva para `123456@bugs.debian.org` (não inclui a pessoa que submeteu, você precisa adicionar `123456-submitter@bugs.debian.org`)
- Alterando o estado do bug:
  - Envie comandos para `control@bugs.debian.org`
  - Interface de linha de comando: comando `bts` em `devscripts`
  - Documentação: `https://www.debian.org/Bugs/server-control`
- Reportando bugs: use `reportbug`
  - Normalmente usado com um servidor de email local: instale `ssmtp` ou `nullmailer`
  - Ou use `reportbug --template`, depois envie (manualmente) para `submit@bugs.debian.org`

Usando o BTS: exemplos

- Enviando um email para o bug e para quem o submeteu:
  `https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680822#10`
- Etiquetando e alterando a severidade:
  `https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680227#10`
- Re-atribuindo, alterando a severidade, mudando o título . . . :
  `https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680822#93`
  - `notfound`, `found`, `notfixed`, `fixed` são para **acompanhamento de versão**
    Veja `https://wiki.debian.org/HowtoUseBTS#Version_tracking`
- Usando usertags:
  `https://bugs.debian.org/cgi-bin/bugreport.cgi?msg=42;bug=642267`
  Veja `https://wiki.debian.org/bugs.debian.org/usertags`
- Documentação do BTS:
  - `https://www.debian.org/Bugs/`
  - `https://wiki.debian.org/HowtoUseBTS`

Mais interessado em Ubuntu?

- Ubuntu majoritariamente gerencia a divergência com o Debian
- Nenhum foco real em pacotes específicos
  Em vez disso, colaboração com as equipes do Debian
- Normalmente é recomendado enviar os novos pacote primeiro para o Debian
  `https://wiki.ubuntu.com/UbuntuDevelopment/NewPackages`
- Possivelmente um plano melhor:
  - Envolva-se numa equipe do Debian e atue como uma ponte com o Ubuntu
  - Ajude a reduzir a divergência, triagem de bugs no Launchpad
  - Muitas ferramentas do Debian podem ajudar:
    - Coluna Ubuntu na visão geral de pacotes de Desenvolvedores
    - Caixa do Ubuntu no Sistema de Acompanhamento de Pacotes
    - Receba bugmail do launchpad via PTS

Sumário

Conclusões

- Agora você tem uma visão geral do empacotamento Debian
- Mas você vai precisar ler mais documentação
- As melhores práticas evoluíram com os anos
  - Em dúvida, use o ajudante de empacotamento **dh**, e o formato **3.0 (quilt)**

Comentários: **packaging-tutorial@packages.debian.org**

Questões legais

Contribua para este manual

- Contribuir:
  - `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`
  - Bugs abertos: `bugs.debian.org/src:packaging-tutorial`
- Envie sugestões:
  - `mailto:packaging-tutorial@packages.debian.org`
    - O que deve ser adicionado a este manual?
    - O que deve ser melhorado?
  - `reportbug packaging-tutorial`

