

# Manual de Empacotamento de Debian

Lucas Nussbaum

`packaging-tutorial@packages.debian.org`

version 0.13 – 2014-06-29



# Acerca deste manual

- ▶ Objectivo: **dizer o que você precisa mesmo saber sobre empacotamento de Debian**
  - ▶ Modificar pacotes existentes
  - ▶ Criar os seus próprios pacotes
  - ▶ Interagir com a comunidade Debian
  - ▶ Tornar-se um utilizador avançado de Debian
- ▶ Cobre os pontos mais importantes, mas não é completo
  - ▶ Você irá precisar de ler mais documentação
- ▶ A maioria do conteúdo também se aplica a distribuições derivadas da Debian
  - ▶ Isso inclui Ubuntu



# Esboço

- 1 Introdução
- 2 Criar pacotes fonte
- 3 Compilando e testando pacotes
- 4 Sessão prática 1: modificar o pacote grep
- 5 Tópicos de empacotamento avançados
- 6 Mantendo pacotes em Debian
- 7 Conclusões
- 8 Sessão prática 2: empacotar o GNUjump
- 9 Sessão prática 3: empacotando uma biblioteca Java
- 10 Sessão prática 4: empacotar uma gema Ruby
- 11 Respostas às sessões práticas



# Esboço

- 1 Introdução
- 2 Criar pacotes fonte
- 3 Compilando e testando pacotes
- 4 Sessão prática 1: modificar o pacote grep
- 5 Tópicos de empacotamento avançados
- 6 Mantendo pacotes em Debian
- 7 Conclusões
- 8 Sessão prática 2: empacotar o GNUjump
- 9 Sessão prática 3: empacotando uma biblioteca Java
- 10 Sessão prática 4: empacotar uma gema Ruby
- 11 Respostas às sessões práticas



# Debian

- ▶ **Distribuição de GNU/Linux**
- ▶ 1ª grande distribuição desenvolvida "abertamente ao espírito de GNU"
- ▶ **Não-comercial**, construída em colaboração por mais de 1000 voluntários
- ▶ 3 funcionalidades principais:
  - ▶ **Qualidade** – cultura de excelência técnica  
*Nós lançamos quando está tudo pronto*
  - ▶ **Liberdade** – desenvolvedores e utilizadores unidos pelo *Contracto Social*  
Promovendo a cultura do Software Livre desde 1993
  - ▶ **Independência** – nenhuma (nem uma) companhia toma conta da Debian  
E processo de decisão-trabalho aberto (*do-ocracy + democracy*)
- ▶ **Amador** no melhor sentido: feito com amor



# Pacotes Debian

- ▶ ficheiros **.deb** (pacotes binários)
- ▶ Uma maneira muito poderosa e conveniente de distribuir software aos utilizadores
- ▶ Um dos dois formatos de pacotes mais comuns (com o RPM)
- ▶ Universal:
  - ▶ 30000 pacotes binários em Debian  
→ a maioria do software livre disponível está empacotado em Debian!
  - ▶ Para 12 portes (arquitecturas), incluindo 2 não-Linux (Hurd; KFreeBSD)
  - ▶ Também usado por 120 distribuições derivadas de Debian



# O formato de pacotes Deb

- ▶ Ficheiro .deb: um arquivo 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 ficheiro deb, "2.0\n"
  - ▶ control.tar.gz: meta-dados acerca do pacote  
control, md5sums, (pre|post)(rm|inst), triggers, shlibs,...
  - ▶ data.tar.gz: ficheiros de dados do pacote
- ▶ Você poderia criar os seus ficheiros .deb manualmente  
[http://tldp.org/HOWTO/html\\_single/Debian-Binary-Package-Building-HOWTO/](http://tldp.org/HOWTO/html_single/Debian-Binary-Package-Building-HOWTO/)
- ▶ Mas a maioria das pessoas não o faz dessa maneira

**Este manual: criar pacotes Debian, à maneira Debian**



# Ferramentas que irá precisar

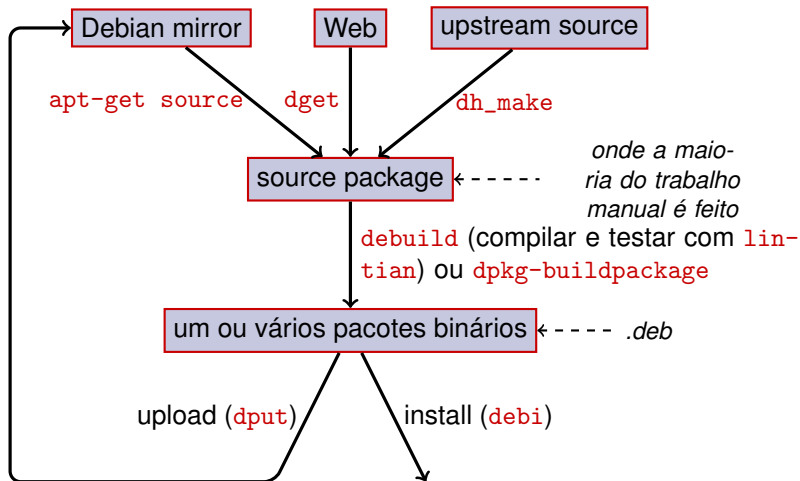
- ▶ Um sistema Debian (ou Ubuntu) (com acesso a root)
- ▶ Alguns pacotes:
  - ▶ **build-essential**: tem dependências nos pacotes que irão ser assumidas para estarem disponíveis na máquina do desenvolvedor (não é preciso especificá-las no campo de controle Build-Depends: do seu pacote)
    - ▶ Inclui a dependência de **dpkg-dev**, a qual contém ferramentas básicas específicas de Debian para criar pacotes
  - ▶ **devscripts**: contém muitos scripts úteis para mantenedores de Debian

Muitas outras ferramentas serão também mencionadas mais tarde, tais como **debhelper**, **cdb**s, **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 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 directório 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 precisa de ter linhas deb-src no seu /etc/apt/sources.list)
- 4 Compile o pacote  

```
cd dash-*
```

```
debuild -us -uc
```

