

Tutoriel : la construction de paquets Debian

Lucas Nussbaum

`packaging-tutorial@packages.debian.org`

Traduction française de
Cédric Boutillier, Jean-Philippe Mengual
et l'équipe francophone de traduction

version 0.30 – 2024-03-16



À propos de ce tutoriel

- ▶ Objectif : **présenter ce que vous devez absolument savoir sur la construction de paquets Debian**
 - ▶ Modifier des paquets existants
 - ▶ Créer vos propres paquets
 - ▶ Interagir avec la communauté Debian
 - ▶ Devenir un utilisateur chevronné de Debian
- ▶ Il couvre les points les plus importants mais n'est pas complet
 - ▶ Vous devrez lire plus de documentation
- ▶ L'essentiel s'applique aussi aux distributions dérivées de Debian
 - ▶ en particulier à Ubuntu



Plan

- 1 Introduction
- 2 Création des paquets source
- 3 Construire et tester les paquets
- 4 Travaux pratiques n° 1 : modifier le paquet grep
- 5 Sujets avancés sur la construction de paquets
- 6 Maintenir des paquets dans Debian
- 7 Conclusions
- 8 Travaux pratiques supplémentaires
- 9 Solutions aux travaux pratiques



Plan

- 1 Introduction
- 2 Création des paquets source
- 3 Construire et tester les paquets
- 4 Travaux pratiques n° 1 : modifier le paquet grep
- 5 Sujets avancés sur la construction de paquets
- 6 Maintenir des paquets dans Debian
- 7 Conclusions
- 8 Travaux pratiques supplémentaires
- 9 Solutions aux travaux pratiques



Debian

- ▶ **Distribution GNU/Linux**
- ▶ 1^{re} distribution majeure développée « ouvertement dans l'esprit GNU »
- ▶ **Non commerciale**, fruit de la collaboration de plus de 1 000 bénévoles
- ▶ 3 caractéristiques principales :
 - ▶ **Qualité** – culture de l'excellence technique
Nous publions quand c'est prêt
 - ▶ **Liberté** – développeurs et utilisateurs adhèrent au *Contrat social*
Promotion de la culture du logiciel libre depuis 1993
 - ▶ **Indépendance** – pas d'entreprise (unique) pour chapeauter Debian
et processus décisionnel ouvert (*volontariat + démocratie*)
- ▶ **Amateur** dans le bon sens du terme : « fait avec amour »



Paquets Debian

- ▶ Fichiers **.deb** (paquets binaires)
- ▶ Moyen puissant et pratique pour distribuer des logiciels aux utilisateurs
- ▶ Un des deux formats de paquets les plus courants avec RPM
- ▶ Universel :
 - ▶ 30 000 paquets binaires dans Debian
→ la plupart des logiciels libres sont empaquetés dans Debian !
 - ▶ 12 portages (architectures), dont 2 non Linux (Hurd et kFreeBSD)
 - ▶ Utilisé aussi par 120 distributions dérivées de Debian



Le format de paquet Deb

- ▶ Fichier `.deb` : une archive `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` : version du format de fichier `.deb`, « 2.0\n »
 - ▶ `control.tar.gz` : métadonnées sur le paquet
control, md5sums, (pre|post)(rm|inst), triggers, shlibs...
 - ▶ `data.tar.gz` : fichiers de données du paquet
- ▶ Vous pourriez créer vos fichiers `.deb` à la main
http://tldp.org/HOWTO/html_single/Debian-Binary-Package-Building-HOWTO/
 - ▶ Mais la plupart des gens ne font pas comme ça

Ce tutoriel : création de paquets Debian à la manière Debian



Outils dont vous avez besoin

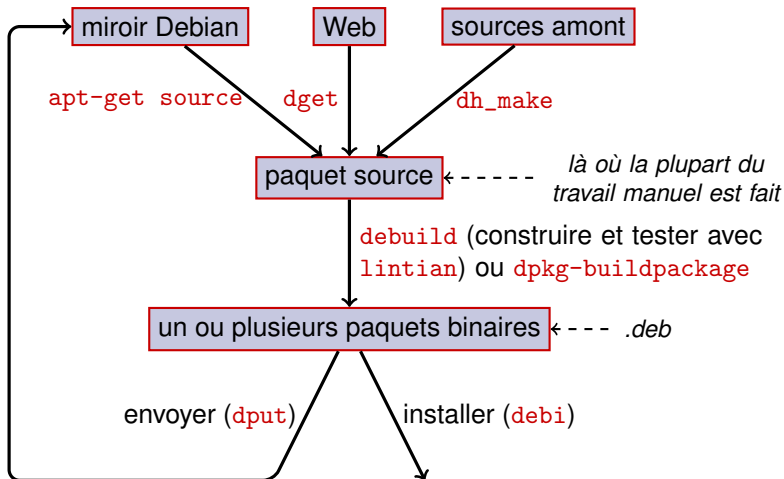
- ▶ Un système Debian (ou Ubuntu) (avec accès superutilisateur)
- ▶ Quelques paquets :
 - ▶ **build-essential** : dépend de paquets supposés disponibles sur la machine du développeur (inutile de les indiquer dans le champ de contrôle `Build-Depends` de votre paquet)
 - ▶ dépend aussi de **dpkg-dev**, contenant les outils de base spécifiques à Debian pour créer des paquets
 - ▶ **devscripts** : contient de nombreux scripts utiles pour les responsables Debian

Beaucoup d'autres outils seront aussi mentionnés plus tard, tels que **debhelper**, **cdb**, **quilt**, **pbuilder**, **sbuild**, **lintian**, **svn-buildpackage**, **git-buildpackage**...

Installez-les au besoin.



Processus général de la construction de paquets



Exemple : reconstruction de dash

- 1 Installez les paquets nécessaires à la construction de dash, ainsi que devscripts

```
apt-get build-dep dash
```

(nécessite des lignes deb-src dans /etc/apt/sources.list)

```
apt-get install --no-install-recommends devscripts fakeroot
```

- 2 Créez un répertoire de travail et entrez-y

```
mkdir /tmp/debian-tutorial ; cd /tmp/debian-tutorial
```

- 3 Récupérez le paquet source de dash

```
apt-get source dash
```

(Il faut pour cela avoir des lignes deb-src dans votre /etc/apt/sources.list)

- 4 Construisez le paquet

```
cd dash-*
```

```
debuild -us -uc (-us -uc désactive la signature du paquet avec GPG)
```

- 5 Vérifiez le résultat

▶ Il y a de nouveaux fichiers .deb dans le répertoire parent

- 6 Regardez le répertoire debian/

▶ C'est là que se fait le travail de construction du paquet



Plan

- 1 Introduction
- 2 Création des paquets source
- 3 Construire et tester les paquets
- 4 Travaux pratiques n° 1 : modifier le paquet grep
- 5 Sujets avancés sur la construction de paquets
- 6 Maintenir des paquets dans Debian
- 7 Conclusions
- 8 Travaux pratiques supplémentaires
- 9 Solutions aux travaux pratiques



Paquet source

- ▶ Un paquet source peut produire plusieurs paquets binaires
Le paquet source `libtar` produit les paquets binaires `libtar0` et `libtar-dev`
- ▶ Deux types de paquets : (en cas de doute, utilisez « non natif »)
 - ▶ natif : normalement pour les logiciels spécifiques à Debian (*dpkg*, *apt*...)
 - ▶ non natif : logiciels développés hors de Debian
- ▶ Fichier principal : `.dsc` (métadonnées)
- ▶ Autres fichiers selon la version du format source
 - ▶ 1.0 ou 3.0 (natif) : `paquet_version.tar.gz`
 - ▶ 1.0 (non natif) :
 - ▶ `paquet_ver.orig.tar.gz` : sources amont
 - ▶ `paquet_debver.diff.gz` : correctif avec des modifications spécifiques à Debian
 - ▶ 3.0 (quilt) :
 - ▶ `paquet_ver.orig.tar.gz` : sources amont
 - ▶ `paquet_debver.debian.tar.gz` : archive tar avec les modifications de Debian

(Consultez `dpkg-source(1)` pour les détails exacts.)