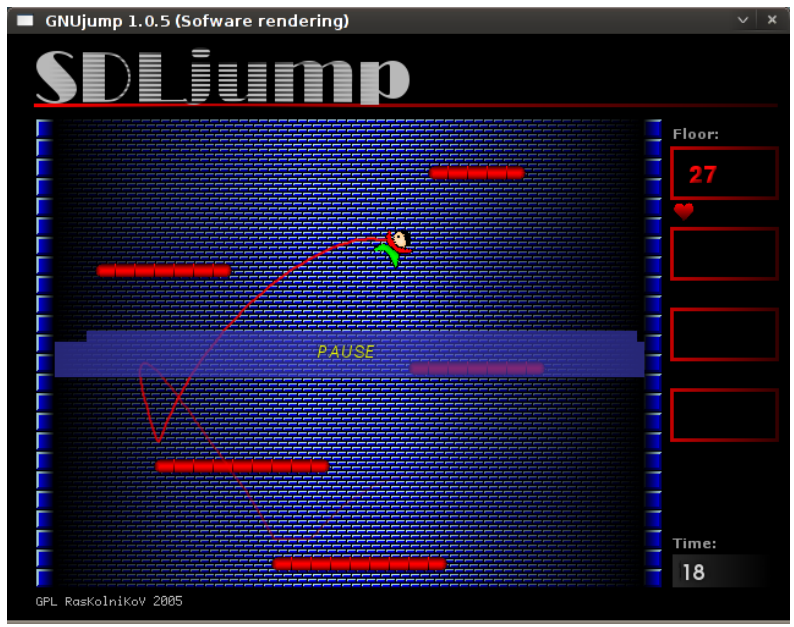
Sumário

Sessão prática 2: empacotando o GNUjump

1. Faça o download de GNUjump 1.0.8 de
   `http://ftp.gnu.org/gnu/gnujump/gnujump-1.0.8.tar.gz`

2. Crie um pacote Debian para ele
   - Instale as dependências de compilação para poder compilar o pacote
   - Corrija bugs
   - Obtenha um pacote funcional básico
   - Termine de preencher `debian/control` e outros arquivos

3. Aprecie



Sessão prática 2: empacotando o GNUjump (dicas)

- Para obter um pacote básico funcional, use `dh_make`
- No começo, criar um pacote fonte *1.0* é mais fácil do que um *3.0 (quilt)* (mude isso em `debian/source/format`)
- Para procurar dependências de compilação faltando, encontre um arquivo em falta, e use o `apt-file` para encontrar o pacote em falta.
- Se você encontrar esse erro:

```
/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
```

  Você precisa adicionar `-lm` à linha de comando do linker: Edite `src/Makefile.am` e substitua

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

  por

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

  Depois rode `autoreconf -i`

Sessão prática 3: empacotar uma biblioteca Java

1. Faça uma leitura rápida em alguma documentação sobre empacotamento 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`
2. Baixe o IRClib de `http://moepii.sourceforge.net/`
3. Empacote-o

Sessão prática 4: empacotando um pacote Ruby

1. Dê uma lida rápida em alguma documentação sobre empacotamento Ruby:
   - `https://wiki.debian.org/Ruby`
   - `https://wiki.debian.org/Teams/Ruby`
   - `https://wiki.debian.org/Teams/Ruby/Packaging`
   - `gem2deb(1)`, `dh_ruby(1)` (no pacote `gem2deb`)
2. Crie um pacote fonte Debian básico a partir do pacote ruby (*gem*) `peach`: `gem2deb peach`
3. Melhore-o para que se torne um pacote Debian apropriado

Sessão prática 5: empacotar um módulo Perl

1. Faça uma leitura rápida em alguma documentação sobre empacotamento Perl:
   - `https://perl-team.pages.debian.net`
   - `https://wiki.debian.org/Teams/DebianPerlGroup`
   - `dh-make-perl(1)`, `dpt(1)` (no pacote `pkg-perl-tools`)
2. Crie um pacote fonte Debian básico a partir da distribuição CPAN `Acme`: `dh-make-perl --cpan Acme`
3. Melhore-o para que se torne um pacote Debian apropriado

Sumário

Respostas às sessões práticas

Sessão prática 1: modificando o pacote grep

1. Visite `http://ftp.debian.org/debian/pool/main/g/grep/` e baixe a versão 2.12-2 do pacote
2. Observe os arquivos em `debian/`
   - Quantos pacotes binários são gerados por este pacote fonte?
   - Qual ajudante de empacotamento este pacote usa?
3. Compile o pacote
4. Agora vamos modificar o pacote. Adicione uma entrada changelog e incremente o número da versão.
5. Agora desative o suporte a perl-regexp (uma opção do `./configure`)
6. Re-compile o pacote
7. Compare os pacotes original e novo com o debdiff
8. instale o pacote que acaba de ser compilado

Obtendo o fonte

1. Visite `http://ftp.debian.org/debian/pool/main/g/grep/` e baixe a versão 2.12-2 do pacote

- Use dget para baixar o arquivo `.dsc`:
  `dget http://cdn.debian.net/debian/pool/main/g/grep/grep_2.12-2.dsc`
- Se você tiver `deb-src` para um lançamento Debian que tem `grep` versão 2.12-2 (descubra em `https://tracker.debian.org/grep`), você pode usar `apt-get source grep=2.12-2`
  ou `apt-get source grep/release` (p.ex. `grep/stable`)
  ou, se tiver com sorte: `apt-get source grep`
- O pacote fonte do `grep` é composto por três arquivos:
  - `grep_2.12-2.dsc`
  - `grep_2.12-2.debian.tar.bz2`
  - `grep_2.12.orig.tar.bz2`