(-us -uc desactiva a assinatura do pacote com GPG)
- 5 Verifique que funcionou
  - ▶ Existem alguns ficheiros .deb novos no directório pai
- 6 Observe o directório debian/
  - ▶ É onde o trabalho de empacotamento é feito



# Esboço

- 1 Introdução
- 2 Criar pacotes fonte
- 3 Compilando e testando pacotes
- 4 Sessão prática 1: modificar o pacote grep
- 5 Tópicos de empacotamento avançados
- 6 Mantendo pacotes em Debian
- 7 Conclusões
- 8 Sessão prática 2: empacotar o GNUjump
- 9 Sessão prática 3: empacotando uma biblioteca Java
- 10 Sessão prática 4: empacotar uma gema Ruby
- 11 Respostas às sessões práticas



# Pacote fonte

- ▶ Um pacote fonte pode gerar vários pacotes binários  
ex, a fonte `libtar` gera pacotes binários `libtar0` e `libtar-dev`
- ▶ Dois tipos de pacotes: (em caso de dúvida, use não-nativo)
  - ▶ Pacotes nativos: normalmente para software específico de Debian (*dpkg*, *apt*)
  - ▶ Pacotes não-nativos: software desenvolvido fora de Debian
- ▶ Ficheiro principal: `.dsc` (meta-dados)
- ▶ Outros ficheiros que dependem da versão do formato fonte
  - ▶ 1.0 or 3.0 (nativo): `package_version.tar.gz`
  - ▶ 1.0 (não-nativo):
    - ▶ `pkg_ver.orig.tar.gz`: fonte da autoria (upstream)
    - ▶ `pkg_debver.diff.gz`: patch para adicionar alterações específicas de Debian
  - ▶ 3.0 (quilt):
    - ▶ `pkg_ver.orig.tar.gz`: fonte da autoria (upstream)
    - ▶ `pkg_debver.debian.tar.gz`: tarball com as alterações de Debian

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



## 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 Kothé <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 de Debian desde 2005)

## ► Do sistema de controlo de versão (declarado):

- `debcheckout pacote`

## ► Após a descarga, extraia com `dpkg-source -x file.dsc`



# Criar um pacote fonte básico

- ▶ Descarregue a fonte do autor (upstream)  
(*upstream source* = aquela dos desenvolvedores originais do software)
- ▶ Renomeie para `<source_package>_<upstream_version>.orig.tar.gz`  
(exemplo: `simgrid_3.6.orig.tar.gz`)
- ▶ Descompacte-o
- ▶ Renomeie o directório para `<source_package>-<upstream_version>`  
(exemplo: `simgrid-3.6`)
- ▶ `cd <source_package>-<upstream_version> && dh_make`  
(do pacote **dh-make**)
- ▶ Existem algumas alternativas ao `dh_make` para conjuntos específicos de pacotes: **dh-make-perl**, **dh-make-php**, ...
- ▶ Directório `debian/` criado, com muitos ficheiros lá dentro



# Ficheiros em debian/

Todo o trabalho de empacotamento deve ser feito ao modificar ficheiros em `debian/`

- ▶ Ficheiros principais:
  - ▶ **control** – meta-dados acerca do pacote (dependências, etc)
  - ▶ **rules** – especifica como compilar o pacote
  - ▶ **copyright** – informação de copyright para o pacote
  - ▶ **changelog** – história do pacote Debian
- ▶ Outros ficheiros:
  - ▶ `compat`
  - ▶ `watch`
  - ▶ `dh_install*` targets  
    `*.dirs`, `*.docs`, `*.manpages`, ...
  - ▶ scripts do mantenedor  
    `*.postinst`, `*.prerm`, ...
  - ▶ fonte/formato
  - ▶ patches/ – se você precisar de modificar as fontes do autor
- ▶ Vários ficheiros usam um formato baseado em RFC 822 (cabeçalhos principais)





# debian/changelog

- ▶ Lista as alterações de empacotamento Debian
- ▶ Dá a versão actual do pacote

1.2.1.1-5  
Upstream Debian  
version revision

- ▶ Editado manualmente ou com **dch**
  - ▶ Crie uma entrada no changelog para um novo lançamento: **dch -i**
- ▶ Formato especial para fechar automaticamente bugs de 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 desta fonte
  - ▶ Nome do pacote, secção, prioridade, mantenedor, quem faz os uploads, dependências de compilação, dependências, descrição, página do projecto, ...
  - ▶ Documentação: Política Debian capítulo 5  
<http://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 nenhuma

Dois tipos de pacotes binários:

- ▶ Pacotes com conteúdos diferentes para cada arquitetura 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 todas as outras arquiteturas para si ao submeter
  - ▶ 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_todas.deb`

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



# debian/rules

- ▶ Makefile
- ▶ Interface usada para compilar pacotes Debian
- ▶ Documentado em Política Debian, capítulo 4.8  
<http://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 irá chamar binary para compilar todos os pacotes, ou binary-arch para compilar apenas os pacotes de Arquitetura: any
  - ▶ clean: limpa o directório fonte



# Ajudantes de empacotamento – debhelper

- ▶ Você podia escrever código de shell directamente em `debian/rules`
  - ▶ Veja o pacote `adduser` como exemplo
- ▶ Melhor prática (usada pela maioria dos pacotes): use um *Ajudante de Empacotamento*
- ▶ O mais popular deles: **debhelper** (usado por 98% dos pacotes)
- ▶ Objectivos:
  - ▶ Factoriza as tarefas comuns em ferramentas standard usadas por todos os pacotes
  - ▶ Corrige alguns 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 de `debian/rules`
- ▶ Configurável usando parâmetros de comandos ou ficheiros em `debian/`