Exemple de paquet source (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
```

Récupération d'un paquet source existant

- ▶ À partir de l'archive Debian :
 - ▶ `apt-get source paquet`
 - ▶ `apt-get source paquet=version`
 - ▶ `apt-get source paquet/distribution`(Vous avez besoin de lignes `deb-src` dans `sources.list`)
- ▶ Depuis Internet :
 - ▶ `dget url-vers.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` fournit tous les paquets de Debian depuis 2005)
- ▶ Depuis le gestionnaire de versions (déclaré) :
 - ▶ `debcheckout paquet`
- ▶ Une fois téléchargé, dépaquetez-le avec `dpkg-source -x fichier.dsc`



Création d'un paquet source de base

- ▶ Téléchargez l'archive des sources amont
(*sources amont* = celles fournies par les développeurs du logiciel)
- ▶ Renommez-la en `<paquet_source>_<version_amont>.orig.tar.gz`
(exemple : `simgrid_3.6.orig.tar.gz`)
- ▶ Décompressez-la
- ▶ Renommez le répertoire en `<paquet_source>-<version_amont>`
(exemple : `simgrid-3.6`)
- ▶ `cd <paquet_source>-<version_amont> && dh_make` (du paquet **dh-make**)
- ▶ Il existe des alternatives à `dh_make` pour des types de paquets spécifiques : **dh-make-perl**, **dh-make-php**...
- ▶ Un répertoire `debian/` est créé, contenant de nombreux fichiers



Fichiers dans debian/

L'empaquetage ne doit se faire qu'en modifiant les fichiers de `debian/`

- ▶ Fichiers principaux :
 - ▶ **control** – métadonnées sur le paquet (dépendances, etc.)
 - ▶ **rules** – indique la manière de construire le paquet
 - ▶ **copyright** – informations de copyright du paquet
 - ▶ **changelog** – journal des modifications du paquet Debian
- ▶ Autres fichiers :
 - ▶ `compat`
 - ▶ `watch`
 - ▶ configuration de `dh_install*` (`*.dirs`, `*.docs`, `*.manpages...`)
 - ▶ scripts du responsable (`*.postinst`, `*.prerm...`)
 - ▶ `source/format`
 - ▶ `patches/` – si vous avez besoin de modifier les sources amont
- ▶ Plusieurs fichiers ont un format basé sur la RFC 822 (en-têtes de courriel)



debian/changelog

- ▶ Liste les modifications dans la construction du paquet Debian
- ▶ Donne la version actuelle du paquet

1.2.1.1-5

Version Révision
amont Debian

- ▶ Édité à la main ou avec `dch`
 - ▶ Pour créer une entrée pour une nouvelle version : `dch -i`
- ▶ Format spécial pour clôturer des bogues Debian ou Ubuntu
Debian : Closes: #595268; Ubuntu : LP: #616929
- ▶ Installé en tant que `/usr/share/doc/paquet/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

- ▶ Métadonnées du paquet
 - ▶ pour le paquet source lui-même
 - ▶ pour chaque paquet binaire construit à partir de ce paquet source
- ▶ Nom du paquet, section, priorité, responsable, *uploaders*, dépendances de construction, dépendances, description, page d'accueil...
- ▶ Documentation : la Charte Debian, chapitre 5
<https://www.debian.org/doc/debian-policy/ch-controlfields.html>

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

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



Architecture : « all » ou « any »

Deux types de paquets binaires :

- ▶ Paquets ayant un contenu différent selon l'architecture Debian
 - ▶ Exemple : programme C
 - ▶ Architecture: any dans debian/control
 - ▶ Si ça ne fonctionne que sur certaines architectures :
Architecture: amd64 i386 ia64 hurd-i386
 - ▶ buildd.debian.org : construit les autres architectures à l'envoi
 - ▶ Nommés `paquet_version_architecture.deb`
- ▶ Paquets ayant le même contenu sur toutes les architectures
 - ▶ Exemple : bibliothèque Perl
 - ▶ Architecture: all dans debian/control
 - ▶ Nommé `paquet_version_all.deb`

Un même paquet source peut engendrer à la fois des paquets binaires
Architecture: any et Architecture: all



debian/rules

- ▶ Makefile
- ▶ Interface utilisée pour construire des paquets Debian
- ▶ Documenté dans la Charte Debian, chapitre 4.8
<https://www.debian.org/doc/debian-policy/ch-source#s-debianrules>
- ▶ Cibles requises :
 - ▶ build, build-arch, build-indep : doit effectuer toute la configuration et la compilation
 - ▶ binary, binary-arch, binary-indep : créent les paquets binaires
 - ▶ dpkg-buildpackage appellera binary pour construire tous les paquets ou binary-arch pour ne construire que les paquets
Architecture: any
 - ▶ clean : nettoie le répertoire des sources



Assistants d'empaquetage – debhelper

- ▶ Vous pourriez écrire du code shell dans le fichier `debian/rules`
 - ▶ Mieux : utilisez un *assistant d'empaquetage* (déjà le cas pour la plupart des paquets)
 - ▶ Le plus populaire : **debhelper** (utilisé par 98 % des paquets)
 - ▶ Objectifs :
 - ▶ Centraliser les tâches courantes dans des outils normalisés qui seront utilisés par tous les paquets
 - ▶ Corriger d'un coup des bogues de construction pour tous les paquets
- `dh_installdirs`, `dh_installchangelogs`, `dh_installdocs`, `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`, ...
- ▶ Appelé depuis `debian/rules`
 - ▶ Configurable avec des paramètres ou des fichiers dans `debian/`
`dirs`, `paquet.docs`, `paquet.examples`, `paquet.install`, `paquet.manpages`...
- ▶ Assistants tiers pour certains types de paquets : **python-support**,
dh_ocaml...
- ▶ `debian/compat` : version de compatibilité de Debhelper
 - ▶ Définit le comportement précis de `dh_*`
 - ▶ Nouvelle syntaxe : `Build-Depends: debhelper-compat (= 13)`



debian/rules en utilisant debhelper (1/2)

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

# Décommentez cette ligne pour passer en mode bavard.
#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
    # Ajoutez ici des commandes pour installer
    # le paquet dans debian/packagename.
    $(MAKE) DESTDIR=$(CURDIR)/debian/packagename install
```



debian/rules en utilisant debhelper (2/2)

```
# Construire ici les fichiers non spécifiques à une architecture.  
binary-indep: build install
```

```
# Construire ici les fichiers spécifiques à une architecture.  
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

- ▶ Avec debhelper, restent beaucoup de redondances entre les paquets
- ▶ Assistants de second niveau incluant des fonctionnalités courantes
 - ▶ p. ex. construction avec `./configure && make && make install`
- ▶ CDBS :
 - ▶ Introduit en 2005, basé sur la magie évoluée de *GNU make*
 - ▶ Documentation : `/usr/share/doc/cdbS/`
 - ▶ Gestion de Perl, Python, Ruby, GNOME, KDE, Java, Haskell. . .
 - ▶ Mais certaines personnes le détestent :
 - ▶ Il est parfois difficile à personnaliser : « *enchevêtrement complexe de makefiles et de variables d'environnement* »
 - ▶ Plus lent que l'utilisation seule de debhelper (beaucoup d'appels inutiles à `dh_*`)

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

# ajouter une action après la construction
build/monpaquet::
    /bin/bash debian/scripts/toto.sh
```