  Isto é típico do formato "3.0 (quilt)".
- Se necessário, descompacte o fonte com
  `dpkg-source -x grep_2.12-2.dsc`

Observando e compilando o pacote

2. Observe os arquivos em `debian/`
   - Quantos pacotes binários são gerados por este pacote fonte?
   - Qual ajudante de empacotamento este pacote usa?

- De acordo com `debian/control`, este pacote gera apenas um pacote binário, chamado `grep`.
- De acordo com `debian/rules`, este pacote é típico de empacotamento debhelper *clássico*, sem usar *CDBS* ou *dh*. Pode-se ver as várias chamadas a comandos `dh_*` em `debian/rules`.

3. Compile o pacote

- Use `apt-get build-dep grep` para obter as dependências de compilação
- Depois `debuild` ou `dpkg-buildpackage -us -uc` (Demora cerca de 1 minuto)

Editando o registro de alterações (*changelog*)

4. Agora vamos modificar o pacote. Adicione uma entrada changelog e incremente o número da versão.

- `debian/changelog` é um arquivo de texto. Você pode editá-lo e adicionar uma nova entrada manualmente.
- Ou você pode usar `dch -i`, que irá adicionar uma entrada e abrir o editor
- O nome e email podem ser definidos usando as variáveis de ambiente `DEBFULLNAME` e `DEBEMAIL`
- Em seguida, recompile o pacote: uma nova versão do pacote é construída
- O "versionamento" do pacote está detalhado na seção 5.6.12 da política Debian
  `https://www.debian.org/doc/debian-policy/ch-controlfields`

Desativando suporte regexp de Perl e recompilando

5. Agora desative o suporte a perl-regexp (uma opção do `./configure`)
6. Re-compile o pacote

- Verifique com `./configure --help`: a opção para desativar Perl regexp é `--disable-perl-regexp`
- Edite `debian/rules` e encontre a linha do `./configure`
- Adicione `--disable-perl-regexp`
- Recompile com `debuild` ou `dpkg-buildpackage -us -uc`

Comparando e testando os pacotes

7. Compare os pacotes original e novo com o debdiff
8. instale o pacote que acaba de ser compilado

- Compare os pacotes binários: `debdiff ../*changes`
- Compare os pacotes fonte: `debdiff ../*dsc`
- Instale o pacote recentemente compilado: `debi`
  Ou `dpkg -i ../grep_<TAB>`
- `grep -P foo` não funciona mais!

Reinstale a versão anterior do pacote:

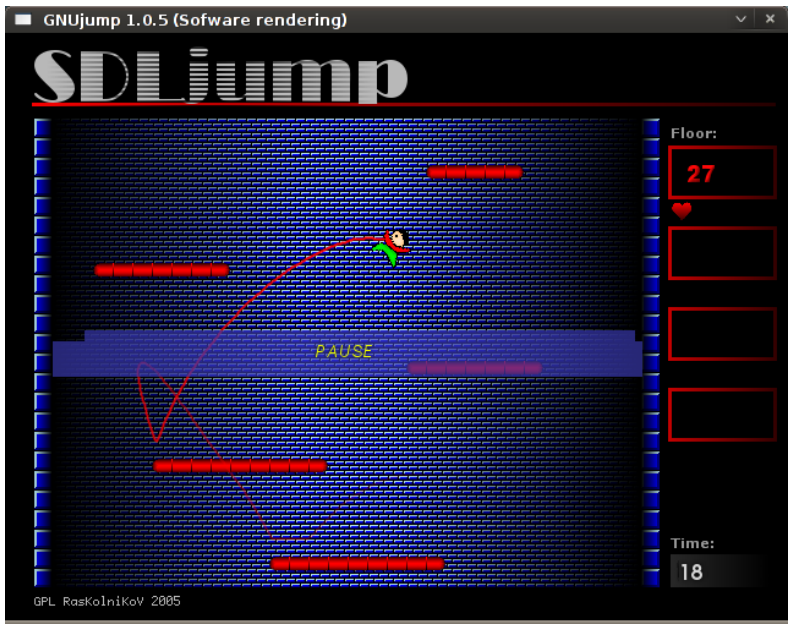
- `apt-get install --reinstall grep=2.6.3-3` *(= versão anterior)*