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



## 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/package.sgml > package.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/package 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 muita redundância entre pacotes
- ▶ Ajudantes de segundo-nível que factorizam funcionalidades comuns
  - ▶ Ex. compilando com `./configure && make && make install` ou CMake
- ▶ CDBS:
  - ▶ Introduzido em 2005, baseado na magia avançada do *GNU make*
  - ▶ Documentação: `/usr/share/doc/cdb/`
  - ▶ Suporte para Perl, Python, Ruby, GNOME, KDE, Java, Haskell, ...
  - ▶ Mas algumas pessoas detestam-o:
    - ▶ Por vezes é difícil personalizar compilações de pacotes:  
*"labirinto distorcido de makefiles e variáveis de ambiente"*
    - ▶ Mais lento que o debhelper simples (muitas chamadas desnecessárias a `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 (aka Debhelper 7, ou dh7)

- ▶ Introduzido em 2008 como o *matador do CDBS*
- ▶ **dh** comando 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 reunião 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
```

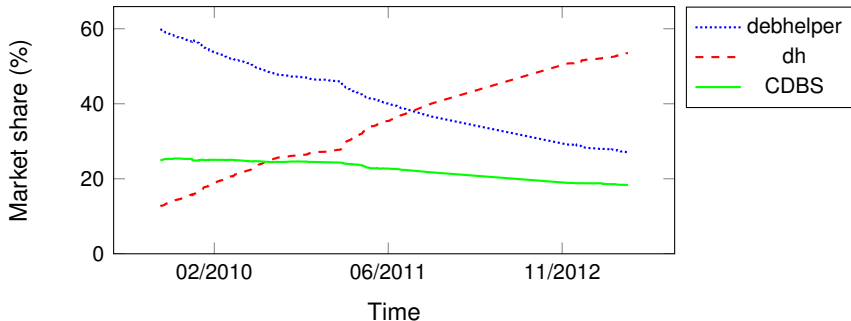
```
override_dh_auto_build:
```

```
make world
```



# debhelper clássico contra CDBS contra dh

- ▶ Mind shares:  
debhelper clássico: 27%   CDBS: 18%   dh: 54%
- ▶ Qual deles devo aprender?
  - ▶ Provavelmente um pouco de todos eles
  - ▶ Você precisa de conhecer o debhelper para usar o dh e o CDBS
  - ▶ Você poderá ter que modificar pacotes CDBS
- ▶ Qual deles devo usar para um pacote novo?
  - ▶ **dh** (solução apenas com um aumento da mind share)



# Esboço

- 1 Introdução
- 2 Criar pacotes fonte
- 3 Compilando e testando pacotes**
- 4 Sessão prática 1: modificar o pacote grep
- 5 Tópicos de empacotamento avançados
- 6 Mantendo pacotes em Debian
- 7 Conclusões
- 8 Sessão prática 2: empacotar o GNUjump
- 9 Sessão prática 3: empacotando uma biblioteca Java
- 10 Sessão prática 4: empacotar uma gema Ruby
- 11 Respostas às sessões práticas



# Compilando pacotes

- ▶ `apt-get build-dep mypackage`  
Instala as *build-dependencies* (para um pacote já em Debian)  
Ou `mk-build-deps -ir` (para um pacote ainda não submetido)
- ▶ `debuild`: compila, testa com `lintian`, assina com GPG
- ▶ Também possível chamar directamente `dpkg-buildpackage`
  - ▶ Normalmente com `dpkg-buildpackage -us -uc`
- ▶ É melhor compilar os pacotes num ambiente limpo & mínimo
  - ▶ `pbuilder` – ajudante para compilar pacotes em *chroot*  
Boa documentação: <https://wiki.ubuntu.com/PbuilderHowto>  
(optimização: `cowbuilder ccache distcc`)
  - ▶ `schroot` e `sbuid`: usados nos daemons de compilação de Debian  
(não tão simples como `pbuilder`, mas permite instantâneos LVM  
veja: <https://help.ubuntu.com/community/SbuildLVMHowto> )
- ▶ Gera ficheiros `.deb` e um ficheiro `.changes`
  - ▶ `.changes`: descreve o que foi compilado; usado para fazer o upload do pacote

# Instalando e testando pacotes

- ▶ Instale o pacote localmente: `debi` (irá usar `.changes` para saber o que instalar)
- ▶ Liste o conteúdo do pacote: `debc` `../mypackage<TAB>.changes`
- ▶ Compara o pacote com a versão anterior:  
`debdiff` `../mypackage_1_*.changes` `../mypackage_2_*.changes`  
ou para comparar as fontes:  
`debdiff` `../mypackage_1_*.dsc` `../mypackage_2_*.dsc`
- ▶ Verifique o pacote com `lintian` (analisador estático):  
`lintian` `../mypackage<TAB>.changes`  
`lintian -i`: dá mais informação acerca de erros  
`lintian -EviIL +pedantic`: mostra mais problemas
- ▶ Faça o upload do pacote para Debian (`dput`) (precisa de configuração)
- ▶ Faça gestão de um arquivo Debian privado com `reprepro`  
Documentação: <http://mirrorer.alioth.debian.org/>



# Esboço

- 1 Introdução
- 2 Criar pacotes fonte
- 3 Compilando e testando pacotes
- 4 Sessão prática 1: modificar o pacote grep
- 5 Tópicos de empacotamento avançados
- 6 Mantendo pacotes em Debian
- 7 Conclusões
- 8 Sessão prática 2: empacotar o GNUjump
- 9 Sessão prática 3: empacotando uma biblioteca Java
- 10 Sessão prática 4: empacotar uma gema Ruby
- 11 Respostas às sessões práticas



# Sessão prática 1: modificar o pacote grep

- ❶ Vá a <http://ftp.debian.org/debian/pool/main/g/grep/> e descarregue a versão 2.6.3-3 do pacote (se você usar Ubuntu 11.10 ou posterior, ou Debian testing ou unstable, então use a versão 2.9-1 ou 2.9-2)
  - ▶ Se o pacote fonte não descompactar automaticamente, descompacte-o com `dpkg-source -x grep_*.dsc`
- ❷ Observe os ficheiros em `debian/`.
  - ▶ Quantos pacotes binários são gerados por este pacote fonte?
  - ▶ Qual o ajudante de empacotamento este pacote usa?
- ❸ Compile o pacote
- ❹ Agora você vai modificar o pacote. Adicione uma entrada changelog e incremente o número da versão.
- ❺ Agora desactive o suporte a perl-regexp (é uma opção de `./configure`)
- ❻ Re-compile o pacote
- ❼ Compare os pacotes original e novo com o `debdiff`
- ❽ instale o pacote compilado recentemente
- ❾ Chore se fez asneira ;)



# Esboço

- 1 Introdução
- 2 Criar pacotes fonte
- 3 Compilando e testando pacotes
- 4 Sessão prática 1: modificar o pacote grep
- 5 Tópicos de empacotamento avançados
- 6 Mantendo pacotes em Debian
- 7 Conclusões
- 8 Sessão prática 2: empacotar o GNUjump
- 9 Sessão prática 3: empacotando uma biblioteca Java
- 10 Sessão prática 4: empacotar uma gema Ruby
- 11 Respostas às sessões práticas





# debian/copyright

- ▶ Informação de copyright e licença para a fonte e o empacotamento
- ▶ Tradicionalmente escrito num ficheiro de texto
- ▶ Novo formato máquina-legível:

<http://www.debian.org/doc/packaging-manuals/copyright-format/1.0/>

---

