Debian Developer's Reference

Version 14.18

Developer's Reference Team

2026-11-02

Table des matières

1 Portée de ce document **3**

2 Candidater pour devenir membre du projet **5**
2.1 Entrée en matière . . . 5
2.2 Mentors et parrains Debian . . . 6
2.3 Enregistrement comme responsable Debian . . . 6

3 Devoirs du développeur Debian **9**
3.1 Devoirs du responsable de paquet . . . 9
3.1.1 Œuvrer pour la prochaine publication `stable` . . . 9
3.1.2 Maintenance de paquets dans `stable` . . . 9
3.1.3 Gestion des bogues critiques pour la publication . . . 9
3.1.4 Coordination avec les développeurs amont . . . 10
3.2 Devoirs administratifs . . . 10
3.2.1 Mise à jour des renseignements auprès de Debian . . . 10
3.2.2 Gestion de clé publique . . . 11
3.2.3 Votes . . . 11
3.2.4 Decision making . . . 11
3.2.5 Départ en vacances poli . . . 11
3.2.6 Démission . . . 12
3.2.7 Revenir après démission . . . 12

4 Ressources pour les membres de Debian **13**
4.1 Listes de diffusion . . . 13
4.1.1 Règles d'utilisation fondamentales . . . 13
4.1.2 Principales listes de diffusion pour les responsables . . . 13
4.1.3 Listes particulières . . . 14
4.1.4 Demander une nouvelle liste pour le développement . . . 14
4.2 Canaux IRC . . . 14
4.3 Documentation . . . 14
4.4 Serveurs Debian . . . 15
4.4.1 Serveur de suivi des bogues (`BTS`) . . . 15
4.4.2 Serveur FTP principal `ftp-master` . . . 15
4.4.3 Serveur web principal `www-master` . . . 16
4.4.4 Serveur web pour pages personnelles `people` . . . 16
4.4.5 salsa.debian.org : dépôts Git et plateforme de développement collaborative . . . 16
4.4.6 GitHub.com : Submitting pull requests to upstream repositories . . . 16
4.5 Base de données des développeurs . . . 16
4.6 Archive Debian . . . 17
4.6.1 Sections . . . 18
4.6.2 Architectures . . . 19

4.6.3 Paquets . . . 19
4.6.4 Distributions . . . 19
4.6.4.1 Stable, testing, et unstable . . . 20
4.6.4.2 Informations complémentaires sur la distribution testing . . . 20
4.6.4.3 Experimental . . . 20
4.6.5 Noms de code des distributions . . . 21
4.7 Miroirs Debian . . . 22
4.8 Système « Incoming » . . . 22
4.9 Informations sur un paquet . . . 22
4.9.1 Sur le web . . . 22
4.9.2 Utilitaire dak ls . . . 23
4.10 Suivi de paquets Debian (Debian Package Tracker) . . . 23
4.11 Vue d'ensemble des paquets d'un développeur . . . 24
4.12 FusionForge pour Debian : Alioth . . . 24
4.13 Cadeaux pour les membres de Debian . . . 24

5 Gestion des paquets **25**
5.1 Nouveaux paquets . . . 25
5.2 Enregistrement des modifications . . . 26
5.3 Tests du paquet . . . 26
5.4 Agencement du paquet source . . . 27
5.5 Choix de distribution . . . 27
5.5.1 Cas particulier : distributions stable et oldstable . . . 27
5.5.2 Cas particulier : la publication stable-updates . . . 28
5.5.3 Cas particulier : testing/testing-proposed-updates . . . 29
5.6 Envois de paquets . . . 29
5.6.1 Source and binary uploads . . . 29
5.6.2 Envois sur ftp-master . . . 30
5.6.3 Envois différés . . . 30
5.6.4 Envois de sécurité . . . 30
5.6.5 Les autres files d'envoi . . . 30
5.6.6 Notifications . . . 30
5.7 Section, sous-section et priorité de paquet . . . 31
5.8 New upstream versions . . . 31
5.9 Manipulation des bogues . . . 32
5.9.1 Supervision des bogues . . . 32
5.9.2 Réponses aux bogues . . . 32
5.9.3 Gestion des bogues . . . 33
5.9.4 Fermeture des rapports de bogue lors des mises à jour . . . 34
5.9.5 Gestion des bogues de sécurité . . . 35
5.9.5.1 Gestionnaire de sécurité Debian (« Debian Security Tracker ») . . . 35
5.9.5.2 Confidentialité . . . 35
5.9.5.3 Annonces de sécurité . . . 36
5.9.5.4 Préparation de paquets pour les problèmes de sécurité . . . 36
5.9.5.5 Mise à jour du paquet corrigé . . . 37
5.10 Subscribing to package updates . . . 38
5.11 Manipulation de paquet dans l'archive . . . 38
5.11.1 Déplacement de paquet . . . 38
5.11.2 Suppression de paquet . . . 38
5.11.2.1 Suppression de paquet d'Incoming . . . 39
5.11.3 Remplacement et changement de nom de paquet . . . 39
5.11.4 Abandon de paquet . . . 39
5.11.5 Adoption de paquet . . . 40
5.11.6 Réintroduction de paquet . . . 40
5.12 Portage . . . 41
5.12.1 Courtoisie avec les porteurs . . . 41
5.12.2 Conseils aux porteurs pour les mises à jour . . . 42
5.12.2.1 Recompilation ou mise à jour indépendante binaire (binNMU) . . . 42