Sessão prática 2: empacotando o GNUjump

1. Faça o download de GNUjump 1.0.8 de
   `http://ftp.gnu.org/gnu/gnujump/gnujump-1.0.8.tar.gz`

2. Crie um pacote Debian para ele
   - Instale as dependências de compilação para poder compilar o pacote
   - Obtenha um pacote funcional básico
   - Termine de preencher `debian/control` e outros arquivos

3. Aprecie



Passo a passo...

- `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`
  - Tipo de pacote: binário simples (por agora)

```
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
```

Passo a passo. . . (2)

- Observe `debian/changelog`, `debian/rules`, `debian/control` (auto-preenchido por **dh_make**)
- Em `debian/control`:
  `Build-Depends: debhelper (>= 7.0.50 ), autotools-dev`
  Lista as *build-dependencies* = pacotes necessários para compilar o pacote
- Tente compilar o pacote como ele está com `debuild` (graças à magia do **dh**)
  - E adicione as dependências de compilação, até que compile
  - Dica: use `apt-cache search` e `apt-file` para encontrar os pacotes
  - Exemplo:

```
checking for sdl-config... no
checking for SDL - version >= 1.2.0... no
[...]
configure: error: *** SDL version 1.2.0 not found!
```

    → Adicione **libsdl1.2-dev** a Build-Depends e instale-o.
  - Melhor: use **pbuilder** para compilar num ambiente limpo

Passo a passo. . . (3)

- As dependências de compilação necessárias são `libsdl1.2-dev`, `libsdl-image1.2-dev`, `libsdl-mixer1.2-dev`
- Depois, você irá provavelmente ao encontro de outro erro:

```
/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
```

- Este problema é causado pelo bitrot: O gnujump não foi ajustado seguindo as alterações do linker.
- Se você estiver usando o formato de fonte versão **1.0** você pode mudar diretamente as fontes do autor.
  - Edite `src/Makefile.am` e substitua

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

    por

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

  - Depois rode `autoreconf -i`

Passo a passo...(4)

- Se estiver usando formato de fonte versão **3.0 (quilt)**, use `quilt` para preparar um patch. (veja `https://wiki.debian.org/UsingQuilt`)
  - `export QUILT_PATCHES=debian/patches`
  - `mkdir debian/patches`
    `quilt new linker-fixes.patch`
    `quilt add src/Makefile.am`
  - Edite `src/Makefile.am` e substitua

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

    por

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

  - `quilt refresh`
  - Desde que o `src/Makefile.am` mudou, o autoreconf tem que ser chamado durante a compilação. Para fazer isso automaticamente com `dh`, altere a chamada `dh` em `debian/rules` de: `dh $ --with autotools-dev`
    para: `dh $ --with autotools-dev --with autoreconf`

Passo a passo... (5)

- O pacote agora deve compilar sem problemas.
- Use `debc` para listar o conteúdo do pacote gerado, e `debi` para o instalar e testar.
- Teste o pacote com `lintian`
  - Embora não seja um requerimento estrito, é recomendado que os pacotes enviados para o Debian sejam *lintian-clean* (passem no teste do lintian)
  - Mais problemas podem ser listados usando `lintian -EviIL +pedantic`
  - Algumas dicas:
    - Remova os arquivos que você não precisa em `debian/`
    - Preencha `debian/control`
    - Instale o executável em `/usr/games` passando por cima do `dh_auto_configure`
    - Use marcadores *hardening* do compilador para aumentar a segurança.
      Veja `https://wiki.debian.org/Hardening`

Passo a passo. . . (6)

- Compare o seu pacote com aquele já empacotado no Debian:
  - Ele separa os arquivos de dados para um segundo pacote, que é o mesmo para todas as arquiteturas ($\rightarrow$ poupa espaço no repositório Debian)
  - Instala um arquivo .desktop (para os menus de GNOME/KDE) e também o integra ao menu Debian
  - Corrige alguns problemas menores usando patches

Sessão prática 3: empacotar uma biblioteca Java

1. Faça uma leitura rápida em alguma documentação sobre empacotamento 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`
2. Baixe o IRClib de `http://moepii.sourceforge.net/`
3. Empacote-o

Passo a passo...