```
Format: http://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]
```



# Modificar a fonte do autor

Muitas vezes necessário:

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

Vários métodos para o fazer:

- ▶ Modificar os ficheiros directamente
  - ▶ Simples
  - ▶ Mas sem modo de acompanhar e documentar as alterações
- ▶ utilizando sistemas de patch
  - ▶ Facilita a contribuição das suas alterações para o autor original (upstream)
  - ▶ Ajuda a partilhar as correcções com os derivados
  - ▶ Dá melhor exibição às alterações  
<http://patch-tracker.debian.org/>



# 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* (*cdb*s), *dpatch*, **quilt**
  - ▶ Cada um suporta dois alvos `debian/rules`:
    - ▶ `debian/rules patch`: aplica todas as patches
    - ▶ `debian/rules unpatch`: retira as alterações de todas as patches
  - ▶ Mais documentação: <http://wiki.debian.org/debian/patches>
- ▶ **Novo formato de pacote fonte com sistema de patch integrado: 3.0 (quilt)**
  - ▶ Solução recomendada
  - ▶ Você precisa de aprender *quilt*  
<http://pkg-perl.alioth.debian.org/howto/quilt.html>
  - ▶ Ferramenta patch-system-agnostic em `devscripts`: `edit-patch`



# Documentação de patches

- ▶ Cabeçalhos standard no início da 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 {
```



# Fazer coisas durante a instalação e remoção

- ▶ Descomprimir o pacote por vezes não é suficiente
- ▶ Criar/remover utilizadores do sistema, iniciar/para serviços, gerir *alternativas*
- ▶ Feito nos *scripts do mantenedor*  
preinst, postinst, prerm, postrm
  - ▶ Podem ser gerados fragmentos para acções comuns pelo debhelper
- ▶ Documentação:
  - ▶ Manual de politicas Debian, capítulo 6  
<http://www.debian.org/doc/debian-policy/ch-maintainerscripts>
  - ▶ Referência dos Desenvolvedores de Debian, capítulo 6.4  
<http://www.debian.org/doc/developers-reference/best-pkging-practices.html>
  - ▶ <http://people.debian.org/~srivasta/MaintainerScripts.html>
- ▶ Questionando o utilizador
  - ▶ Tem de ser feito com **debconf**
  - ▶ Documentação: debconf-devel(7) (pacote debconf-doc)



# Monitorizando versões do autor (upstream)

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

```
version=3
```

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

- ▶ Infraestrutura Debian que faz uso de `debian/watch`:

## **Debian External Health Status**

<http://dehs.alioth.debian.org/>

- ▶ Mantenedor do pacote avisado por emails enviados para o Sistema de Acompanhamento de Pacotes

<http://packages.qa.debian.org/>

- ▶ `uscan`: corre uma verificação manual
- ▶ `uupdate`: tenta actualizar o seu pacote para a versão do autor mais recente



# Empacotar com um Sistema de Controlo de Versão

- ▶ Várias ferramentas para ajudar a gerir ramos e etiquetas para o seu trabalho de empacotamento:  
svn-buildpackage, git-buildpackage
- ▶ Exemplo: git-buildpackage
  - ▶ upstream ramo para acompanhar a autoria com as etiquetas  
upstream/*version*
  - ▶ master ramo que acompanha o pacote Debian
  - ▶ debian/*version* etiquetas para cada envio (upload)
  - ▶ pristine-tar ramo para ser possível recompilar o tarball do autor
- ▶ Vcs-\* campos em debian/control para localizar o repositório
  - ▶ <http://wiki.debian.org/Alioth/Git>
  - ▶ <http://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](http://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](http://svn.debian.org/viewsvn/pkg-perl/trunk/libwww-perl/)

- ▶ Interface VCS-agnostic: debcheckout, debcommit, debrelease



# Portar pacotes para trás (backporting)

- ▶ Objectivo: usar uma nova versão de um pacote num sistema mais antigo  
ex. usar *mutt* de Debian *unstable* em Debian *stable*
- ▶ Ideia geral:
  - ▶ Obtenha o pacote fonte de Debian *unstable*
  - ▶ Modifique para que compile e funcione bem em Debian *stable*
    - ▶ Às vezes é trivial (sem alterações necessárias)
    - ▶ Às vezes é difícil
    - ▶ Às vezes é impossível (muitas dependências não disponíveis)
- ▶ Alguns "backports" são disponibilizados e suportados pelo projecto Debian  
<http://backports.debian.org/>





# Esboço

- 1 Introdução
- 2 Criar pacotes fonte
- 3 Compilando e testando pacotes
- 4 Sessão prática 1: modificar o pacote grep
- 5 Tópicos de empacotamento avançados
- 6 Mantendo pacotes em Debian**
- 7 Conclusões
- 8 Sessão prática 2: empacotar o GNUjump
- 9 Sessão prática 3: empacotando uma biblioteca Java
- 10 Sessão prática 4: empacotar uma gema Ruby
- 11 Respostas às sessões práticas



# Várias maneiras de contribuir para Debian

---

▶ **Pior** maneira de contribuir:

- 1 Empacote a sua própria aplicação
- 2 Entre para a Debian
- 3 Desapareça

▶ **Melhores** maneiras de contribuir:

- ▶ Envolver-se com as equipas de empacotamento
  - ▶ Muitas equipas que se focam em conjuntos de pacotes, e precisam de ajuda
  - ▶ Lista disponível em <http://wiki.debian.org/Teams>
  - ▶ uma excelente maneira de aprender a partir de contribuintes mais experientes
- ▶ Adotar pacotes não mantidos existentes (*pacotes órfãos*)
- ▶ Traga novo software para Debian
  - ▶ Apenas se for suficientemente interessante/útil, por favor
  - ▶ Existem alternativas já empacotadas em Debian?



