

Inleiding tot het maken van Debian-pakketten

Lucas Nussbaum

`packaging-tutorial@packages.debian.org`

Vertaling naar het Nederlands door
Frans Spiesschaert
en het Nederlandstalig lokalisatieteam

version 0.30 – 2024-03-16



Over deze handleiding

- ▶ Doel: **u vertellen wat u echt moet weten over het maken van Debian-pakketten**
 - ▶ Bestaande pakketten aanpassen
 - ▶ Uw eigen pakketten maken
 - ▶ In interactie treden met de Debian-gemeenschap
 - ▶ Een intensieve gebruiker van Debian worden
- ▶ Behandelt de belangrijkste punten, maar is niet volledig
 - ▶ U zult meer documentatie moeten lezen
- ▶ Het grootste deel van de inhoud is ook van toepassing op de van Debian afgeleide distributies
 - ▶ Met inbegrip van Ubuntu



Overzicht

- 1 Inleiding
- 2 Broncodepakketten maken
- 3 Pakketten bouwen en testen
- 4 Praktijkoefening 1: het grep-pakket aanpassen
- 5 Geavanceerde verpakkingsonderwerpen
- 6 Pakketten onderhouden in Debian
- 7 Conclusies
- 8 Extra praktijkoefeningen
- 9 Oplossingen voor de praktijkoefeningen



Overzicht

- 1 Inleiding
- 2 Broncodepakketten maken
- 3 Pakketten bouwen en testen
- 4 Praktijkoefening 1: het grep-pakket aanpassen
- 5 Geavanceerde verpakkingsonderwerpen
- 6 Pakketten onderhouden in Debian
- 7 Conclusies
- 8 Extra praktijkoefeningen
- 9 Oplossingen voor de praktijkoefeningen



Debian

- ▶ **GNU/Linux distributie**
- ▶ eerste belangrijke distributie die “openlijk in de geest van GNU” ontwikkeld werd
- ▶ **Niet-commercieel**, samen gemaakt door meer dan 1.000 vrijwilligers
- ▶ drie belangrijke kenmerken:
 - ▶ **Kwaliteit** – cultuur van technische uitmuntendheid
We geven een release vrij wanneer deze klaar is
 - ▶ **Vrijheid** – ontwikkelaars en gebruikers zijn gebonden door het *Sociaal Contract*
Bevorderen van de cultuur van Vrije Software sinds 1993
 - ▶ **Onafhankelijkheid** – geen (enkel) bedrijf babysit op Debian
En een open besluitvormingproces (*doe-ocratie + democratie*)
- ▶ **Liefhebber** in de beste betekenis: gedaan uit liefde ervoor



Debian-pakketten

- ▶ **.deb**-bestanden (binaire pakketten)
- ▶ Een zeer krachtige en handige manier om software naar gebruikers te brengen
- ▶ Een van de twee meest voorkomende pakketformaten (samen met RPM)
- ▶ Universeel:
 - ▶ 30.000 binaire pakketten in Debian
→ het merendeel van de beschikbare vrije software wordt in Debian verpakt!
 - ▶ Voor 12 ports (architecturen), waaronder 2 niet-Linux (Hurd; KFreeBSD)
 - ▶ Ook gebruikt door 120 van Debian afgeleide distributies



De Deb-pakketindeling

- ▶ .deb-bestand: een ar-archief

```
$ 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: versie van de deb-bestandsindeling, "2.0\n"
 - ▶ control.tar.gz: metadata over het pakket
control, md5sums, (pre|post)(rm|inst), triggers, shlibs,...
 - ▶ data.tar.gz: data-bestanden van het pakket
- ▶ U zou uw .deb-bestanden handmatig kunnen aanmaken
http://tldp.org/HOWTO/html_single/Debian-Binary-Package-Building-HOWTO/
 - ▶ Maar de meeste mensen doen het niet op die manier

Deze inleiding: Debian-pakketten maken op de manier van Debian



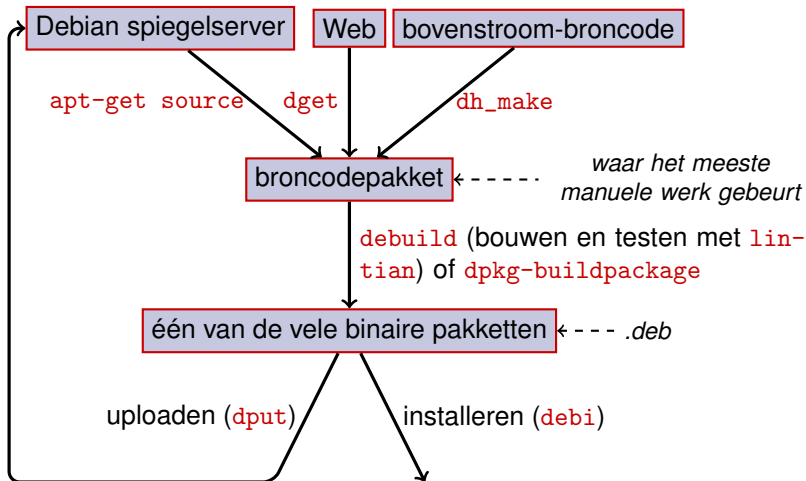
Hulpmiddelen die u nodig heeft

- ▶ Een Debian- (of Ubuntu)-systeem (met systeembeheerders-toegang)
- ▶ Enkele pakketten:
 - ▶ **build-essential**: bevat de vereisten van de pakketten waarvan wordt aangenomen dat ze beschikbaar zijn op de machine van de ontwikkelaar (onnodig om deze te vermelden in het controleveld `Build-Depends`: van uw pakket)
 - ▶ waaronder de vereiste **dpkg-dev**, welke basale Debian-specifieke hulpmiddelen voor het aanmaken van pakketten bevat
 - ▶ **devscripts**: bevat veel nuttige scripts voor Debian-onderhouders

Veel andere hulpmiddelen zullen ook later vermeld worden, zoals **debhelper**, **cdb**s, **quilt**, **pbuilder**, **sbuild**, **lintian**, **svn-buildpackage**, **git-buildpackage**, ...
Installeer deze wanneer u ze nodig heeft.



Algemene werkwijze bij het verpakken



Voorbeeld: dash opnieuw bouwen

- 1 Installeer pakketten nodig om dash te bouwen, plus devscripts

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

(vereist deb-src-regels in /etc/apt/sources.list)

```
sudo apt-get install --no-install-recommends devscripts fakeroot
```
- 2 Maak een werkmap aan en ga er binnen:

```
mkdir /tmp/debian-tutorial ; cd /tmp/debian-tutorial
```
- 3 Haal het broncodepakket dash op

```
apt-get source dash
```

(Daarvoor moet u deb-src-regels hebben in uw /etc/apt/sources.list)
- 4 Bouw het pakket

```
cd dash-*
```

```
debuild -us -uc
```

(-us -uc om pakket niet te ondertekenen met GPG)
- 5 Controleer of het werkte
 - ▶ Er staan enkele nieuwe .deb-bestanden in de bovenliggende map
- 6 Bekijk de map debian/
 - ▶ Daar vindt het verpakkingswerk plaats



Overzicht

- 1 Inleiding
- 2 Broncodepakketten maken
- 3 Pakketten bouwen en testen
- 4 Praktijkoefening 1: het grep-pakket aanpassen
- 5 Geavanceerde verpakkingsonderwerpen
- 6 Pakketten onderhouden in Debian
- 7 Conclusies
- 8 Extra praktijkoefeningen
- 9 Oplossingen voor de praktijkoefeningen



Broncodepakket

- ▶ Eén broncodepakket kan meerdere binaire pakketten geven
bijv. de broncode `libtar` genereert de binaire pakketten `libtar0` en `libtar-dev`
- ▶ Twee soorten pakketten: (gebruik bij twijfel non-native)
 - ▶ Native (eigen) pakketten: normaal Debian-specifieke software (*dpkg*, *apt*)
 - ▶ Non-native (niet-eigen) pakketten: buiten Debian ontwikkelde software
- ▶ Hoofdbestand: `.dsc` (meta-gegevens)
- ▶ Andere bestanden zijn afhankelijk van de versie van de broncode-indeling
 - ▶ 1.0 of 3.0 (native): `pakket_versie.tar.gz`
 - ▶ 1.0 (non-native):
 - ▶ `pkt_ver.orig.tar.gz`: bovenstroomse broncode
 - ▶ `pkt_debver.diff.gz`: patch voor het toevoegen van Debian-specifieke aanpassingen
 - ▶ 3.0 (quilt):
 - ▶ `pkt_ver.orig.tar.gz`: bovenstroomse broncode
 - ▶ `pkt_debver.debian.tar.gz`: tar-archief met Debian-wijzigingen

(Zie `dpkg-source(1)` voor exacte details)



Voorbeeld broncodepakket (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
```