Dh (aussi appelé Debhelper 7, ou dh7)

- ▶ Introduit en 2008, avec l'objectif de remplacer CDBS
- ▶ Commande **dh** qui appelle `dh_*`
- ▶ Fichier `debian/rules` simple, ne contenant que les redéfinitions
- ▶ Plus facile à personnaliser que CDBS
- ▶ Doc : pages de man (`debhelper(7)`, `dh(1)`) et présentation à 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 classique vs CDBS vs dh

- ▶ Parts de marché :
debhelper classique : 15 % CDBS : 15 % dh : 68 %
- ▶ Lequel apprendre ?
 - ▶ Probablement un peu de chaque
 - ▶ Vous devez connaître debhelper pour utiliser dh et CDBS
 - ▶ Vous pourriez avoir à modifier des paquets CDBS
- ▶ Lequel utiliser pour un nouveau paquet ?
 - ▶ **dh** (seule solution de plus en plus utilisée)
 - ▶ Voir <https://trends.debian.net/#build-systems>



Plan

- 1 Introduction
- 2 Création des paquets source
- 3 Construire et tester les paquets**
- 4 Travaux pratiques n° 1 : modifier le paquet grep
- 5 Sujets avancés sur la construction de paquets
- 6 Maintenir des paquets dans Debian
- 7 Conclusions
- 8 Travaux pratiques supplémentaires
- 9 Solutions aux travaux pratiques



Construire les paquets

- ▶ `apt-get build-dep monpaquet`
Installer les *dépendances de construction* (pour un paquet dans Debian)
Ou `mk-build-deps -ir` (pour un paquet pas encore envoyé dans Debian)
- ▶ `debuild` : construire, tester avec `lintian`, signer avec GPG
- ▶ Vous pouvez aussi faire appel directement à `dpkg-buildpackage`
 - ▶ En général, avec `dpkg-buildpackage -us -uc`
- ▶ Il vaut mieux construire les paquets dans un environnement minimal
 - ▶ `pbuilder` – assistant pour la construction de paquets dans un *chroot*
Bonne documentation : <https://wiki.ubuntu.com/PbuilderHowto>
(optimisation : `cowbuilder ccache distcc`)
 - ▶ `schroot` et `sbuild` : utilisé sur les démons de construction Debian
(pas aussi simple que `pbuilder`, mais permet des copies LVM
voir : <https://help.ubuntu.com/community/SbuildLVMHowto>)
- ▶ Crée les fichiers `.deb` et un fichier `.changes`
 - ▶ `.changes` : décrit ce qui a été construit ; utilisé pour envoyer le paquet

Installation et test des paquets

- ▶ Installer le paquet : `debi` (utilise `.changes` pour savoir quoi installer)
- ▶ Afficher le contenu du paquet : `debc .. /monpaquet<TAB>.changes`
- ▶ Comparer le paquet avec une version précédente :
`debdiff ../monpaquet_1_*.changes ../monpaquet_2_*.changes`
ou pour comparer les sources :
`debdiff ../monpaquet_1_*.dsc ../monpaquet_2_*.dsc`
- ▶ Vérifier le paquet avec `lintian` (analyseur statique) :
`lintian ../monpaquet<TAB>.changes`
`lintian -i` : donne plus d'informations sur les erreurs
`lintian -EviIL +pedantic` : montre encore plus de problèmes
- ▶ Envoyer le paquet dans Debian (`dput`) (exige un peu de configuration)
- ▶ Gérer une archive privée avec `reprepro` ou `aptly`
Documentation :
<https://wiki.debian.org/HowToSetupADebianRepository>



Plan

- 1 Introduction
- 2 Création des paquets source
- 3 Construire et tester les paquets
- 4 Travaux pratiques n° 1 : modifier le paquet grep**
- 5 Sujets avancés sur la construction de paquets
- 6 Maintenir des paquets dans Debian
- 7 Conclusions
- 8 Travaux pratiques supplémentaires
- 9 Solutions aux travaux pratiques



Travaux pratiques n° 1 : modifier le paquet grep

- 1 Rendez-vous sur `http://ftp.debian.org/debian/pool/main/g/grep/` et téléchargez la version 2.12-2 du paquet
 - ▶ Si le paquet source n'est pas décompressé automatiquement, décompressez-le avec `dpkg-source -x grep_*.dsc`
- 2 Regardez les fichiers contenus dans `debian/`.
 - ▶ Combien de paquets binaires sont produits par ce paquet source ?
 - ▶ Quel assistant d'emballage ce paquet utilise-t-il ?
- 3 Construisez le paquet
- 4 Nous allons maintenant modifier le paquet. Ajoutez une entrée au journal des modifications et augmentez le numéro de version.
- 5 Désactivez maintenant la gestion des expressions rationnelles de Perl (c'est une option de `./configure`)
- 6 Reconstituez le paquet
- 7 Comparez le paquet d'origine et le nouveau avec `debdiff`
- 8 Installez le paquet nouvellement construit



Plan

- 1 Introduction
- 2 Création des paquets source
- 3 Construire et tester les paquets
- 4 Travaux pratiques n° 1 : modifier le paquet grep
- 5 Sujets avancés sur la construction de paquets**
- 6 Maintenir des paquets dans Debian
- 7 Conclusions
- 8 Travaux pratiques supplémentaires
- 9 Solutions aux travaux pratiques



debian/copyright

- ▶ Informations de copyright et de licence pour les sources et l'empaquetage
- ▶ Écrites traditionnellement dans un fichier texte
- ▶ Nouveau format en langage machine :

<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:
[TEXTE DE LA LICENCE]
```



Modifier les sources amont

Souvent nécessaire :

- ▶ Corriger des bogues ou faire des modifications spécifiques à Debian
- ▶ Rétroporter des corrections depuis une version amont plus récente

Plusieurs méthodes existent :

- ▶ Modifier directement les fichiers
 - ▶ Simple
 - ▶ Mais aucun moyen de suivre et de documenter les modifications
- ▶ Utiliser les systèmes de gestion de correctifs
 - ▶ Facilite l'intégration de vos modifications en amont
 - ▶ Facilite le partage des corrections avec les dérivées
 - ▶ Donne plus de visibilité à vos modifications

<http://patch-tracker.debian.org/> (actuellement en panne)



Systèmes de gestion de correctifs

- ▶ Principe : les modifications sont stockées sous forme de correctifs dans `debian/patches/`
- ▶ Correctifs appliqués et retirés lors de la construction
- ▶ Avant : plusieurs implémentations – *simple-patchsys (cdbs)*, *dpatch*, ***quilt***
 - ▶ Chacune prend en charge deux cibles `debian/rules` :
 - ▶ `debian/rules patch` : applique tous les correctifs
 - ▶ `debian/rules unpatch` : retire tous les correctifs
 - ▶ Plus de documentation :
<https://wiki.debian.org/debian/patches>
- ▶ **Nouveau format de paquet source avec système de gestion de correctifs intégré : 3.0 (quilt)**
 - ▶ Solution recommandée
 - ▶ Vous devez apprendre *quilt*
<http://perl-team.pages.debian.net/howto/quilt.html>
 - ▶ Outil indépendant du système de correctifs dans `devscripts` :
`edit-patch`



Documentation des correctifs

- ▶ En-têtes normalisés au début du correctif
- ▶ Documentation dans DEP-3 – Patch Tagging Guidelines (lignes directrices de l'étiquetage d'un correctif)
<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 {
```