# Adoptando pacotes órfãos

- ▶ Muitos pacotes não mantidos em Debian
- ▶ Lista completa + processo: <http://www.debian.org/devel/wnpp/>
- ▶ Instalado na sua máquina: `wnpp-alert`
- ▶ Estados diferentes:
  - ▶ **Orphaned**: o pacote não é mantido  
Sinta-se livre para o adoptar
  - ▶ **RFA: Request For Adopter**  
O mantenedor procura quem adopte, mas entretanto continua a trabalhar  
Sinta-se livre para adoptar. É cortês enviar um mail ao actual mantenedor
  - ▶ **ITA: Intent To Adopt**  
Alguém tenciona adoptar o pacote  
Você pode propor-se a ajudar!
  - ▶ **RFH: Request For Help**  
O mantenedor procura ajuda
- ▶ Alguns pacotes não mantidos e não detectados → 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 anterior mantenedor (especialmente se o pacote estava em RFA, não órfão)
- ▶ É uma boa ideia contactar a autoria do projecto



# Colocando o seu pacote na Debian

- ▶ Você não precisa de nenhum estado oficial para ter o seu pacote na Debian
  - ➊ Submeter um **ITP** bug (**I**ntend **T**o **P**ackage) usando `reportbug wnpp`
  - ➋ Preparar um pacote fonte
  - ➌ Encontre um Desenvolvedor Debian que patrocine o seu pacote
- ▶ Estado oficial (quando você é um mantenedor de pacotes experiente)
  - ▶ **Mantenedor Debian (DM):**  
Permissão para submeter os seus próprios pacotes  
Veja <http://wiki.debian.org/DebianMaintainer>
  - ▶ **Desenvolvedor Debian (DD):**  
Membro do projecto Debian; pode votar e submeter (upload) qualquer pacote



# Coisas a verificar antes de pedir patrocínio

---

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



# Onde encontrar ajuda?

Ajuda que irá precisar:

- ▶ Conselhos e respostas para as suas questões, revisões de código
- ▶ Patrocinador para os seus envios (uploads), assim que o seu pacote esteja pronto

Você pode obter ajuda de:

- ▶ **Outros membros de uma equipa de empacotamento**
  - ▶ Lista de equipas: <http://wiki.debian.org/Teams>
- ▶ **O Debian Mentors group** (se o seu pacote não encaixar numa equipa)
  - ▶ <http://wiki.debian.org/DebianMentorsFaq>
  - ▶ Lista de mail: [debian-mentors@lists.debian.org](mailto:debian-mentors@lists.debian.org)  
(também uma boa maneira de aprender por acaso)
  - ▶ IRC: #debian-mentors em [irc.debian.org](http://irc.debian.org)
  - ▶ <http://mentors.debian.net/>
  - ▶ Documentação: <http://mentors.debian.net/intro-maintainers>
- ▶ **Listas de mail localizadas** (obtenha ajuda na sua linguagem)
  - ▶ [debian-devel-{french,italian,portuguese,spanish}@lists.d.o](mailto:debian-devel-{french,italian,portuguese,spanish}@lists.d.o)
  - ▶ Lista completa: <https://lists.debian.org/devel.html>
  - ▶ Ou listas de utilizadores: <https://lists.debian.org/users.html>



# Mais documentação

- ▶ O Cantinho dos Desenvolvedores de Debian  
<http://www.debian.org/devel/>  
Links para muitos recursos acerca do desenvolvimento de Debian
- ▶ Guia dos Novos Mantenedores de Debian  
<http://www.debian.org/doc/maint-guide/>  
Uma introdução ao empacotamento de Debian, mas que precisa de uma actualização
- ▶ Referência dos Desenvolvedores de Debian  
<http://www.debian.org/doc/developers-reference/>  
Maioritariamente acerca dos procedimentos de Debian, mas também algumas das melhores práticas de empacotamento (parte 6)
- ▶ Política de Debian  
<http://www.debian.org/doc/debian-policy/>
  - ▶ Todos os requerimentos que cada pacote deve satisfazer
  - ▶ Políticas específicas para Perl, Java, Python, ...
- ▶ Guia de Empacotamento de Ubuntu  
<http://developer.ubuntu.com/resources/tools/packaging/>





# Bancadas Debian para mantenedores

- ▶ **Centrado no pacote fonte:** Sistema de Acompanhamento de Pacotes (PTS)  
<http://packages.qa.debian.org/dpkg>
- ▶ **Centrado em mantenedor/equipa:** Visão Geral de Pacotes de Desenvolvedores (DDPO)  
<http://qa.debian.org/developer.php?login=pkg-ruby-extras-maintainers@lists.alioth.debian.org>
- ▶ **Lista-A-FAZER orientada:** Bancada do Mantenedor Debian (DMD)  
<http://udd.debian.org/dmd.cgi>