Een bestaand broncodepakket ophalen

- ▶ Uit het archief van Debian:
 - ▶ `apt-get source pakket`
 - ▶ `apt-get source pakket=versie`
 - ▶ `apt-get source pakket/release`(U moet deb-src-regels hebben in `sources.list`)
- ▶ Van het internet:
 - ▶ `dget url-naar.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` bevat alle Debian-pakketten sinds 2005)
- ▶ Van het (opgegeven) versiebeheersysteem:
 - ▶ `debcheckout pakket`
- ▶ Extraheer na het downloaden met `dpkg-source -x file.dsc`



Een basaal broncodepakket aanmaken

- ▶ Download de bovenstroomse broncode
(*bovenstroomse broncode* = die van de originele softwareontwikkelaars)
- ▶ Hernoem naar `<bron_pakket>_<bovenstroom_versie>.orig.tar.gz`
(bijvoorbeeld: `simgrid_3.6.orig.tar.gz`)
- ▶ Pak uit
- ▶ Hernoem de map naar `<bron_pakket>-<bovenstroom_versie>`
(voorbeeld: `simgrid-3.6`)
- ▶ `cd <bron_pakket>-<bovenstroom_versie> && dh_make`
(uit het pakket **dh-make**)
- ▶ Er bestaan enkele alternatieven voor `dh_make` voor specifieke groepen pakketten: **dh-make-perl**, **dh-make-php**, ...
- ▶ Er werd een map `debian/` aangemaakt met daarin een heleboel bestanden



Bestanden in debian/

Alle verpakkingswerk gebeurt door bestanden in `debian/` te wijzigen

- ▶ Hoofdbestanden:
 - ▶ **control** – meta-gegevens over het pakket (vereisten, enz.)
 - ▶ **rules** – geeft aan hoe het pakket gebouwd moet worden
 - ▶ **copyright** – auteursrechtinformatie voor het pakket
 - ▶ **changelog** – geschiedenis van het Debian-pakket
- ▶ Andere bestanden:
 - ▶ `compat`
 - ▶ `watch`
 - ▶ `dh_install* targets`
`*.dirs`, `*.docs`, `*.manpages`, ...
 - ▶ `onderhoudsscripts`
`*.postinst`, `*.prerm`, ...
 - ▶ `source/format`
 - ▶ `patches/` – indien u de bovenstroomse broncode moet aanpassen
- ▶ Verschillende bestanden hebben een op RFC 822 (mail-kopregels) gebaseerde indeling



debian/changelog

- ▶ Vermeldt de Debian verpakkingswijzigingen
- ▶ Vermeldt de huidige versie van het pakket

1.2.1.1-5

Upstream Debian
versie revisie

- ▶ Handmatig ingevoerd of bewerkt met **dch**
 - ▶ Een changelog-item aanmaken voor een nieuwe release: **dch -i**
- ▶ Speciale indeling om automatisch Debian of Ubuntu bugs te sluiten
Debian: Closes: #595268; Ubuntu: LP: #616929
- ▶ Geïnstalleerd als `/usr/share/doc/pakket/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

- ▶ Pakket-metagegevens
 - ▶ Voor het broncodepakket zelf
 - ▶ Voor elk binair pakket dat uit deze broncode gebouwd wordt
 - ▶ Pakketnaam, sectie, prioriteit, onderhouder, uploaders, bouw-vereisten, vereisten, beschrijving, homepage, ...
 - ▶ Documentatie: Debian beleidshandboek hoofdstuk 5
<https://www.debian.org/doc/debian-policy/ch-controlfields>
-

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

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



Architectuur: all of any

Twee soorten binaire pakketten:

- ▶ Pakketten met verschillende inhoud voor elke Debian-architectuur
 - ▶ Bijvoorbeeld: C-programma
 - ▶ Architecture: any in debian/control
 - ▶ Of, als het enkel werkt op een deelgroep architecturen:
Architecture: amd64 i386 ia64 hurd-i386
 - ▶ buildd.debian.org: bouwt alle andere architecturen voor u na het uploaden
 - ▶ Genaamd `pakket_versie_architectuur.deb`
- ▶ Pakketten met dezelfde inhoud op alle architecturen
 - ▶ Bijvoorbeeld: Perl-bibliotheek
 - ▶ Architecture: all in debian/control
 - ▶ Genaamd `pakket_versie_all.deb`

Uit een broncodepakket kan een mengeling van Architecture: any en Architecture: all binaire pakketten gegenereerd worden



debian/rules

- ▶ Make-bestand
- ▶ Interface die gebruikt wordt om Debian-pakketten te bouwen
- ▶ Beschreven in Debian beleidshandboek, hoofdstuk 4.8
<https://www.debian.org/doc/debian-policy/ch-source#s-debianrules>
- ▶ Vereiste doelen:
 - ▶ build, build-arch, build-indep: moet alle configuratie en compilatie uitvoeren
 - ▶ binary, binary-arch, binary-indep: binaire pakketten bouwen
 - ▶ dpkg-buildpackage aanroep binary om alle pakketten te bouwen of binary-arch om enkel de Architecture: any-pakketten te bouwen
 - ▶ clean: de broncodemap opruimen



Verpakkingshulpprogramma's – debhelper

- ▶ U kunt in `debian/rules` rechtstreeks shell-code schrijven
- ▶ Beter (voor de meeste pakketten): een *verpakkingshulpprogramma*
- ▶ Het populairste is: **debhelper** (voor 98% van de pakketten gebruikt)
- ▶ Doelstellingen:
 - ▶ Gewone taken opnemen in door alle pakketten gebruikt gereedschap
 - ▶ Sommige verpakkingsfouten in eenmaal oplossen voor alle pakketten

`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`, ...