Agir durant l'installation et la suppression

- ▶ Décompresser le paquet ne suffit pas toujours
- ▶ Créer/supprimer des utilisateurs système, démarrer/arrêter des services, gérer des *alternatives*
- ▶ Cela se fait dans *les scripts du responsable*
preinst, postinst, prerm, postrm
 - ▶ debhelper peut créer des bribes pour les actions classiques
- ▶ Documentation :
 - ▶ La Charte Debian, chapitre 6
<https://www.debian.org/doc/debian-policy/ch-maintainerscripts>
 - ▶ Manuel de référence du développeur, chapitre 6.4
<https://www.debian.org/doc/developers-reference/best-pkging-practices.html>
 - ▶ <https://people.debian.org/~srivasta/MaintainerScripts.html>
- ▶ Interagir avec l'utilisateur
 - ▶ Cela doit se faire avec **debconf**
 - ▶ Documentation : `debconf-devel(7)` (paquet `debconf-doc`)



Surveiller les versions amont

- ▶ Préciser dans `debian/watch` où chercher (voir `uscan(1)`)

```
version=3
```

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

- ▶ Il y a des systèmes de suivi automatique des nouvelles versions amonts, qui notifient le mainteneur sur plusieurs systèmes d'information dont <https://tracker.debian.org/> et <https://udd.debian.org/dmd/>
- ▶ `uscan` : lance une vérification à la main
- ▶ `uupdate` : tente de mettre à jour votre paquet vers la version la plus récente



Empaqueter avec un gestionnaire de versions :

- ▶ Plusieurs outils facilitent la gestion des branches et des étiquettes lors de votre empaquetage : `svn-buildpackage`, `git-buildpackage`
- ▶ Exemple : `git-buildpackage`
 - ▶ la branche `upstream` pour suivre les sources amont avec les étiquettes `upstream/version`
 - ▶ la branche `master` suit le paquet Debian
 - ▶ les étiquettes `debian/version` pour chaque envoi
 - ▶ la branche `pristine-tar` pour pouvoir reconstruire l'archive tar amont

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

- ▶ Les champs `Vcs-*` de `debian/control` pour localiser le dépôt
 - ▶ <https://wiki.debian.org/Salsa>

Vcs-Browser : <https://salsa.debian.org/debian/devscripts>

Vcs-Git : <https://salsa.debian.org/debian/devscripts.git>

Vcs-Browser : <https://salsa.debian.org/perl-team/modules/packages/libwww-perl>

Vcs-Git : <https://salsa.debian.org/perl-team/modules/packages/libwww-perl.git>

- ▶ Interface indépendante du VCS : `debcheckout`, `debcommit`, `debrelease`
 - ▶ `debcheckout grep` : rapatrie le paquet source du dépôt Git



Rétroportage de paquets

- ▶ But : utiliser une version plus récente du paquet sur un système plus ancien
p.ex. utiliser *mutt* de Debian *unstable* sur Debian *stable*
- ▶ Idée générale :
 - ▶ Prendre le paquet source de Debian unstable
 - ▶ Le modifier pour qu'il se construise et fonctionne correctement sur Debian stable
 - ▶ Parfois trivial (aucun changement nécessaire)
 - ▶ Parfois difficile
 - ▶ Parfois impossible (nombreuses dépendances indisponibles)
- ▶ Certains rétroportages sont fournis et maintenus par le projet Debian
<http://backports.debian.org/>

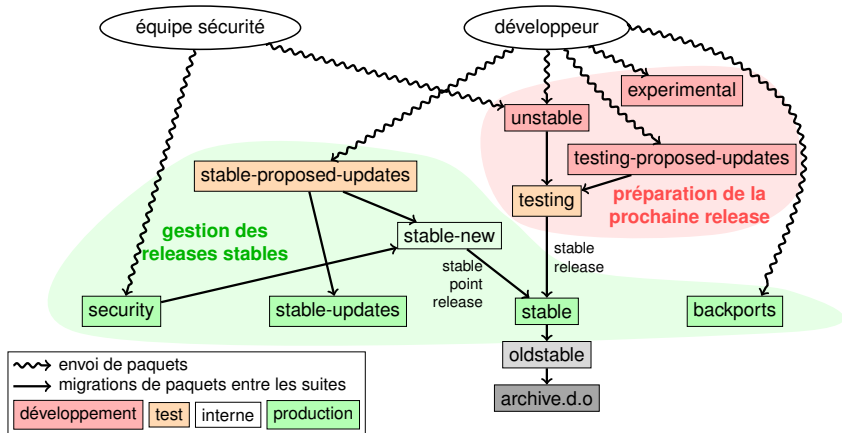


Plan

- 1 Introduction
- 2 Création des paquets source
- 3 Construire et tester les paquets
- 4 Travaux pratiques n° 1 : modifier le paquet grep
- 5 Sujets avancés sur la construction de paquets
- 6 Maintenir des paquets dans Debian**
- 7 Conclusions
- 8 Travaux pratiques supplémentaires
- 9 Solutions aux travaux pratiques



Archive Debian et suites



Basé sur un schéma par Antoine Beaupré. <https://salsa.debian.org/debian/package-cycle>



Suites pour le développement

- ▶ Les nouvelles versions de paquets sont envoyées dans **unstable** (**sid**)
- ▶ Les paquets migrent de **unstable** vers **testing** sur la base de plusieurs critères (par exemple, a été dans unstable depuis 10 jours, sans régression)
- ▶ Les nouveaux paquets peuvent aussi être envoyés vers :
 - ▶ **experimental** (pour des paquets plus *expérimentaux*, comme quand la nouvelle version n'est pas prête pour remplacer la version dans unstable)
 - ▶ **testing-proposed-updates**, pour mettre à jour la version dans **testing** sans passer par **unstable** (c'est rarement utilisé)



Freezer et releaser

- ▶ À un instant donné pendant le cycle d'une nouvelle version, l'équipe *release* décide de geler (*freezer*) testing : les migrations automatiques de **unstable** vers **testing** sont arrêtées, et remplacées par une relecture manuelle
- ▶ Quand l'équipe *release* considère **testing** comme prêt pour la sortie d'une nouvelle version :
 - ▶ La suite **testing** devient la nouvelle suite **stable**
 - ▶ De manière similaire, l'ancienne **stable** devient **oldstable**
 - ▶ Les versions plus supportées sont déplacées vers `archive.debian.org`
- ▶ Voir <https://release.debian.org/>



Suites des versions stables et leur gestion

- ▶ Plusieurs suites sont utilisées pour fournir les paquets des versions stables :
 - ▶ **stable** : la suite principale
 - ▶ **security** fournit les mises à jour de sécurité sur `security.debian.org` (utilisé par l'équipe sécurité). Les mises-à-jour sont annoncées sur la liste `debian-security-announce`
 - ▶ **stable-updates** : mises-à-jour non liées à la sécurité, mais qui doivent être installées de manière urgente (sans attendre la prochaine *point release*) : base de données d'antivirus, fuseaux horaires, etc. Annoncées sur la liste `debian-stable-announce`
 - ▶ **backports** : nouvelles versions majeures, basées sur la version dans **testing**
- ▶ La suite **stable** est mise à jour tous les quelques mois par des *stable point releases* (qui n'incluent que des corrections de bugs)
 - ▶ Les paquets visant la prochaine *stable point release* sont envoyés vers **stable-proposed-updates** et relus par l'équipe *release*
- ▶ La version **oldstable** a le même ensemble de suites