# Usando o Debian Bug Tracking System (BTS)

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



# Usando o BTS: exemplos

- ▶ Enviar um email para o bug e para quem o submeteu:  
`http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680822#10`
- ▶ Etiquetar e alterar a severidade:  
`http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680227#10`
- ▶ Re-atribuir, alterar a severidade, mudar o título ...:  
`http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680822#93`
  - ▶ `notfound`, `found`, `notfixed`, `fixed` são para **version-tracking**  
Veja `https://wiki.debian.org/HowtoUseBTS#Version\_tracking`
- ▶ Usando usertags:  
`http://bugs.debian.org/cgi-bin/bugreport.cgi?msg=42;bug=642267`  
Veja `https://wiki.debian.org/bugs.debian.org/usertags`
- ▶ Documentação de BTS:
  - ▶ `http://www.debian.org/Bugs/`
  - ▶ `https://wiki.debian.org/HowtoUseBTS`



## Mais interessado em Ubuntu?

- ▶ Ubuntu maioritariamente gere a divergência com Debian
- ▶ Nenhuma focagem real em pacotes específicos  
Em vez disso, colaboração com as equipas de Debian
- ▶ Normalmente é recomendado enviar primeiro os novos pacote para Debian  
<https://wiki.ubuntu.com/UbuntuDevelopment/NewPackages>
- ▶ Possivelmente um plano melhor:
  - ▶ Envolve-se numa equipa de Debian e actue como uma ponte com Ubuntu
  - ▶ Ajuda reduz divergência, triagem de bugs no Launchpad
  - ▶ Muitas ferramentas de Debian podem ajudar:
    - ▶ Coluna do Ubuntu na visão geral de pacotes de Desenvolvedores
    - ▶ Caixa do Ubuntu no Sistema de Acompanhamento de Pacotes
    - ▶ Recebe bugmail do launchpad via PTS



# Esboço

- 1 Introdução
- 2 Criar pacotes fonte
- 3 Compilando e testando pacotes
- 4 Sessão prática 1: modificar o pacote grep
- 5 Tópicos de empacotamento avançados
- 6 Mantendo pacotes em Debian
- 7 Conclusões
- 8 Sessão prática 2: empacotar o GNUjump
- 9 Sessão prática 3: empacotando uma biblioteca Java
- 10 Sessão prática 4: empacotar uma gema Ruby
- 11 Respostas às sessões práticas



# Conclusões

- ▶ Agora você tem uma visão geral completa do empacotamento de Debian
- ▶ Mas você irá precisar de ler mais documentação
- ▶ As melhores práticas evoluíram com os anos
  - ▶ Em dúvida, use o ajudante de empacotamento **dh**, e o formato **3.0 (quilt)**
- ▶ Coisas que não foram cobertas por este manual:
  - ▶ UCF – gere as alterações do utilizador nos ficheiros de configuração quando actualiza
  - ▶ dpkg triggers – agrupa e junta acções semelhantes de scripts de mantenedor
  - ▶ Organização de desenvolvimento de Debian:
    - ▶ Suites: stable, testing, unstable, experimental, security, \*-updates, backports, ...
    - ▶ Debian Blends – subconjuntos de Debian que apontam a grupos específicos

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



# Matérias legais

Copyright ©2011–2013 Lucas Nussbaum – [lucas@debian.org](mailto: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/>



# 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`

## ► Forneça comentários de retorno (Feedback):

- `mailto:packaging-tutorial@packages.debian.org`
  - o que deve ser adicionado a este manual?
  - O que deve ser melhorado?
- `reportbug packaging-tutorial`





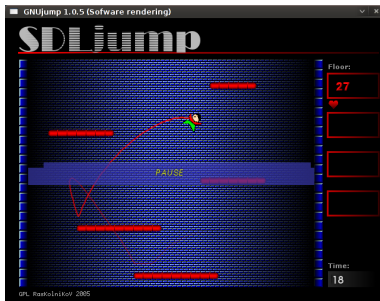
# Esboço

- 1 Introdução
- 2 Criar pacotes fonte
- 3 Compilando e testando pacotes
- 4 Sessão prática 1: modificar o pacote grep
- 5 Tópicos de empacotamento avançados
- 6 Mantendo pacotes em Debian
- 7 Conclusões
- 8 Sessão prática 2: empacotar o GNUjump
- 9 Sessão prática 3: empacotando uma biblioteca Java
- 10 Sessão prática 4: empacotar uma gema Ruby
- 11 Respostas às sessões práticas



# Sessão prática 2: empacotar 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 que possa compilar o pacote
  - ▶ Obtenha um pacote funcional básico
  - ▶ Acabe de preencher `debian/control` e outros ficheiros
- 3 Aprecie



# Esboço

- 1 Introdução
- 2 Criar pacotes fonte
- 3 Compilando e testando pacotes
- 4 Sessão prática 1: modificar o pacote grep
- 5 Tópicos de empacotamento avançados
- 6 Mantendo pacotes em Debian
- 7 Conclusões
- 8 Sessão prática 2: empacotar o GNUjump
- 9 Sessão prática 3: empacotando uma biblioteca Java**
- 10 Sessão prática 4: empacotar uma gema Ruby
- 11 Respostas às sessões práticas



## Sessão prática 3: empacotando uma biblioteca Java

- 1 Faça uma leitura rápida a alguma documentação sobre empacotamento de Java:

- ▶ <http://wiki.debian.org/Java>
- ▶ <http://wiki.debian.org/Java/Packaging>
- ▶ <http://www.debian.org/doc/packaging-manuals/java-policy/>
- ▶ <http://pkg-java.alioth.debian.org/docs/tutorial.html>
- ▶ Papel e slides de uma reunião Debconf10 acerca de javahelper:  
<http://pkg-java.alioth.debian.org/docs/debconf10-javahelper-paper.pdf>  
<http://pkg-java.alioth.debian.org/docs/debconf10-javahelper-slides.pdf>

- 2 Descarregue o IRClib de <http://moepii.sourceforge.net/>

- 3 Empacote-o



# Esboço

- 1 Introdução
- 2 Criar pacotes fonte
- 3 Compilando e testando pacotes
- 4 Sessão prática 1: modificar o pacote grep
- 5 Tópicos de empacotamento avançados
- 6 Mantendo pacotes em Debian
- 7 Conclusões
- 8 Sessão prática 2: empacotar o GNUjump
- 9 Sessão prática 3: empacotando uma biblioteca Java
- 10 Sessão prática 4: empacotar uma gema Ruby
- 11 Respostas às sessões práticas