- ▶ Wordt aangeroepen vanuit `debian/rules`
 - ▶ Configureerbaar met commandoparameters of bestanden in `debian/pakket.docs`, `pakket.examples`, `pakket.install`, `pakket.manpages`, ...
- ▶ Hulpprogramma's van derden voor bepaalde groepen pakketten: **python-support**, **dh_ocaml**, ...
- ▶ `debian/compat`: Debhelper-compatibiliteitsversie
 - ▶ Definieert het precieze gedrag van `dh_*`
 - ▶ Nieuwe syntaxis: `Build-Depends: debhelper-compat (= 13)`



debian/rules met debhelper (1/2)

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

# Verwijder commentaartekenen om breedsprakige modus aan te zetten.
#export DH_VERBOSE=1

build:
    $(MAKE)
    #docbook-naar-man debian/pakketnaam.sgml > pakketnaam.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
    # Hier commando's voor pakketinstallatie in debian/pakketnaam.
    $(MAKE) DESTDIR=$(CURDIR)/debian/pakketnaam install
```



debian/rules met debhelper (2/2)

```
# Hier architectuur-onafhankelijke bouwbestanden.  
binary-indep: build install
```

```
# Hier architectuur-afhankelijke bouwbestanden.  
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

- ▶ Met debhelper is er nog steeds veel redundantie tussen pakketten
- ▶ Hulpprogramma's van het tweede niveau met algemene functionaliteit
 - ▶ Bijv. bouwen met `./configure && make && make install` of CMake
- ▶ CDBS:
 - ▶ Geïntroduceerd in 2005, gebaseerd op geavanceerde *GNU make*-magie
 - ▶ Documentatie: `/usr/share/doc/cdb/`
 - ▶ Ondersteunt Perl, Python, Ruby, GNOME, KDE, Java, Haskell, ...
 - ▶ Maar sommige mensen haten het:
 - ▶ Soms moeilijk om het bouwen van pakketten aan te passen:
"kromme puinhoop make-bestanden en omgevingsvariabelen"
 - ▶ Trager dan debhelper zelf (veel nutteloos aanroepen van `dh_*`)

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

```
# een actie toevoegen na het bouwen
build/mijnpakket::
    /bin/bash debian/scripts/foo.sh
```



Dh (ook bekend als Debhelper 7 of dh7)

- ▶ Geïntroduceerd in 2008 als een *CDBS-doder*
- ▶ **dh** commando dat `dh_*` aanroept
- ▶ Eenvoudige *debian/rules*, enkel opsomming overrides (overschrijvingen)
- ▶ Makkelijker aan te passen dan CDBS
- ▶ Doc: man-pagina's (`debhelper(7)`, `dh(1)`) + dia's van DebConf9-presentatie
<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
```



Klassieke debhelper vs CDBS vs dh

- ▶ Aanhang:
Klassieke debhelper: 15% CDBS: 15% dh: 68%
- ▶ Welk moet ik leren?
 - ▶ Wellicht een beetje van allemaal
 - ▶ U moet debhelper kennen om dh en CDBS te gebruiken
 - ▶ Mogelijk moet u CDBS-pakketten aanpassen
- ▶ Welk moet ik gebruiken voor een nieuw pakket?
 - ▶ **dh** (enig hulpmiddel met een groeiende aanhang)
 - ▶ Zie <https://trends.debian.net/#build-systems>



Overzicht

- 1 Inleiding
- 2 Broncodepakketten maken
- 3 Pakketten bouwen en testen**
- 4 Praktijkoefening 1: het grep-pakket aanpassen
- 5 Geavanceerde verpakkingsonderwerpen
- 6 Pakketten onderhouden in Debian
- 7 Conclusies
- 8 Extra praktijkoefeningen
- 9 Oplossingen voor de praktijkoefeningen



Pakketten bouwen

- ▶ `apt-get build-dep mijnpakket`
Installeert *build-dependencies* (voor reeds in Debian aanwezig pakket)
Of `mk-build-deps -ir` (voor een pakket dat nog niet geüpload werd)
- ▶ `debuild`: bouwen, testen met `lintian`, ondertekenen met GPG
- ▶ Het is ook mogelijk om rechtstreeks `dpkg-buildpackage` aan te roepen
 - ▶ Gewoonlijk met `dpkg-buildpackage -us -uc`
- ▶ Het is best om pakketten te bouwen in een zuivere & minimale omgeving
 - ▶ `pbuilder` – hulpprogramma om een pakket te bouwen in een *chroot*
Goede documentatie: <https://wiki.ubuntu.com/PbuilderHowto>
(optimalisatie: `cowbuilder ccache distcc`)
 - ▶ `schroot` en `sbuid`: gebruikt op de Debian bouwachtergronddiensten
(minder eenvoudig dan `pbuilder`, wel LVM-momentopname mogelijk
zie: <https://help.ubuntu.com/community/SbuildLVMHowto>)
- ▶ Genereert `.deb`-bestanden en een `.changes`-bestand
 - ▶ `.changes`: beschrijft wat gebouwd werd; gebruikt om het pakket te uploaden



Pakketten installeren en testen

- ▶ Het pakket lokaal installeren: `debi` (zal gebruik maken van `.changes` om te weten wat geïnstalleerd moet worden)
- ▶ De inhoud van het pakket tonen: `debc` `../mijnpakket<TAB>.changes`
- ▶ Het pakket vergelijken met een eerdere versie:
`debdiff` `../mijnpakket_1_*.changes` `../mijnpakket_2_*.changes`
of om de broncode te vergelijken:
`debdiff` `../mijnpakket_1_*.dsc` `../mijnpakket_2_*.dsc`
- ▶ Het pakket controleren met `lintian` (statische analysator):
`lintian` `../mijnpakket<TAB>.changes`
`lintian -i`: geeft meer informatie over de fouten
`lintian -EviIL +pedantic`: toont meer problemen
- ▶ Het pakket naar Debian uploaden (`dput`) (daarvoor is configuratie nodig)
- ▶ Een persoonlijk Debian-archief beheren met `reprepro` of `aptly`
Documentatie:
<https://wiki.debian.org/HowToSetupADebianRepository>



Overzicht

- 1 Inleiding
- 2 Broncodepakketten maken
- 3 Pakketten bouwen en testen
- 4 Praktijkoefening 1: het grep-pakket aanpassen**
- 5 Geavanceerde verpakkingsonderwerpen
- 6 Pakketten onderhouden in Debian
- 7 Conclusies
- 8 Extra praktijkoefeningen
- 9 Oplossingen voor de praktijkoefeningen



Praktijkoefening 1: het grep-pakket aanpassen

- ❶ Ga naar `http://ftp.debian.org/debian/pool/main/g/grep/` en download versie 2.12-2 van het pakket
 - ▶ Indien het broncodepakket niet automatisch uitgepakt wordt, pak het dan uit met `dpkg-source -x grep_*.dsc`
- ❷ Bekijk de bestanden in `debian/`.
 - ▶ Hoeveel binaire pakketten worden uit dit bronpakket gegenereerd?
 - ▶ Welk verpakkingshulpprogramma wordt gebruikt voor dit pakket?
- ❸ Bouw het pakket
- ❹ Nu gaan we het pakket aanpassen. Voeg een changelog-item toe en verhoog het versienummer.
- ❺ Schakel ondersteuning voor perl-regexp uit (is een `./configure`-optie)
- ❻ Bouw het pakket nogmaals
- ❼ Vergelijk het originele en het nieuwe pakket met `debdiff`
- ❽ Installeer het nieuwgebouwde pakket



Overzicht

- 1 Inleiding
- 2 Broncodepakketten maken
- 3 Pakketten bouwen en testen
- 4 Praktijkoefening 1: het grep-pakket aanpassen
- 5 Geavanceerde verpakkingsonderwerpen**
- 6 Pakketten onderhouden in Debian
- 7 Conclusies
- 8 Extra praktijkoefeningen
- 9 Oplossingen voor de praktijkoefeningen



debian/copyright

- ▶ Copyright- en licentie-informatie in verband met broncode en verpakking
- ▶ Traditioneel geschreven als een tekstbestand
- ▶ Nieuwe machine-leesbare indeling:

<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+

Dit programma is vrije software; u mag het verspreiden
[...]
.
Op Debian-systemen is de volledige tekst van de GNU General
Public License versie 2 te vinden in het bestand
‘/usr/share/common-licenses/GPL-2’.
```

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



De bovenstroomse broncode aanpassen

Vaak nodig:

- ▶ Bugs repareren of Debian-specifieke aanpassingen toevoegen
- ▶ Reparaties uit een recentere bovenstroomse versie naar een eerdere versie overbrengen (backport)

Verschillende manieren om dit te doen:

- ▶ De bestanden rechtstreeks wijzigen
 - ▶ Eenvoudig
 - ▶ Maar geen mogelijkheid om de wijzigingen bij te houden en te documenteren
- ▶ Patch-systemen gebruiken
 - ▶ Maakt het makkelijker om uw aanpassingen terug te koppelen naar de bovenstroomse ontwikkelaar
 - ▶ Helpt het delen van de reparaties met afgeleide distributies
 - ▶ Geeft meer zicht op de veranderingen

<http://patch-tracker.debian.org/> (momenteel uitgezet)



Patch-systemen

- ▶ Principe: wijzigingen worden als patches opgeslagen in `debian/patches/`
- ▶ Toegepast of niet toegepast tijdens het bouwen
- ▶ Vroeger: meerdere toepassingen – *simple-patchsys* (*cdb*s), *dpatch*, ***quilt***
 - ▶ Elk van hen ondersteunt twee `debian/rules`-targets:
 - ▶ `debian/rules patch`: alle patches toepassen
 - ▶ `debian/rules unpatch`: geen patches toepassen
 - ▶ Meer documentatie: <https://wiki.debian.org/debian/patches>
- ▶ **Nieuwe bronpakketindeling met ingebouwd patch-systeem: 3.0 (quilt)**
 - ▶ Aanbevolen werkwijze
 - ▶ U moet *quilt* leren
<https://perl-team.pages.debian.net/howto/quilt.html>
 - ▶ Patch-systeem-agnostisch hulpmiddel in `devscripts`: `edit-patch`



Documentatie van patches

- ▶ Standaard kopregels bij het begin van de patch
- ▶ Gedocumenteerd in DEP-3 - Richtlijnen voor het merken van patches
<http://dep.debian.net/deps/dep3/>

```
Description: Snelheid van morrelen met widgets repareren
Te snel morrelen met widgets gaf explosiegevaar.
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 {
```



Dingen doen tijdens installatie en verwijdering

- ▶ Het pakket uitpakken volstaat soms niet
- ▶ Systeemgebruikers maken/verwijderen, diensten starten/stoppen, *alternatives* beheren
- ▶ Gebeurt in *maintainerscripts*
preinst, postinst, prerm, postrm
 - ▶ Met debhelper kan men fragmenten voor algemene acties genereren
- ▶ Documentatie:
 - ▶ Debian Beleidshandboek, hoofdstuk 6
<https://www.debian.org/doc/debian-policy/ch-maintainerscripts>
 - ▶ Debian Referentiehandleiding voor ontwikkelaars, hoofdstuk 6.4
<https://www.debian.org/doc/developers-reference/best-pkging-practices.html>
 - ▶ <https://people.debian.org/~srivasta/MaintainerScripts.html>
- ▶ De gebruiker een vraag stellen
 - ▶ Moet met **debconf** gebeuren
 - ▶ Documentatie: `debconf-devel(7)` (pakket `debconf-doc`)



Bovenstroomse versies volgen

- ▶ Waar gekeken moet worden, opgeven in `debian/watch` (zie `uscan(1)`)

```
version=3
```

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

- ▶ Er zijn geautomatiseerde volgers van nieuwe bovenstroomse versies, welke de onderhouder via verschillende instrumentenborden informeren, zoals <https://tracker.debian.org/> en <https://udd.debian.org/dmd/>
- ▶ `uscan`: een handmatige controle uitvoeren
- ▶ `uupdate`: uw pakket proberen opwaarderen naar de laatste bovenstroomse versie



Verpakken met een versiecontrolesysteem

- ▶ Diverse middelen beheren aftakkingen en tags voor uw verpakkingswerk: `svn-buildpackage`, `git-buildpackage`
- ▶ Bijvoorbeeld: `git-buildpackage`
 - ▶ `upstream`-tak om bovenstroom te volgen met `upstream/versie`-tags
 - ▶ `master`-tak volgt het Debian-pakket
 - ▶ `debian/versie`-tags voor elke upload
 - ▶ `pristine-tar`-tak om bovenstroom-tarball te kunnen herbouwen

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

- ▶ `Vcs-*`-velden in `debian/control` om het archief te vinden
 - ▶ <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>

- ▶ VCS-agnostische interface: `debcheckout`, `debcommit`, `debrelease`
 - ▶ `debcheckout` `grep` → haalt het broncodepakket op uit Git



Pakketten geschikt maken voor een eerdere release

- ▶ Bedoeling: een recentere versie van een pakket op een ouder systeem gebruiken
bijv. *mutt* uit Debian *unstable* gebruiken op Debian *stable*
- ▶ Algemeen idee:
 - ▶ Het broncodepakket uit Debian unstable nemen
 - ▶ Het aanpassen zodat het gebouwd kan worden en goed werkt op Debian stable
 - ▶ Soms onbeduidend (geen aanpassingen nodig)
 - ▶ Soms moeilijk
 - ▶ Soms onmogelijk (veel niet-beschikbare vereisten)
- ▶ Sommige van deze zogenaamde backports worden door het Debian-project ondersteund en ter beschikking gesteld
<http://backports.debian.org/>

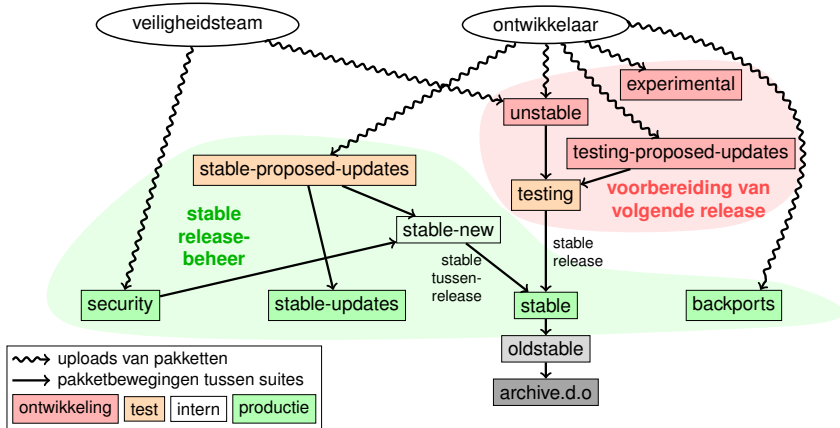


Overzicht

- 1 Inleiding
- 2 Broncodepakketten maken
- 3 Pakketten bouwen en testen
- 4 Praktijkoefening 1: het grep-pakket aanpassen
- 5 Geavanceerde verpakkingsonderwerpen
- 6 Pakketten onderhouden in Debian**
- 7 Conclusies
- 8 Extra praktijkoefeningen
- 9 Oplossingen voor de praktijkoefeningen



Debian-archief en suites



Gebaseerd op een diagram van Antoine Beaupré. <https://salsa.debian.org/debian/package-cycle>



Ontwikkelingssuites

- ▶ Nieuwe pakketversies worden geüpload naar **unstable** (**sid**)
- ▶ Pakketten verschuiven van **unstable** naar **testing** op basis van verschillende criteria (bijv. was gedurende 10 dagen in unstable zonder regressies)
- ▶ Nieuwe pakketten kunnen ook geüpload worden naar:
 - ▶ **experimental** (voor meer *experimentele* pakketten, zoals wanneer de nieuwe versie niet klaar is om die welke nu in unstable zit, te vervangen)
 - ▶ **testing-proposed-updates**, om de versie in **testing** te updaten zonder via **unstable** te gaan (dit wordt zelden gebruikt)



Bevriezen en uitbrengen

- ▶ Op een gegeven moment in de release-cyclus neemt het release-team de beslissing om testing *te bevriezen*: automatische verschuivingen van **unstable** naar **testing** worden gestopt en vervangen door een handmatig nazicht
- ▶ Wanneer het release-team **testing** klaar voor release beschouwt:
 - ▶ De suite **testing** wordt de nieuwe suite **stable**
 - ▶ Op dezelfde wijze wordt het oude **stable** **oldstable**
 - ▶ Niet-ondersteunde releases worden verplaatst naar `archive.debian.org`
- ▶ Zie <https://release.debian.org/>