Plusieurs manières de contribuer à Debian

▶ La pire :

- 1 Empaqueter votre propre application
- 2 L'intégrer à Debian
- 3 Disparaître

▶ Les meilleures :

- ▶ S'impliquer dans des équipes d'empaquetage Debian
 - ▶ Beaucoup d'équipes se concentrent sur un ensemble de paquets et ont besoin d'aide
 - ▶ Liste disponible sur <https://wiki.debian.org/Teams>
 - ▶ Excellente façon d'apprendre de contributeurs plus expérimentés
- ▶ Adopter des paquets existants non maintenus (*paquets orphelins*)
- ▶ Apporter un nouveau logiciel à Debian
 - ▶ Seulement s'il est intéressant ou utile, s'il vous plaît
 - ▶ Y a-t-il une alternative déjà empaquetée pour Debian ?



Adopter des paquets orphelins

- ▶ Beaucoup de paquets non maintenus dans Debian
- ▶ Liste complète et marche à suivre :
<https://www.debian.org/devel/wnpp/>
- ▶ Ceux installés sur votre machine : `wnpp-alert`
Ceux installés sur votre machine : `wnpp-alert`
Ou mieux : `how-can-i-help`
- ▶ Différents états :
 - ▶ **O**rphelin : le paquet n'est pas maintenu
Adoptez-le s'il vous sied
 - ▶ **RFA** : **R**equest **F**or **A**dopter (cherche un adoptant)
Le responsable cherche un adoptant, mais il continue son travail en attendant
Adoptez-le s'il vous sied. L'envoi d'un courriel au responsable actuel est poli
 - ▶ **ITA** : **I**ntent **T**o **A**dopt (en cours d'adoption)
Quelqu'un prévoit d'adopter le paquet. Vous pourriez proposer votre aide !
 - ▶ **RFH** : **R**equest **F**or **H**elp (recherche d'aide)
Le responsable cherche de l'aide
- ▶ Certains paquets non maintenus ne sont pas identifiés comme tels → pas encore officiellement orphelins