## Sessão prática 4: empacotar uma gema Ruby

---

- ❶ Dê uma leitura rápida a alguma documentação acerca de empacotamento de Ruby:
  - ▶ <http://wiki.debian.org/Ruby>
  - ▶ <http://wiki.debian.org/Teams/Ruby>
  - ▶ <http://wiki.debian.org/Teams/Ruby/Packaging>
  - ▶ `gem2deb(1)`, `dh_ruby(1)` (no pacote `gem2deb`)
- ❷ Crie um pacote fonte Debian básico a partir da gema `net-ssh`:  
`gem2deb net-ssh`
- ❸ Melhore-o para que se torne num pacote Debian apropriado



# Esboço

- 1 Introdução
- 2 Criar pacotes fonte
- 3 Compilando e testando pacotes
- 4 Sessão prática 1: modificar o pacote grep
- 5 Tópicos de empacotamento avançados
- 6 Mantendo pacotes em Debian
- 7 Conclusões
- 8 Sessão prática 2: empacotar o GNUjump
- 9 Sessão prática 3: empacotando uma biblioteca Java
- 10 Sessão prática 4: empacotar uma gema Ruby
- 11 Respostas às sessões práticas



# Respostas para sessões práticas





# Sessão prática 1: modificar o pacote grep

- 1 Vá a <http://ftp.debian.org/debian/pool/main/g/grep/> e descarregue a versão 2.6.3-3 do pacote (se você usar Ubuntu 11.10 ou posterior, ou Debian testing ou unstable, então use a versão 2.9-1 ou 2.9-2)
- 2 Observe os ficheiros em `debian/`.
  - ▶ Quantos pacotes binários são gerados por este pacote fonte?
  - ▶ Qual o ajudante de empacotamento este pacote usa?
- 3 Compile o pacote
- 4 Agora você vai modificar o pacote. Adicione uma entrada changelog e incremente o número da versão.
- 5 Agora desactive o suporte a perl-regexp (é uma opção de `./configure`)
- 6 Re-compile o pacote
- 7 Compare os pacotes original e novo com o `debdiff`
- 8 instale o pacote compilado recentemente
- 9 Chore se fez asneira ;)



## Obtendo a fonte

- ❶ Vá a `http://ftp.debian.org/debian/pool/main/g/grep/` e descarregue a versão 2.6.3-3 do pacote
- ▶ Use o `dget` para descarregar o ficheiro `.dsc`:  
`dget http://cdn.debian.net/debian/pool/main/g/grep/grep_2.6.3-3.dsc`
- ▶ De acordo com `http://packages.qa.debian.org/grep`, `grep` a versão 2.6.3-3 está actualmente em *stable* (*squeeze*). Se você tem linhas `deb-src` para *squeeze* no seu `/etc/apt/sources.list`, pode usar:  
`apt-get source grep=2.6.3-3`  
ou `apt-get source grep/stable`  
ou, se estiver com sorte: `apt-get source grep`
- ▶ O pacote fonte do `grep` é composto por três ficheiros:
  - ▶ `grep_2.6.3-3.dsc`
  - ▶ `grep_2.6.3-3.debian.tar.bz2`
  - ▶ `grep_2.6.3.orig.tar.bz2`Isto é típico do formato "3.0 (quilt)".
- ▶ Se necessário, descomprima a fonte com  
`dpkg-source -x grep_2.6.3-3.dsc`



# Observando e compilando o pacote

## 2 Observe os ficheiros em `debian/`.

- ▶ Quantos pacotes binários são gerados por este pacote fonte?
- ▶ Qual o ajudante de empacotamento este pacote usa?
- ▶ De acordo com `debian/control`, este pacote apenas gera um pacote binário, chamado `grep`.
- ▶ De acordo com `debian/rules`, este pacote é típico de empacotamento *classic* debhelper, 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 registo de alterações (changelog)

- ④ Agora você vai modificar o pacote. Adicione uma entrada changelog e incremente o número da versão.
- ▶ `debian/changelog` é um ficheiro 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`.
- ▶ Após isso, recompile o pacote: é compilada uma nova versão do pacote.
- ▶ O "versionamento" do pacote está detalhado na secção 5.6.12 da política Debian.  
<http://www.debian.org/doc/debian-policy/ch-controlfields>



# Desactivando suporte regexp de Perl e recompilando

- 5 Agora desactive o suporte a perl-regexp (é uma opção de `./configure`)
- 6 Re-compile o pacote
  - ▶ Verifique com `./configure --help`: a opção para desactivar Perl regexp é `--disable-perl-regexp`
  - ▶ Edite `debian/rules` e encontre a linha `./configure`
  - ▶ Adicione `--disable-perl-regexp`
  - ▶ Recompile com `debuild` ou `dpkg-buildpackage -us -uc`



# Comparar e testar os pacotes

- 7 Compare os pacotes original e novo com o debdiff
- 8 instale o pacote compilado recentemente
  - ▶ 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!
- 9 Chore se fez asneira ;)

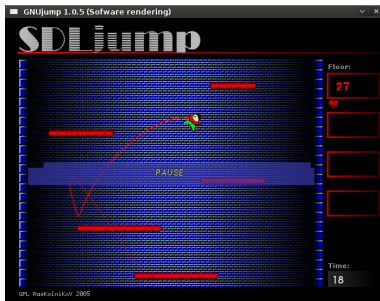
Ou não: reinstale a versão anterior do pacote:

- ▶ `apt-get install --reinstall grep=2.6.3-3 (= previous version)`



# Sessão prática 2: empacotar 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 que possa compilar o pacote
  - ▶ Obtenha um pacote funcional básico
  - ▶ Acabe de preencher `debian/control` e outros ficheiros
- 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`
  - ▶ 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
emacsens-install.ex manpage.xml.ex      source
emacsens-remove.ex menu.ex             watch.ex
emacsens-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**)
- ▶ In `debian/control`:  
Build-Depends: `debhelper (>= 7.0.50 )`, `autotools-dev`  
Lista as *build-dependencies* = pacotes necessários para compilar o pacote
- ▶ Tenta compilar o pacote como está (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** às Build-Depends e instale-o.