Suites en beheer van de stabiele release (stable)

- ▶ Verschillende suites stellen pakketten uit stable ter beschikking:
 - ▶ **stable**: de hoofdsuite
 - ▶ **security** suite met updates welke ter beschikking gesteld wordt op `security.debian.org`, gebruikt door het veiligheidsteam. Updates worden aangekondigd op de mailinglijst `debian-security-announce`
 - ▶ **stable-updates**: geen veiligheidsupdates, maar die toch met spoed geïnstalleerd moeten worden (zonder te wachten op de volgende tussenrelease): antivirusdatabases, tijdzonegerelateerde pakketten, enz. Aangekondigd op de mailinglijst `debian-stable-announce`
 - ▶ **backports**: nieuwe upstream-versie, gebaseerd op de **testing**-versie
- ▶ De suite **stable** wordt om de enkele maanden geüpdatet via *tussenreleases van stable* (dit betreft enkel bugreparaties)
 - ▶ Pakketten voor de volgende stable-tussenrelease worden geüpload naar **stable-proposed-updates** en nagezien door het releaseteam
- ▶ De release **oldstable** heeft dezelfde reeks suites



Verschillende wijzen van bijdragen aan Debian

► **Slechtste** manier om bij te dragen:

- 1 Uw eigen toepassing verpakken
- 2 Deze in Debian binnenbrengen
- 3 Verdwijnen

► **Bettere** manieren om bij te dragen:

- Meewerken met verpakkingsteams
 - Veel teams focussen op een reeks pakketten en zoeken hulp
 - Lijst te vinden op <https://wiki.debian.org/Teams>
 - Excellente manier om te leren van meer ervaren medewerkers
- Bestaande niet-onderhouden pakketten adopteren (*verweesde pakketten*)
- Nieuwe software binnenbrengen in Debian
 - Liefst enkel wanneer deze voldoende interessant/bruikbaar is
 - Bestaan er reeds voor Debian verpakte alternatieven?



Verweesde pakketten adopteren

- ▶ Veel niet-onderhouden pakketten in Debian
- ▶ Volledige lijst + proces: <https://www.debian.org/devel/wnpp/>
- ▶ Geïnstalleerd op uw computer: `wnpp-alert`
Of beter: `how-can-i-help`
- ▶ Verschillende toestanden:
 - ▶ **Orphaned** (Verweesd): het pakket wordt niet onderhouden
U mag het gerust adopteren
 - ▶ **RFA: Request For Adopter** (Adoptant gevraagd)
Onderhouder zoekt een adoptant, maar blijft intussen verder werken
Adopteer het gerust. Een e-mail naar de huidige onderhouder is beleefd
 - ▶ **ITA: Intent To Adopt** (Adoptie-intentie)
Iemand is zinnens het pakket te adopteren. Bied uw hulp aan!
 - ▶ **RFH: Request For Help** (Hulp gevraagd)
De onderhouder is op zoek naar hulp
- ▶ Niet-gedetectedeerde niet-onderhouden pakketten → nog niet verweesd
- ▶ Bij twijfel navragen op `debian-qa@lists.debian.org`
of `#debian-qa` op `irc.debian.org`