Adopter un paquet : un exemple

```
From: Vous <vous@votredomaine>  
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,

Vous

- ▶ Il est poli de contacter le responsable précédent (surtout si le paquet était signalé comme cherchant un adoptant et non comme orphelin)
- ▶ Très bonne idée aussi de contacter le projet amont



Intégrer votre paquet dans Debian

- ▶ Aucun besoin d'un statut officiel pour intégrer son paquet dans Debian
 - ➊ Soumettez un bogue **ITP** (Intent To Package) avec `reportbug wnpp`
 - ➋ Préparez un paquet source
 - ➌ Trouvez un développeur Debian qui va parrainer votre paquet
- ▶ Statuts officiels (quand vous serez un responsable de paquets expérimenté) :
 - ▶ **Debian Maintainer (DM)** :
Droit d'envoyer vos propres paquets
Voir <https://wiki.debian.org/DebianMaintainer>
 - ▶ **Debian Developer (DD)** :
Membre du projet Debian
Peut voter et envoyer n'importe quel paquet



Points à vérifier avant de demander un parrainage

- ▶ Debian met **fortement l'accent sur la qualité**
- ▶ En général, **les parrains et marraines sont difficiles à trouver et très occupés**
 - ▶ Assurez-vous que votre paquet est prêt avant de demander un parrainage
- ▶ Points à vérifier :
 - ▶ Évitez les oublis de dépendances de construction : assurez-vous que la construction de votre paquet fonctionne dans un environnement *chroot sid* propre
 - ▶ L'utilisation de `pbuilder` est recommandée
 - ▶ Lancez `lintian -EviIL +pedantic` sur votre paquet
 - ▶ Vous devez corriger les erreurs et faire de votre mieux pour corriger les autres problèmes
 - ▶ Testez largement votre paquet, bien sûr
- ▶ En cas de doute, demandez de l'aide



Où trouver de l'aide ?

L'aide dont vous avez besoin :

- ▶ conseils et réponses à vos questions, relecture de code
- ▶ parrainage pour les envois, une fois votre paquet prêt

Vous pouvez obtenir de l'aide :

- ▶ **autres membres d'une équipe d'empaquetage : la meilleure solution**
 - ▶ liste des équipes : <https://wiki.debian.org/Teams>
- ▶ le groupe **Debian Mentors** (si le paquet ne correspond à aucune équipe)
 - ▶ <https://wiki.debian.org/DebianMentorsFaq>
 - ▶ liste de diffusion : debian-mentors@lists.debian.org
(une autre manière d'apprendre par hasard)
 - ▶ IRC : #debian-mentors sur irc.debian.org
 - ▶ <http://mentors.debian.net/>
 - ▶ documentation : <http://mentors.debian.net/intro-maintainers>
- ▶ **listes de diffusion localisées** (pour obtenir de l'aide dans votre langue)
 - ▶ debian-devel-{french,italian,portuguese,spanish}@lists.d.o
 - ▶ liste complète : <https://lists.debian.org/devel.html>
 - ▶ ou les listes d'utilisateurs : <https://lists.debian.org/users.html>

Plus de documentation :

- ▶ Le coin des développeurs Debian
<https://www.debian.org/devel/>
Liens vers de nombreuses ressources sur le développement Debian
- ▶ Guide pour les nouveaux mainteneurs
<https://www.debian.org/doc/manuals/debmake-doc/>
- ▶ Manuel de référence du développeur Debian
<https://www.debian.org/doc/developers-reference/>
Essentiellement sur les procédures Debian mais contient aussi quelques bonnes pratiques d'empaquetage (chapitre 6)
- ▶ La Charte Debian
<https://www.debian.org/doc/debian-policy/>
 - ▶ Toutes les exigences que doit satisfaire chaque paquet
 - ▶ Règles spécifiques pour Perl, Java, Python...
- ▶ Guide d'empaquetage d'Ubuntu
<https://packaging.ubuntu.com/html/>



Tableau de bord Debian pour les responsables

- ▶ **Pour une vision par paquet source :**
`https://tracker.debian.org/dpkg`
- ▶ **Pour une vision par responsable/équipe :** aperçu des paquets d'un développeur (DDPO : Developer's Packages Overview)
`https://qa.debian.org/developer.php?login=
pkg-ruby-extras-maintainers@lists.alioth.debian.org`
- ▶ **Pour une vision en liste de tâches :** tableau de bord du responsable Debian (DMD : Debian Maintainer Dashboard)
`https://udd.debian.org/dmd/`



Utilisation du système de suivi des bogues (BTS)

- ▶ Une façon unique de gérer les bogues
 - ▶ interface web pour visualiser les bogues
 - ▶ interface par courrier électronique pour modifier les bogues
- ▶ Ajouter des informations complémentaires aux bogues :
 - ▶ écrivez un message à `123456@bugs.debian.org` (pour mettre en copie la personne ayant soumis le bogue, ajoutez `123456-submitter@bugs.debian.org`)
- ▶ Changer le statut d'un bogue :
 - ▶ envoyez des commandes à `control@bugs.debian.org`
 - ▶ interface en ligne de commande : commande `bts` dans le paquet `devscripts`
 - ▶ documentation : <https://www.debian.org/Bugs/server-control>
- ▶ Soumettre un rapport de bogue : utilisez `reportbug`
 - ▶ normalement utilisé avec un serveur de courrier local : installez `ssmtp` ou `nullmailer`
 - ▶ ou utilisez `reportbug --template`, puis envoyez « à la main » le message à l'adresse `submit@bugs.debian.org`



Utilisation du BTS : exemples

- ▶ Envoyer un message au bogue et à la personne qui l'a soumis :
`https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680822#10`
- ▶ Étiqueter et changer la sévérité :
`https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680227#10`
- ▶ Réattribuer, changer la sévérité, changer le titre... :
`https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680822#93`
 - ▶ `notfound`, `found`, `notfixed`, `fixed` utilisé pour le **suivi de version**
Lisez `https://wiki.debian.org/HowtoUseBTS#Version_tracking`
- ▶ Utiliser des étiquettes utilisateur : `https://bugs.debian.org/cgi-bin/bugreport.cgi?msg=42;bug=642267`
Consultez `https://wiki.debian.org/bugs.debian.org/usertags`
- ▶ Documentation du BTS :
 - ▶ `https://www.debian.org/Bugs/`
 - ▶ `https://wiki.debian.org/HowtoUseBTS`



Plus intéressé par Ubuntu ?

- ▶ Ubuntu gère essentiellement les différences avec Debian
- ▶ Pas de concentration sur des paquets spécifiques
Mais une collaboration avec les équipes Debian
- ▶ Il est recommandé en général d'envoyer les nouveaux paquets d'abord dans Debian
<https://wiki.ubuntu.com/UbuntuDevelopment/NewPackages>
- ▶ Peut-être une meilleure idée :
 - ▶ S'impliquer dans une équipe Debian et faire le lien avec Ubuntu
 - ▶ Faciliter la réduction des divergences, trier les bogues sur Launchpad
 - ▶ Beaucoup d'outils Debian peuvent aider :
 - ▶ Colonne Ubuntu sur l'aperçu des paquets du développeur
 - ▶ Encart Ubuntu sur le système de suivi des paquets
 - ▶ Réception des courriels de bogues Launchpad au moyen du PTS



Plan

- 1 Introduction
- 2 Création des paquets source
- 3 Construire et tester les paquets
- 4 Travaux pratiques n° 1 : modifier le paquet grep
- 5 Sujets avancés sur la construction de paquets
- 6 Maintenir des paquets dans Debian
- 7 Conclusions**
- 8 Travaux pratiques supplémentaires
- 9 Solutions aux travaux pratiques



Conclusions

- ▶ Vous avez un aperçu complet de la construction de paquets Debian
- ▶ Mais vous devrez lire plus de documentation
- ▶ Les bonnes pratiques ont évolué avec le temps
 - ▶ Si vous n'êtes pas sûr, utilisez l'assistant d'empaquetage de paquets **dh**, et le format **3.0 (quilt)**

Vos retours à : **packaging-tutorial@packages.debian.org**



Mentions légales

Copyright © 2011–2019 Lucas Nussbaum – lucas@debian.org

Ce document est un logiciel libre : vous pouvez le redistribuer et le modifier, selon votre choix, sous :

- ▶ les termes de la General Public License GNU publiée par la Fondation du logiciel libre, version 3 de la License, ou (si vous préférez) toute version supérieure.
<http://www.gnu.org/licenses/gpl.html>
- ▶ les termes de la licence Creative Commons Attribution-ShareAlike 3.0 Unported.
<http://creativecommons.org/licenses/by-sa/3.0/>



Contribuer à ce tutoriel

▶ Contribuer :

- ▶ `apt-get source packaging-tutorial`
- ▶ `debcheckout packaging-tutorial`
- ▶ `git clone`
`https://salsa.debian.org/debian/packaging-tutorial.git`
- ▶ `https://salsa.debian.org/debian/packaging-tutorial`
- ▶ Soumettez des rapports de bogues :
`bugs.debian.org/src:packaging-tutorial`

▶ Envoyez vos retours :

- ▶ `mailto:packaging-tutorial@packages.debian.org`
 - ▶ Qu'est-ce qui doit être ajouté à ce tutoriel ?
 - ▶ Qu'est-ce qui doit être amélioré ?
- ▶ `reportbug packaging-tutorial`



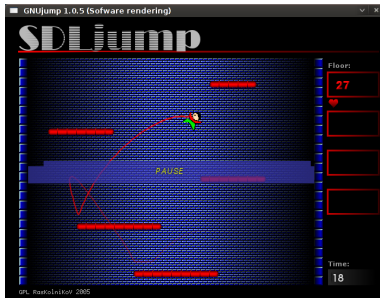
Plan

- 1 Introduction
- 2 Création des paquets source
- 3 Construire et tester les paquets
- 4 Travaux pratiques n° 1 : modifier le paquet grep
- 5 Sujets avancés sur la construction de paquets
- 6 Maintenir des paquets dans Debian
- 7 Conclusions
- 8 Travaux pratiques supplémentaires**
- 9 Solutions aux travaux pratiques



Travaux pratiques n° 2 : emballer GNUJump

- 1 Téléchargez GNUJump 1.0.8 depuis
<http://ftp.gnu.org/gnu/gnujump/gnujump-1.0.8.tar.gz>
- 2 Créez un paquet Debian
 - ▶ Installez les dépendances de construction du paquet
 - ▶ Corrigez des bugs
 - ▶ Vous obtenez un paquet de base fonctionnel
 - ▶ Terminez en complétant `debian/control` et d'autres fichiers
- 3 Profitez



Travaux pratiques n° 2 : GNUJump (astuces)

- ▶ Créez un paquet source basique : `dh_make`
- ▶ Pour commencer, créer un fichier source `1.0` est plus facile que `3.0` (*quilt*) (à changer dans `debian/source/format`)
- ▶ Pour chercher des dépendances de construction manquantes, trouvez un fichier manquant, et utilisez `apt-file` pour trouver le paquet manquant correspondant
- ▶ Si vous rencontrez cette erreur :