- ▶ Melhor: use **pbuilder** para compilar num ambiente limpo



## Passo a passo... (3)

- ▶ Após instalar `libSDL1.2-dev`, `libSDL-image1.2-dev`, `libSDL-mixer1.2-dev`, o pacote compila bem.
- ▶ Use `debnc` para listar o conteúdo do pacote gerado.
- ▶ Use `debi` para o instalar e testar.
- ▶ Teste o pacote com `lintian`
  - ▶ Embora não seja um requerimento estrito, é recomendado que os pacotes enviados para Debian sejam *lintian-clean* (passaram o teste do `lintian`)
  - ▶ Mais problemas podem ser listados usando `lintian -EviIL +pedantic`
  - ▶ Algumas dicas:
    - ▶ Remova os ficheiros que você não precisa em `debian/`
    - ▶ Preencha `debian/control`
    - ▶ Instale o executável para `/usr/games` ao sobrepor `dh_auto_configure`
    - ▶ Use flags de compilador *hardening* para aumentar a segurança. Veja <http://wiki.debian.org/Hardening>

## Passo a passo... (4)

- ▶ Compare o seu pacote com aquele já empacotado em Debian:
  - ▶ Divide os ficheiros de dados para um segundo pacote, que é o mesmo para todas as arquitecturas (→ poupa espaço no arquivo de Debian)
  - ▶ Instala um ficheiro .desktop (para os menus de GNOME/KDE) e também o integra no menu Debian
  - ▶ Corrige alguns problemas menores usando patches



## Sessão prática 3: empacotando uma biblioteca Java

- ❶ Faça uma leitura rápida a alguma documentação sobre empacotamento de Java:
  - ▶ <http://wiki.debian.org/Java>
  - ▶ <http://wiki.debian.org/Java/Packaging>
  - ▶ <http://www.debian.org/doc/packaging-manuals/java-policy/>
  - ▶ <http://pkg-java.alioth.debian.org/docs/tutorial.html>
  - ▶ Papel e slides de uma reunião Debconf10 acerca de javahelper:  
<http://pkg-java.alioth.debian.org/docs/debconf10-javahelper-paper.pdf>  
<http://pkg-java.alioth.debian.org/docs/debconf10-javahelper-slides.pdf>
- ❷ Descarregue o IRCLib de <http://moepii.sourceforge.net/>
- ❸ Empacote-o



## Passo a passo...

- ▶ `apt-get install javahelper`
- ▶ Crie um pacote fonte básico: `jh_makepkg`
  - ▶ Biblioteca
  - ▶ Nenhum
  - ▶ Compilador/executor em tempo real Livre Predefinido
- ▶ 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: empacotar uma gema Ruby

---

- ❶ Dê uma leitura rápida a alguma documentação acerca de empacotamento de Ruby:
  - ▶ `http://wiki.debian.org/Ruby`
  - ▶ `http://wiki.debian.org/Teams/Ruby`
  - ▶ `http://wiki.debian.org/Teams/Ruby/Packaging`
  - ▶ `gem2deb(1)`, `dh_ruby(1)` (no pacote `gem2deb`)
- ❷ Crie um pacote fonte Debian básico a partir da gema `net-ssh`:  
`gem2deb net-ssh`
- ❸ Melhore-o para que se torne num pacote Debian apropriado



## Passo a passo...

`gem2deb net-ssh:`

- ▶ Descarrega a gema de `rubygems.org`
- ▶ Cria um arquivo `.orig.tar.gz` apropriado e descompacta-o
- ▶ Inicializa um pacote fonte Debian baseado nos meta-dados da gema
  - ▶ Chamado `ruby-gemname`
- ▶ Tenta compilar o pacote binário Debian (isto pode falhar)

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

- ▶ Compila extensões de C para cada versão de Ruby
- ▶ Copie os ficheiros para o sue directório de destino
- ▶ Actualiza shebangs nos scripts executáveis
- ▶ Corra os testes definido em `debian/ruby-tests.rb` ou `debian/ruby-test-files.yaml`, assim como várias outras verificações



## Passo a passo... (2)

Melhore o pacote gerado

- ▶ Corra `debclean` para limpar a árvore fonte. Veja em `debian/`.
- ▶ `changelog` e `compat` devem estar correctos
- ▶ Edite `debian/control`: descomente Homepage, melhore Description
- ▶ Escreva um ficheiro `copyright` apropriado com base nos ficheiros do autor
- ▶ `ruby-net-ssh.docs`: instale `README.rdoc`
- ▶ `ruby-tests.rb`: corre os testes. Neste caso é suficiente fazer:  

```
$: << 'test' << 'lib' << '.'  
require 'test/test_all.rb'
```





## Passo a passo... (3)

Compile o pacote. Ele falha a compilar. Existem dois problemas:

- ▶ Você precisa de desactiva a chamada *gem* na suite de teste.  
Em `test/common.rb`, remova a linha `gem "test-unit"`:
  - ▶ `edit-patch disable-gem.patch`
  - ▶ Edite `test/common.rb`, remova a linha `gem`. Saia da sub-shell
  - ▶ Descreva as alterações em `debian/changelog`
  - ▶ Documente a patch em `debian/patches/disable-gem.patch`
- ▶ O pacote tem falta de dependência de compilação em *ruby-mocha*, que é usado pela suite de teste (talvez tenha que compilar o seu pacote num ambiente limpo, usando `pbuilder`, para reproduzir este problema)
  - ▶ Adicione *ruby-mocha* às Build-Depends do pacote
  - ▶ *gem2deb* copia as dependências documentadas em *gem* como comentários em `debian/control`, mas *mocha* não está listado como uma dependência de desenvolvimento pela gema (é um bug na gema)

Compare o seu pacote com o pacote *ruby-net-ssh* no arquivo Debian



# Tradução

Américo Monteiro

Se encontrar algum erro na tradução deste documento, por favor comunique para <a\_monteiro@gmx.com>. OU <traduz@debianpt.org>.