Een pakket adopteren: een voorbeeld

```
From: Jij <jij@jouwdomein>  
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,

Jij

- ▶ Het is beleefd om de vroegere onderhouder te contacteren (in het bijzonder wanneer het een RFA betreft en het pakket niet verweesd werd)
- ▶ Zeer goede gewoonte om het bovenstrooms project te contacteren



Uw pakket in Debian krijgen

- ▶ Geen enkele officiële status nodig om uw pakket in Debian te krijgen
 - ① Dien een **ITP**-bugrapport (**I**ntent **T**o **P**ackage) (verpakkingsintentie) in met `reportbug wnpp`
 - ② Maak uw broncodepakket klaar
 - ③ Zoek een ontwikkelaar van Debian die uw pakket zal sponsoren
- ▶ Officiële status (wanneer u een ervaren pakketonderhouder bent):
 - ▶ **Debian Maintainer (DM) (onderhouder van Debian):**
Toelating om uw eigen pakketten te uploaden
Zie <https://wiki.debian.org/DebianMaintainer>
 - ▶ **Debian Developer (DD) (ontwikkelaar van Debian):**
Lid van het Debian-project; kan stemmen en elk pakket uploaden



Te controleren zaken voor u sponsoring vraagt

- ▶ Debian besteedt **veel aandacht aan kwaliteit**
- ▶ Algemeen **zijn sponsors moeilijk te vinden en hebben het druk**
 - ▶ Zorg ervoor dat uw pakket klaar is voor u om sponsoring vraagt
- ▶ Te controleren zaken:
 - ▶ Vermijd ontbrekende build-dependencies: zorg ervoor dat uw pakket probleemloos kan gebouwd worden in een zuivere *sid chroot*
 - ▶ Het gebruik van `pbuilder` is aanbevolen
 - ▶ Voer het commando `lintian -EviIL +pedantic` uit op uw pakket
 - ▶ Fouten moeten hersteld worden, alle ander problemen zouden opgelost moeten worden
 - ▶ Test uiteraard uitgebreid uw pakket
- ▶ Vraag bij twijfel hulp



Waar vindt u hulp?

Hulp die u nodig zult hebben:

- ▶ Advies en antwoorden op uw vragen, nazicht van code
- ▶ Sponsoring voor uw uploads wanneer uw pakket klaar is

U kunt hulp krijgen bij:

- ▶ **Andere leden van een verpakkingsteam**
 - ▶ Lijst van teams: <https://wiki.debian.org/Teams>
- ▶ De groep **Debian Mentors** (indien uw pakket niet bij een team past)
 - ▶ <https://wiki.debian.org/DebianMentorsFaq>
 - ▶ Mailinglijst: debian-mentors@lists.debian.org
(ook een goede manier om al doende te leren)
 - ▶ IRC: #debian-mentors op irc.debian.org
 - ▶ <http://mentors.debian.net/>
 - ▶ Documentatie: <http://mentors.debian.net/intro-maintainers>
- ▶ **Gelocaliseerde mailinglijsten** (hulp krijgen in uw eigen taal)
 - ▶ debian-devel-{french,italian,portuguese,spanish}@lists.d.o
 - ▶ Basislijst: <https://lists.debian.org/devel.html>
 - ▶ Of gebruikerslijsten: <https://lists.debian.org/users.html>



Meer documentatie

- ▶ Ontwikkelaarshoek van Debian
<https://www.debian.org/devel/>
Links naar veel informatiebronnen over de ontwikkeling van Debian
- ▶ Gids voor onderhouders van Debian
<https://www.debian.org/doc/manuals/debmake-doc/>
- ▶ Referentiehandleiding voor ontwikkelaars van Debian
<https://www.debian.org/doc/developers-reference/>
Hoofdzakelijk over werkwijzen van Debian, maar ook sommige goede verpakkingsmethodes (deel 6)
- ▶ Debian beleidshandboek
<https://www.debian.org/doc/debian-policy/>
 - ▶ Alle vereisten waaraan een pakket moet voldoen
 - ▶ Specifieke beleidsrichtlijnen voor Perl, Java, Python, ...
- ▶ Verpakkingshandleiding voor Ubuntu
<https://packaging.ubuntu.com/html/>



Instrumentenborden voor Debian-onderhouders

- ▶ **Toegespitst op broncodepakket:**
<https://tracker.debian.org/dpkg>
- ▶ **Toegespitst op onderhouder/team:** Developer's Packages Overview (DDPO - Pakketoverzicht voor ontwikkelaars)
<https://qa.debian.org/developer.php?login=pkg-ruby-extras-maintainers@lists.alioth.debian.org>
- ▶ **TODO-lijst georiënteerd:** Debian Maintainer Dashboard (DMD)
<https://udd.debian.org/dmd/>



Het Debian Bug Tracking System (BTS) gebruiken

- ▶ Bugvolgsysteem - Een vrij unieke manier voor bugbeheer
 - ▶ Webinterface voor het bekijken van bugs
 - ▶ E-mailinterface om veranderingen aan te brengen aan bugs
- ▶ Information toevoegen aan bugs:
 - ▶ Schrijf naar `123456@bugs.debian.org` (sluit de indiener niet in, voeg daarvoor `123456-submitter@bugs.debian.org` toe)
- ▶ De toestand van een bug wijzigen:
 - ▶ Stuur commando's naar `control@bugs.debian.org`
 - ▶ Commandoregelinterface: `bts-commando` in `devscripts`
 - ▶ Documentatie: <https://www.debian.org/Bugs/server-control>
- ▶ Bugs rapporteren: gebruik `reportbug`
 - ▶ Gewoonlijk gebruikt met een lokale mail-server: installeer `ssmtp` of `nullmailer`
 - ▶ Of gebruik `reportbug --template` en stuur dan (handmatig) naar `submit@bugs.debian.org`



Het BTS gebruiken: voorbeelden

- ▶ Een e-mail sturen naar de bug en de indiener:
`https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680822#10`
- ▶ Taggen en de ernst wijzigen:
`https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680227#10`
- ▶ Opnieuw toewijzen, de ernst wijzigen, een nieuwe titel geven ...:
`https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680822#93`
 - ▶ notfound, found, notfixed, fixed zijn voor **versie-opvolging**
Zie `https://wiki.debian.org/HowtoUseBTS#Version\_tracking`
- ▶ Usertags (gebruikerslabels) gebruiken: `https://bugs.debian.org/cgi-bin/bugreport.cgi?msg=42;bug=642267`
Zie `https://wiki.debian.org/bugs.debian.org/usertags`
- ▶ BTS-documentatie:
 - ▶ `https://www.debian.org/Bugs/`
 - ▶ `https://wiki.debian.org/HowtoUseBTS`



Meer geïnteresseerd in Ubuntu?

- ▶ Ubuntu beheert vooral de divergentie met Debian
- ▶ Geen echte focus op specifieke pakketten
Eerder samenwerking met Debian-teams
- ▶ Beveelt meestal aan nieuwe pakketten eerst naar Debian te uploaden
<https://wiki.ubuntu.com/UbuntuDevelopment/NewPackages>
- ▶ Mogelijk een betere werkwijze:
 - ▶ U engageren in een Debian-team en fungeren als brug met Ubuntu
 - ▶ De divergentie helpen verkleinen, triage van bugs in Launchpad
 - ▶ Verschillende hulpmiddelen van Debian kunnen behulpzaam zijn:
 - ▶ De Ubuntu-kolom op het pakketoverzicht voor ontwikkelaars
 - ▶ Ubuntu-vakje in het pakketvolgsysteem
 - ▶ Bug-e-mails uit launchpad ontvangen via het PTS



Overzicht

- 1 Inleiding
- 2 Broncodepakketten maken
- 3 Pakketten bouwen en testen
- 4 Praktijkoefening 1: het grep-pakket aanpassen
- 5 Geavanceerde verpakkingsonderwerpen
- 6 Pakketten onderhouden in Debian
- 7 Conclusies**
- 8 Extra praktijkoefeningen
- 9 Oplossingen voor de praktijkoefeningen



Conclusies

- ▶ U heeft nu een volledig overzicht over verpakken voor Debian
- ▶ Maar u zult meer documentatie moeten lezen
- ▶ Goede praktijken evolueerden in de loop der jaren
 - ▶ Gebruik bij twijfel het verpakkingshulpprogramma **dh** en de indeling **3.0 (quilt)**

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



Juridische zaken

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

Dit document is vrije software: u kunt het verder verspreiden en/of wijzigen, onder (volgens uw voorkeur):

- ▶ De bepalingen van de GNU General Public License, zoals gepubliceerd door de Free Software Foundation, versie 3 of (volgens uw goeddunken) een recentere versie.

<http://www.gnu.org/licenses/gpl.html>

- ▶ De bepalingen van de Creative Commons Attribution-ShareAlike 3.0 Unported License.

<http://creativecommons.org/licenses/by-sa/3.0/>



Meewerken aan deze handleiding

▶ Meewerken:

- ▶ `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`
- ▶ Open bugs: `bugs.debian.org/src:packaging-tutorial`

▶ Feedback geven:

- ▶ `mailto:packaging-tutorial@packages.debian.org`
 - ▶ Wat zou nog toegevoegd moeten worden aan deze handleiding?
 - ▶ Wat zou verbeterd moeten worden?
- ▶ `reportbug packaging-tutorial`



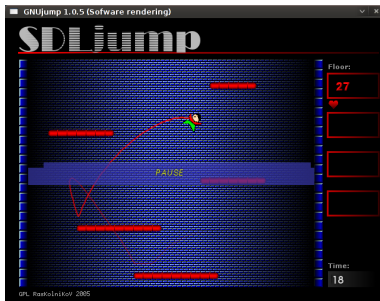
Overzicht

- 1 Inleiding
- 2 Broncodepakketten maken
- 3 Pakketten bouwen en testen
- 4 Praktijkoefening 1: het grep-pakket aanpassen
- 5 Geavanceerde verpakkingsonderwerpen
- 6 Pakketten onderhouden in Debian
- 7 Conclusies
- 8 Extra praktijkoefeningen**
- 9 Oplossingen voor de praktijkoefeningen



Praktijkoefening 2: GNUjump verpakken

- 1 Download GNUjump 1.0.8 van
<http://ftp.gnu.org/gnu/gnujump/gnujump-1.0.8.tar.gz>
- 2 Creëer er een Debian-pakket voor
 - ▶ De build-dependencies installeren, zodat u het pakket kunt bouwen
 - ▶ Bugs repareren
 - ▶ Een basaal werkend pakket bekomen
 - ▶ Afronden met het invullen van `debian/control` en andere bestanden
- 3 Veel plezier



Praktijkoefening 2: GNUjump verpakken (tips)

- ▶ Gebruik `dh_make` om een basaal werkend pakket te bekomen
- ▶ Om te beginnen is het creëren van een *1.0* broncodepakket makkelijker dan *3.0* (*quilt*) (pas dat aan in `debian/source/format`)
- ▶ Ontbrekende bouwvereisten (build-dependencies) vinden: zoek welk bestand ontbreekt en gebruik `apt-file` om ontbrekende pakket te vinden
- ▶ Indien u op de volgende foutmelding stoot:

```
/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
```

moet u `-lm` toevoegen aan de commandoregel voor de linker:

Bewerk `src/Makefile.am` en vervang

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

door

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

En geef daarna het commando `autoreconf -i`



Praktijkoefening 3: een Java-bibliotheek verpakken

- ❶ Bekijk snel enige documentatie over het verpakken van 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](#)
- ❷ Download IRClib van <http://moepii.sourceforge.net/>
- ❸ Verpak het



Praktijkoefening 4: een Ruby gem verpakken

- 1 Bekijk snel enige documentatie over het verpakken van Ruby:
 - ▶ <https://wiki.debian.org/Ruby>
 - ▶ <https://wiki.debian.org/Teams/Ruby>
 - ▶ <https://wiki.debian.org/Teams/Ruby/Packaging>
 - ▶ `gem2deb(1)`, `dh_ruby(1)` (in het pakket `gem2deb`)
- 2 Maak een basaal Debian-broncodepakket van de `peach`-gem:
`gem2deb peach`
- 3 Verbeter het, zodat het een volwaardig Debian-pakket wordt



Praktijkoefening 5: een Perl-module verpakken

- 1 Bekijk snel enige documentatie over het verpakken van Perl:
 - ▶ <https://perl-team.pages.debian.net>
 - ▶ <https://wiki.debian.org/Teams/DebianPerlGroup>
 - ▶ `dh-make-perl(1)`, `dpt(1)` (in het pakket `pkg-perl-tools`)
- 2 Maak een basaal Debian-broncodepakket van de `Acme` CPAN-distributie:
`dh-make-perl --cpan Acme`
- 3 Verbeter het, zodat het een volwaardig Debian-pakket wordt



Overzicht

- 1 Inleiding
- 2 Broncodepakketten maken
- 3 Pakketten bouwen en testen
- 4 Praktijkoefening 1: het grep-pakket aanpassen
- 5 Geavanceerde verpakkingsonderwerpen
- 6 Pakketten onderhouden in Debian
- 7 Conclusies
- 8 Extra praktijkoefeningen
- 9 Oplossingen voor de praktijkoefeningen



Oplossingen voor de praktijkoefeningen



Praktijkoefening 1: het grep-pakket aanpassen

- ➊ Ga naar `http://ftp.debian.org/debian/pool/main/g/grep/` en download versie 2.12-2 van het pakket
- ➋ Bekijk de bestanden in `debian/`.
 - ▶ Hoeveel binaire pakketten worden uit dit bronpakket gegenereerd?
 - ▶ Welk verpakkingshulpprogramma wordt gebruikt voor dit pakket?
- ➌ Bouw het pakket
- ➍ Nu gaan we het pakket aanpassen. Voeg een changelog-item toe en verhoog het versienummer.
- ➎ Schakel ondersteuning voor perl-regexp uit (is een `./configure`-optie)
- ➏ Bouw het pakket nogmaals
- ➐ Vergelijk het originele en het nieuwe pakket met `debdiff`
- ➑ Installeer het nieuwgebouwde pakket



De broncode ophalen

- ❶ Ga naar `http://ftp.debian.org/debian/pool/main/g/grep/` en download versie 2.12-2 van het pakket
- ▶ Gebruik `dget` om het `.dsc`-bestand te downloaden:
`dget http://cdn.debian.net/debian/pool/main/g/grep/grep_2.12-2.dsc`
- ▶ Indien u een `deb-src`-regel heeft voor een release van Debian met daarin `grep` versie 2.12-2 (zoek dit uit op `https://tracker.debian.org/grep`), kunt u `apt-get source grep=2.12-2` gebruiken of `apt-get source grep/release` (bijv. `grep/stable`) of, indien u geluk heeft: `apt-get source grep`
- ▶ Het broncodepakket `grep` bestaat uit drie bestanden:
 - ▶ `grep_2.12-2.dsc`
 - ▶ `grep_2.12-2.debian.tar.bz2`
 - ▶ `grep_2.12.orig.tar.bz2`Dit is typerend voor de indeling "3.0 (quilt)".
- ▶ Pak indien nodig de broncode uit met `dpkg-source -x grep_2.12-2.dsc`



Rondkijken en het pakket bouwen

2 Bekijk de bestanden in `debian/`

- ▶ Hoeveel binaire pakketten worden uit dit bronpakket gegenereerd?
- ▶ Welk verpakkingshulpprogramma wordt gebruikt voor dit pakket?
- ▶ Volgens `debian/control` wordt uit dit pakket slechts één binair pakket gegenereerd, genaamd `grep`.
- ▶ Volgens `debian/rules` is dit pakket typerend voor een *klassiek* verpakken met `debhelper` zonder *CDBS* of *dh* te gebruiken. Men kan de verschillende aanroepen van `dh_*`-commando's zien in `debian/rules`.

3 Bouw het pakket

- ▶ Gebruik `apt-get build-dep grep` om de build-dependencies (bouwvereisten) op te halen
- ▶ Daarna `debuild` of `dpkg-buildpackage -us -uc` (Neemt ongeveer 1 min in beslag)



Het changelog-bestand bewerken

- 4 Nu gaan we het pakket aanpassen. Voeg een changelog-item toe en verhoog het versienummer.
- ▶ `debian/changelog` is a tekstbestand. U kunt het handmatig bewerken om er een nieuw item aan toe te voegen.
- ▶ Of u kunt `dch -i` gebruiken, wat een item zal toevoegen en de editor zal openen
- ▶ De naam en e-mail kunnen gedefinieerd worden met de omgevingsvariabelen `DEBFULLNAME` en `DEBEMAIL`
- ▶ Bouw nadien het pakket opnieuw: een nieuwe versie van het pakket wordt gebouwd
- ▶ Het versiebeheer van pakketten wordt uitgelegd in sectie 5.6.12 van het beleidshandboek van Debian
<https://www.debian.org/doc/debian-policy/ch-controlfields>



Perl-regexp-ondersteuning uitzetten en herbouwen

- ⑤ Schakel ondersteuning voor perl-regexp uit (is een `./configure`-optie)
- ⑥ Bouw het pakket nogmaals
 - ▶ Controleer met `./configure --help`: de optie om Perl regexp uit te zetten is `--disable-perl-regexp`
 - ▶ Bewerk `debian/rules` en zoek de regel `./configure`
 - ▶ Voeg `--disable-perl-regexp` toe
 - ▶ Bouw opnieuw met `debuild` of `dpkg-buildpackage -us -uc`



De pakketten vergelijken en testen

- 7 Vergelijk het originele en het nieuwe pakket met `debdiff`
- 8 Installeer het nieuwgebouwde pakket
 - ▶ Vergelijk de binaire pakketten: `debdiff ../changes`
 - ▶ Vergelijk de broncodepakketten: `debdiff ../dsc`
 - ▶ Installeer het net gebouwde pakket: `debi`
Of `dpkg -i ../grep_<TAB>`
 - ▶ `grep -P foo` werkt niet langer!

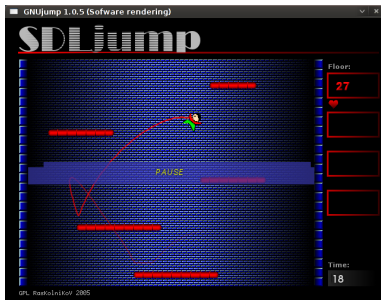
Herinstalleer de vroegere versie van het pakket:

- ▶ `apt-get install --reinstall grep=2.6.3-3` (= *vroegere versie*)



Praktijkoefening 2: GNUjump verpakken

- 1 Download GNUjump 1.0.8 van
<http://ftp.gnu.org/gnu/gnujump/gnujump-1.0.8.tar.gz>
- 2 Creëer er een Debian-pakket voor
 - ▶ De build-dependencies installeren, zodat u het pakket kunt bouwen
 - ▶ Een basaal werkend pakket bekomen
 - ▶ Afronden met het invullen van `debian/control` en andere bestanden
- 3 Veel plezier



Stap voor stap...

- ▶ `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 pakket: enkel binair (momenteel)

```
gnujump-1.0.8$ ls debian/
```

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



Stap voor stap... (2)

- ▶ Kijk naar `debian/changelog`, `debian/rules`, `debian/control` (automatisch ingevuld door **dh_make**)
- ▶ In `debian/control`:
Build-Depends: `debhelper (>= 7.0.50 )`, `autotools-dev`
Vermeldt de *build-dependencies* = pakketten, nodig om pakket te bouwen
- ▶ Probeer het pakket ongewijzigd te bouwen met `debuild` (dankzij **dh**-magie)
 - ▶ En voeg build-dependencies toe tot het gebouwd kan worden
 - ▶ Suggestie: gebruik `apt-cache search` en `apt-file` om de pakketten te zoeken
 - ▶ Voorbeeld:

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

→ Voeg **libsdl1.2-dev** toe aan Build-Depends en installeer het.
 - ▶ Beter: gebruik **pbuilder** om in een zuivere omgeving te bouwen



Stap voor stap... (3)

- ▶ Vereiste build-dependencies zijn libSDL1.2-dev, libSDL-image1.2-dev, libSDL-mixer1.2-dev
- ▶ Dan zult u wellicht tegen een andere foutmelding aanlopen:

```
/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
```

- ▶ Dit probleem wordt veroorzaakt door bitrot: gnujump werd niet aangepast na linker-wijzigingen.
- ▶ Indien u indelingsversie **1.0** voor broncode gebruikt, kunt u de bovenstroomse broncode rechtstreeks aanpassen.
 - ▶ Bewerk src/Makefile.am en vervang
`gnujump_LDFLAGS = $(all_libraries)`
door
`gnujump_LDFLAGS = -Wl,--as-needed`
`gnujump_LDADD = $(all_libraries) -lm`
 - ▶ En geef daarna het commando `autoreconf -i`



Stap voor stap... (4)

- ▶ Als u broncode-indelingsversie **3.0 (quilt)** gebruikt, gebruik dan quilt om een patch klaar te maken. (zie <https://wiki.debian.org/UsingQuilt>)
 - ▶ `export QUILT_PATCHES=debian/patches`
 - ▶ `mkdir debian/patches`
`quilt new linker-fixes.patch`
`quilt add src/Makefile.am`
 - ▶ Bewerk `src/Makefile.am` en vervang
`gnujump_LDFLAGS = $(all_libraries)`
door
`gnujump_LDFLAGS = -Wl,--as-needed`
`gnujump_LDADD = $(all_libraries) -lm`
 - ▶ `quilt refresh`
 - ▶ Omdat `src/Makefile.am` gewijzigd werd, is aanroep van autoreconf bij bouwen nodig. Dat automatisch doen met `dh` kan door `dh`-aanroep in `debian/rules` te wijzigen van: `dh $ --with autotools-dev`
naar: `dh $ --with autotools-dev --with autoreconf`



Stap voor stap... (5)

- ▶ Nu zou het pakket probleemloos gebouwd moeten kunnen worden.
- ▶ Gebruik `debnc` om de inhoud van het gegenereerde pakket op te lijsten en `debi` om het te installeren en te testen.
- ▶ Test het pakket met `lintian`
 - ▶ Hoewel het geen strikte vereiste is, is het aanbevolen dat pakketten die naar Debian geüpload worden, *lintian-clean* zijn
 - ▶ Meer problemen zijn te vinden met `lintian -EviIL +pedantic`
 - ▶ Enkele hints:
 - ▶ Verwijder de bestanden die u niet nodig heeft, uit `debian/`
 - ▶ Vul `debian/control` in
 - ▶ Installeer het uitvoerbaar programma in `/usr/games` door `dh_auto_configure` te overschrijven
 - ▶ Gebruik *hardening*-compilatievlaggen voor verhoogde veiligheid. Zie <https://wiki.debian.org/Hardening>



Stap voor stap... (6)

- ▶ Vergelijk uw pakket met het pakket dat reeds voor Debian verpakt is:
 - ▶ Dit splits de gegevensbestanden af in een tweede pakket dat identiek is voor alle architecturen (→ bespaart ruimte in het Debian-archief)
 - ▶ Het installeert een .desktop-bestand (voor het menu van GNOME/KDE) en geeft het ook een plaats in het Debian-menu
 - ▶ Het repareert enkele kleinere problemen met patches



Praktijkoefening 3: een Java-bibliotheek verpakken

- ❶ Bekijk snel enige documentatie over het verpakken van 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](#)
- ❷ Download IRClib van <http://moepii.sourceforge.net/>
- ❸ Verpak het



Stap voor stap...

- ▶ `apt-get install javahelper`
- ▶ Creëer een basaal broncodepakket: `jh_makepkg`
 - ▶ Bibliotheek
 - ▶ Geen
 - ▶ Standaard vrije compiler/runtime
- ▶ Kijk naar en repareer `debian/*`
- ▶ `dpkg-buildpackage -us -uc` Of `debuild`
- ▶ `lintian`, `debc`, enz.
- ▶ Vergelijk uw resultaat met het broncodepakket `libirclib-java`



Praktijkoefening 4: een Ruby gem verpakken

- 1 Bekijk snel enige documentatie over het verpakken van Ruby:
 - ▶ <https://wiki.debian.org/Ruby>
 - ▶ <https://wiki.debian.org/Teams/Ruby>
 - ▶ <https://wiki.debian.org/Teams/Ruby/Packaging>
 - ▶ `gem2deb(1)`, `dh_ruby(1)` (in het pakket `gem2deb`)
- 2 Maak een basaal Debian-broncodepakket van de `peach`-gem:
`gem2deb peach`
- 3 Verbeter het, zodat het een volwaardig Debian-pakket wordt



Stap voor stap...

`gem2deb peach:`

- ▶ Downloadt het gem van `rubygems.org`
- ▶ Creëert een geschikt `.orig.tar.gz`-archief en pakt het uit
- ▶ Initialiseert een Debian broncodepakket op basis van de gem-metadata
 - ▶ Genaamd `ruby-gemnaam`
- ▶ Tracht het Debian binaire pakket te bouwen (dit kan mislukken)

`dh_ruby` (zit in `gem2deb`) en voert de Ruby-specifieke taken uit:

- ▶ C-uitbreidingen bouwen voor elke Ruby-versie
- ▶ Bestanden naar hun doelmap kopiëren
- ▶ shebangs in uitvoerbare scripts updaten
- ▶ Testen uitvoeren die gedefinieerd worden in `debian/ruby-tests.rb`, `debian/ruby-tests.rake`, of `debian/ruby-test-files.yaml`, evenals verschillende andere controles



Stap voor stap... (2)

Verbeter het gegenereerde pakket:

- ▶ Voer `debclean` uit om de broncodeboom op te ruimen. Kijk naar `debian/`.
- ▶ `changelog` en `compat` zouden correct moeten zijn
- ▶ Bewerk `debian/control`: verbeter `Description`
- ▶ Schrijf een passend `copyright`-bestand op basis van de bovenstroomse bestanden
- ▶ Bouw het pakket
- ▶ Vergelijk uw pakket met het pakket `ruby-peach` uit het Debian-archief



Praktijkoefening 5: een Perl-module verpakken

- 1 Bekijk snel enige documentatie over het verpakken van Perl:
 - ▶ <https://perl-team.pages.debian.net>
 - ▶ <https://wiki.debian.org/Teams/DebianPerlGroup>
 - ▶ `dh-make-perl(1)`, `dpt(1)` (in het pakket `pkg-perl-tools`)
- 2 Maak een basaal Debian-broncodepakket van de `Acme` CPAN-distributie:
`dh-make-perl --cpan Acme`
- 3 Verbeter het, zodat het een volwaardig Debian-pakket wordt



Stap voor stap...

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

- ▶ Downloadt het tar-archief uit het CPAN
- ▶ Creëert een passend `.orig.tar.gz`-archief en pakt het uit
- ▶ Initialiseert een Debian broncodepakket, gebaseerd op de metagegevens van de distributie
 - ▶ Genaamd `libdistnaam-perl`



Stap voor stap... (2)

Verbeter het gegenereerde pakket:

- ▶ `debian/changelog`, `debian/compat`, `debian/libacme-perl.docs` en `debian/watch` zouden correct moeten zijn
- ▶ Bewerk `debian/control`: verbeter `Description` en verwijder de standaardtekst onderaan
- ▶ Bewerk `debian/copyright`: verwijder de paragraaf met standaardtekst bovenaan, voeg jaren van copyright toe aan de `Files: *` stanza



Vertaling

Deze handleiding werd uit het Engels vertaald door Frans Spiesschaert en het Nederlandstalig lokalisatieteam.

Signaleer fouten in de vertaling en geef commentaar op het e-mailadres `<debian-l10n-dutch@lists.debian.org>`.