```
/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
```

Vous devez ajouter `-lm` sur la ligne de commande du lieur :
Editez `src/Makefile.am` et remplacez

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

par

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

Puis exécutez `autoreconf -i`



Travaux pratiques n° 3 : une bibliothèque Java

- 1 Jetez un coup d'œil sur la documentation pour la construction de paquets 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 Téléchargez IRCLib sur <http://moepii.sourceforge.net/>
- 3 Empaquetez-la



Travaux pratiques n° 4 : emballer un gem Ruby

- 1 Jetez un coup d'œil sur la documentation pour la construction de paquets Ruby :
 - ▶ <https://wiki.debian.org/Ruby>
 - ▶ <https://wiki.debian.org/Teams/Ruby>
 - ▶ <https://wiki.debian.org/Teams/Ruby/Packaging>
 - ▶ `gem2deb(1)`, `dh_ruby(1)` (dans le paquet `gem2deb`)
- 2 Créez un paquet source Debian élémentaire à partir du gem `peach` :
`gem2deb peach`
- 3 Améliorez-le pour qu'il devienne un paquet Debian à part entière



Travaux pratiques n° 5 : module Perl

- 1 Jetez un coup d'œil sur la documentation pour la construction de paquets Perl :
 - ▶ <https://perl-team.pages.debian.net>
 - ▶ <https://wiki.debian.org/Teams/DebianPerlGroup>
 - ▶ `dh-make-perl(1)`, `dpt(1)` (in the `pkg-perl-tools` package)
- 2 Créez un paquet source Debian élémentaire à partir du module `Acme` de CPAN :`dh-make-perl --cpan Acme`
- 3 Améliorez-le pour qu'il devienne un paquet Debian à part entière



Plan

- 1 Introduction
- 2 Création des paquets source
- 3 Construire et tester les paquets
- 4 Travaux pratiques n° 1 : modifier le paquet grep
- 5 Sujets avancés sur la construction de paquets
- 6 Maintenir des paquets dans Debian
- 7 Conclusions
- 8 Travaux pratiques supplémentaires
- 9 Solutions aux travaux pratiques



Solutions aux travaux pratiques



Travaux pratiques n° 1 : modifier le paquet grep

- 1 Rendez-vous sur `http://ftp.debian.org/debian/pool/main/g/grep/` et téléchargez la version 2.12-2 du paquet
- 2 Regardez les fichiers contenus dans `debian/`.
 - ▶ Combien de paquets binaires sont produits par ce paquet source ?
 - ▶ Quel assistant d'empaquetage ce paquet utilise-t-il ?
- 3 Construisez le paquet
- 4 Nous allons maintenant modifier le paquet. Ajoutez une entrée au journal des modifications et augmentez le numéro de version.
- 5 Désactivez maintenant la gestion des expressions rationnelles de Perl (c'est une option de `./configure`)
- 6 Reconstituez le paquet
- 7 Comparez le paquet d'origine et le nouveau avec `debdiff`
- 8 Installez le paquet nouvellement construit



Récupérer les sources

- ➊ Rendez-vous sur `http://ftp.debian.org/debian/pool/main/g/grep/` et téléchargez la version 2.12-2 du paquet
 - ▶ Utilisez `dget` pour télécharger le fichier `.dsc` :
`dget http://cdn.debian.net/debian/pool/main/g/grep/grep_2.12-2.dsc`
 - ▶ Si vous avez des lignes `deb-src` pour une version de Debian qui contient `grep` version 2.12-2 (à vérifier sur `https://tracker.debian.org/grep`), vous pouvez utiliser :
`apt-get source grep=2.12-2`
ou `apt-get source grep/release` (par exemple `grep/stable`)
ou, si vous avez de la chance : `apt-get source grep`
 - ▶ Le paquet source de `grep` se compose de trois fichiers :
 - ▶ `grep_2.12-2.dsc`
 - ▶ `grep_2.12-2.debian.tar.bz2`
 - ▶ `grep_2.12.orig.tar.bz2`
- C'est le cas typique du format « 3.0 (quilt) »
- ▶ Si nécessaire, décompressez le paquet source avec
`dpkg-source -x grep_2.12-2.dsc`



Faites le tour et construisez le paquet

- 2 Regardez les fichiers contenus dans `debian/`
 - ▶ Combien de paquets binaires sont produits par ce paquet source ?
 - ▶ Quel assistant d'empaquetage ce paquet utilise-t-il ?
 - ▶ D'après `debian/control`, ce paquet ne génère qu'un seul paquet binaire, nommé `grep`.
 - ▶ D'après `debian/rules`, ce paquet est un exemple typique de construction avec l'assistant *classique* `debhelper`, n'utilisant ni *CDBS* ni *dh*. On peut voir les différents appels aux commandes `dh_*` dans `debian/rules`.
- 3 Construisez le paquet
 - ▶ Utilisez la commande `apt-get build-dep grep` pour installer les dépendances de construction
 - ▶ Puis `debuild` ou `dpkg-buildpackage -us -uc` (prend environ 1 min)



Éditer le journal des modifications

- ④ Nous allons maintenant modifier le paquet. Ajoutez une entrée au journal des modifications et augmentez le numéro de version.
- ▶ `debian/changelog` est un fichier texte. Vous pourriez l'éditer et ajouter une nouvelle entrée à la main.
- ▶ Vous pouvez aussi utiliser `dch -i`, qui ajoutera une entrée et ouvrira un éditeur
- ▶ On peut définir son nom et son adresse électronique via les variables d'environnement `DEBFULLNAME` et `DEBEMAIL`
- ▶ Reconstituez le paquet : une nouvelle version du paquet est construite
- ▶ Le système des versions est décrit à la section 5.6.12 de la Charte Debian <https://www.debian.org/doc/debian-policy/ch-controlfields>



Désactiver les expressions rationnelles Perl

- 5 Désactivez maintenant la gestion des expressions rationnelles de Perl (c'est une option de `./configure`)
- 6 Reconstituez le paquet
 - ▶ Vérifiez avec `./configure --help` : l'option pour désactiver les expressions rationnelles Perl est `--disable-perl-regexp`
 - ▶ Éditez `debian/rules` et cherchez la ligne `./configure`
 - ▶ Ajoutez `--disable-perl-regexp`
 - ▶ Reconstituez avec `debuild` ou `dpkg-buildpackage -us -uc`



Comparer et tester les paquets

- 7 Comparez le paquet d'origine et le nouveau avec `debdiff`
 - 8 Installez le paquet nouvellement construit
- ▶ Comparez les paquets binaires : `debdiff ../changes`
 - ▶ Comparez les paquets source : `debdiff ../dsc`
 - ▶ Installez le paquet nouvellement construit : `debi`
ou `dpkg -i ../grep_<TAB>`
 - ▶ `grep -P foo` ne fonctionne plus !

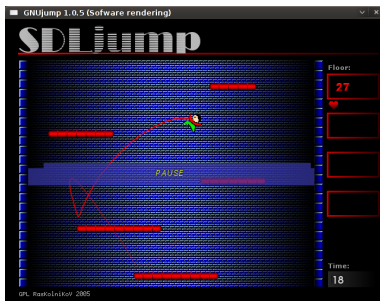
Réinstallez la version précédente du paquet :

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



Travaux pratiques n° 2 : emballer GNUjump

- 1 Téléchargez GNUjump 1.0.8 depuis
<http://ftp.gnu.org/gnu/gnujump/gnujump-1.0.8.tar.gz>
- 2 Créez un paquet Debian
 - ▶ Installez les dépendances de construction du paquet
 - ▶ Vous obtenez un paquet de base fonctionnel
 - ▶ Terminez en complétant `debian/control` et d'autres fichiers
- 3 Profitez



Pas à pas...

- ▶ `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`
 - ▶ Type de paquet : un seul binaire (pour l'instant)

```
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
```



Pas à pas... (2)

- ▶ Regardez `debian/changelog`, `debian/rules`, `debian/control` (remplis automatiquement par **dh_make**)
- ▶ Dans `debian/control` :
Build-Depends: `debhelper (>= 7.0.50)`, `autotools-dev`
Liste des *dépendances de construction* = paquets nécessaires pour construire le paquet
- ▶ Essayez de construire le paquet comme ça, avec `debuild` (grâce à la magie de **dh**)
 - ▶ Et ajoutez des dépendances jusqu'à ce que la construction puisse se terminer
 - ▶ Astuce : utilisez `apt-cache search` et `apt-file` pour chercher les paquets manquants
 - ▶ Exemple :

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

→ Ajoutez **libsdl1.2-dev** à Build-Depends et installez-le.

- ▶ Mieux : construction dans un environnement propre avec **pbuilder**



Pas à pas... (3)

- ▶ Les dépendances de construction nécessaires sont `libsdl1.2-dev`, `libsdl-image1.2-dev`, `libsdl-mixer1.2-dev`
- ▶ Ensuite, vous devriez rencontrer une autre erreur :