- `apt-get install javahelper`
- Crie um pacote fonte básico: `jh_makepkg`
  - Biblioteca
  - Nenhum
  - Compilador/runtime livre padrão
- Observe e corrija `debian/*`
- `dpkg-buildpackage -us -uc` ou `debuild`
- `lintian`, `debc`, etc.
- Compare o seu resultado com o pacote fonte `libirclib-java`

Sessão prática 4: empacotando um pacote Ruby

1. Dê uma lida rápida em alguma documentação sobre empacotamento Ruby:
   - `https://wiki.debian.org/Ruby`
   - `https://wiki.debian.org/Teams/Ruby`
   - `https://wiki.debian.org/Teams/Ruby/Packaging`
   - `gem2deb(1)`, `dh_ruby(1)` (no pacote `gem2deb`)
2. Crie um pacote fonte Debian básico a partir do pacote ruby (*gem*) `peach`: `gem2deb peach`
3. Melhore-o para que se torne um pacote Debian apropriado

Passo a passo...

`gem2deb peach`:

- Baixa o pacote (*gem*) de rubygems.org
- Cria um arquivo .orig.tar.gz apropriado e descompacta-o
- Inicializa um pacote fonte Debian baseado nos meta-dados do gem
  - Chamado `ruby-`*`gemname`*
- Tenta compilar o pacote binário Debian (pode falhar)

`dh_ruby` (incluído em *gem2deb*) faz as tarefas específicas de Ruby:

- Compila extensões de C para cada versão de Ruby
- Copia os arquivos para o seu diretório de destino
- Atualiza shebangs nos scripts executáveis
- Roda os testes definidos em `debian/ruby-tests.rb`, `debian/ruby-tests.rake`, ou `debian/ruby-test-files.yaml`, assim como várias outras verificações

Passo a passo. . . (2)

Melhore o pacote gerado

- Rode `debclean` para limpar a árvore fonte. Observe `debian/`
- `changelog` e `compat` devem estar corretos
- Edite `debian/control`: melhore o campo `Description`
- Escreva um arquivo `copyright` apropriado com base nos arquivos do autor
- Compile o pacote
- Compare o seu pacote com o pacote `ruby-peach` no repositório Debian

Sessão prática 5: empacotar um módulo Perl

1. Faça uma leitura rápida em alguma documentação sobre empacotamento Perl:
   - `https://perl-team.pages.debian.net`
   - `https://wiki.debian.org/Teams/DebianPerlGroup`
   - `dh-make-perl(1)`, `dpt(1)` (no pacote `pkg-perl-tools`)
2. Crie um pacote fonte Debian básico a partir da distribuição CPAN `Acme`: `dh-make-perl --cpan Acme`
3. Melhore-o para que se torne um pacote Debian apropriado

Passo a passo...

`dh-make-perl --cpan Acme`:

- Baixa o tarball a partir de CPAN
- Cria um arquivo .orig.tar.gz apropriado e descompacta-o
- Inicializa um pacote fonte Debian baseado nos meta-dados da distribuição
  - Chamado `lib`*`distname`*`-perl`

Passo a passo. . . (2)

Melhore o pacote gerado

- `debian/changelog`, `debian/compat`, `debian/libacme-perl.docs`, e `debian/watch` devem estar corretos
- Edite `debian/control`: melhore o campo `Description`, e remova o texto padrão no final
- Edite `debian/copyright`: remova o parágrafo de texto padrão no topo, adicione anos de copyright à estrofe de `Files: *`

Tradução

Este tutorial foi traduzido por Tássia Camões Araújo e Leandro Luiz Pereira, usando a tradução de Américo Monteiro (português de Portugal) como ponto de partida.

Se você encontrar algum erro na tradução deste documento, por favor entre em contato com <tassia@debian.org>, <leandro@fullonmorning.com> ou <debian-l10n-portuguese@lists.debian.org>.