```
/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
```

- ▶ Ce problème est causé par la *pourriture logicielle* : `gnujump` n'a pas été adapté suite à des changements du lieu.
- ▶ Si vous utilisez le format source **1.0**, vous pouvez directement changer les sources amont.
 - ▶ Éditer `src/Makefile.am` and remplacer
`gnujump_LDFLAGS = $(all_libraries)`
par
`gnujump_LDFLAGS = -Wl,--as-needed`
`gnujump_LDADD = $(all_libraries) -lm`
 - ▶ Puis exécutez `autoreconf -i`



Pas à pas... (4)

- ▶ Si vous utilisez le format source **3.0 (quilt)**, utilisez `quilt` pour préparer un patch. (voir <https://wiki.debian.org/UsingQuilt>)
 - ▶ `export QUILT_PATCHES=debian/patches`
 - ▶ `mkdir debian/patches`
`quilt new linker-fixes.patch`
`quilt add src/Makefile.am`
 - ▶ Éditer `src/Makefile.am` and remplacer
`gnujump_LDFLAGS = $(all_libraries)`

`par`

`gnujump_LDFLAGS = -Wl,--as-needed`
`gnujump_LDADD = $(all_libraries) -lm`
 - ▶ `quilt refresh`
 - ▶ comme `src/Makefile.am` a été changé, `autoreconf` doit être appelé pendant la compilation. Pour le faire automatiquement avec `dh`, changer l'appel à `dh` dans `debian/rules` de : `dh $ --with autotools-dev`
en : `dh $ --with autotools-dev --with autoreconf`



Pas à pas... (5)

- ▶ Le paquet devrait maintenant se construire correctement.
- ▶ Utilisez `debnc` pour lister le contenu du paquet créé, et `debi` pour l'installer et le tester.
- ▶ Testez le paquet avec `lintian`
 - ▶ Absence de problèmes signalés par `lintian` recommandée (même si pas strictement nécessaire) pour les paquets envoyés dans Debian
 - ▶ Détection de plus de problèmes : `lintian -EviIL +pedantic`
 - ▶ Quelques indices :
 - ▶ Supprimez les fichiers inutiles dans `debian/`
 - ▶ Complétez `debian/control`
 - ▶ Installez l'exécutable dans `/usr/games` en surchargeant `dh_auto_configure`
 - ▶ Utilisez les attributs de consolidation du compilateur pour augmenter la sécurité.
Consultez <https://wiki.debian.org/Hardening>



Pas à pas... (6)

- ▶ Comparez votre paquet avec celui déjà empaqueté dans Debian :
 - ▶ Il rassemble les fichiers de données dans un deuxième paquet, identique pour toutes les architectures (→ gain de place dans l'archive Debian)
 - ▶ Il installe aussi un fichier .desktop (pour les menus GNOME/KDE) et l'intègre dans le menu Debian
 - ▶ Il corrige de petits problèmes en utilisant des correctifs



Travaux pratiques n° 3 : une bibliothèque Java

- 1 Jetez un coup d'œil sur la documentation pour la construction de paquets 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 Téléchargez IRCLib sur <http://moepii.sourceforge.net/>
- 3 Empaquetez-la



Pas à pas...

- ▶ `apt-get install javahelper`
- ▶ Créez un paquet source de base : `jh_makepkg`
 - ▶ Bibliothèque
 - ▶ Aucun
 - ▶ Compilateur et exécution libres fournis par défaut
- ▶ Regardez et corrigez `debian/*`
- ▶ `dpkg-buildpackage -us -uc OU debuild`
- ▶ `lintian, debc, etc.`
- ▶ Comparez votre résultat avec le paquet source `libirclib-java`



Travaux pratiques n° 4 : emballer un gem Ruby

- 1 Jetez un coup d'œil sur la documentation pour la construction de paquets Ruby :
 - ▶ <https://wiki.debian.org/Ruby>
 - ▶ <https://wiki.debian.org/Teams/Ruby>
 - ▶ <https://wiki.debian.org/Teams/Ruby/Packaging>
 - ▶ `gem2deb(1)`, `dh_ruby(1)` (dans le paquet `gem2deb`)
- 2 Créez un paquet source Debian élémentaire à partir du gem `peach` :
`gem2deb peach`
- 3 Améliorez-le pour qu'il devienne un paquet Debian à part entière



Pas à pas...

gem2deb peach :

- ▶ Télécharge le gem depuis rubygems.org
- ▶ Crée une archive `.orig.tar.gz` adéquate, et la décompresse
- ▶ Initialise un paquet source Debian basé sur les métadonnées du gem
 - ▶ Nommé `ruby-gemname`
- ▶ Tente de construire le paquet binaire Debian (ceci peut échouer)

dh_ruby (inclus dans `gem2deb`) effectue les tâches spécifiques à Ruby :

- ▶ Construction d'extensions C pour chaque version de Ruby
- ▶ Copie les fichiers dans leur répertoire de destination
- ▶ Mise à jour des interpréteurs à utiliser (« shebangs ») pour les scripts exécutables
- ▶ Exécution des tests définis dans `debian/ruby-tests.rb`, `debian/ruby-tests.rake`, ou `debian/ruby-test-files.yaml` ainsi que d'autres vérifications



Pas à pas... (2)

Améliorez le paquet créé :

- ▶ Exécutez `debclean` pour nettoyer l'arborescence. Regardez `debian/`.
- ▶ `changelog` et `compat` devraient être corrects
- ▶ Éditez `debian/control` : améliorez `Description`
- ▶ Écrivez un fichier `copyright` approprié se basant sur les fichiers amont
- ▶ Construisez le paquet
- ▶ Comparez votre paquet avec le paquet `ruby-peach` présent dans l'archive Debian



Travaux pratiques n° 5 : module Perl

- 1 Jetez un coup d'œil sur la documentation pour la construction de paquets Perl :
 - ▶ <https://perl-team.pages.debian.net>
 - ▶ <https://wiki.debian.org/Teams/DebianPerlGroup>
 - ▶ `dh-make-perl(1)`, `dpt(1)` (in the `pkg-perl-tools` package)
- 2 Créez un paquet source Debian élémentaire à partir du module `Acme` de CPAN :`dh-make-perl --cpan Acme`
- 3 Améliorez-le pour qu'il devienne un paquet Debian à part entière



Pas à pas...

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

- ▶ Télécharge l'archive de CPAN
- ▶ Crée une archive `.orig.tar.gz` adéquate, et la décompresse
- ▶ Initialise un paquet source Debian basé sur les métadonnées de CPAN
 - ▶ Named `libdistname-perl`



Pas à pas... (2)

Améliorez le paquet créé :

- ▶ `debian/changelog`, `debian/compat`, `debian/libacme-perl.docs`, and `debian/watch` devraient être corrects
- ▶ Éditez `debian/control` : améliorez `Description`, et enlevez le texte à la fin
- ▶ Éditez `debian/copyright` : enlevez le premier paragraphe en haut, ajoutez les années du copyright aux paragraphes `Files: *`



Traduction

Ce tutoriel a été traduit de l'anglais par Cédric Boutillier, Jean-Philippe Mengual et l'équipe francophone de traduction.

Veuillez signaler toute erreur de traduction ou adresser vos commentaires par courrier électronique, à l'adresse `<debian-l10n-french@lists.debian.org>`.