5.12.2.2 Quand utiliser une NMU source pour un portage . . . 43
5.12.3 Infrastructure de portage et automatisation . . . 43
5.12.3.1 Listes de diffusion et pages web . . . 43
5.12.3.2 Outils pour les porteurs . . . 43
5.12.3.3 `wanna-build` . . . 43
5.12.4 Paquet *non* portable . . . 44
5.12.5 Paquets non libres pouvant être automatiquement construits . . . 44
5.13 Mises à jour indépendantes (`NMU`) . . . 45
5.13.1 NMU : quand et comment . . . 45
5.13.2 NMU et `debian/changelog` . . . 46
5.13.3 Utilisation de la file d'attente `DELAYED/` . . . 46
5.13.4 NMU d'un point de vue du responsable . . . 47
5.13.5 Mise à jour indépendante source (`NMU`) vs binaire (`binNMU`) . . . 47
5.13.6 NMU et envoi de `QA` . . . 47
5.13.7 NMU et envoi d'équipe . . . 48
5.14 Sauvetage de paquets . . . 48
5.14.1 Quand un paquet devient-il éligible au sauvetage de paquet ? . . . 48
5.14.2 Comment sauver un paquet ? . . . 49
5.15 Maintenance collective . . . 49
5.16 La distribution `testing` . . . 50
5.16.1 Bases . . . 50
5.16.2 Mise à jour depuis `unstable` . . . 50
5.16.2.1 Désynchronisation . . . 51
5.16.2.2 Suppression de `testing` . . . 51
5.16.2.3 Dépendances circulaires . . . 51
5.16.2.4 Influence d'un paquet dans `testing` . . . 52
5.16.2.5 Détails . . . 52
5.16.3 Mises à jour directes dans `testing` . . . 52
5.16.4 Foire aux questions . . . 53
5.16.4.1 Quels sont les bogues bloquant l'intégration dans la version stable et comment sont-ils pris en compte ? . . . 53
5.16.4.2 Comment l'installation d'un paquet dans `testing` peut-elle casser d'autres paquets ? . . . 53
5.17 The Stable backports archive . . . 53
5.17.1 Bases . . . 53
5.17.2 Exception to the testing-first rule . . . 54
5.17.3 Who can maintain packages in the stable-backports archive ? . . . 54
5.17.4 When can one start uploading to stable-backports ? . . . 54
5.17.5 How long must a package be maintained when uploaded to stable-backports ? . . . 54
5.17.6 How often shall one upload to stable-backports ? . . . 54
5.17.7 How can one learn more about backporting ? . . . 54

6 Meilleures pratiques d'empaquetage **55**
6.1 Meilleures pratiques pour `debian/rules` . . . 55
6.1.1 Scripts d'assistance . . . 55
6.1.2 Paquets binaires multiples . . . 56
6.2 Meilleures pratiques pour `debian/control` . . . 56
6.2.1 The package name . . . 56
6.2.2 Conseils généraux pour les descriptions de paquets . . . 56
6.2.3 Résumé, ou description courte, d'un paquet . . . 57
6.2.4 Description longue . . . 57
6.2.5 Page d'accueil amont . . . 58
6.2.6 Emplacement du système de gestion de versions . . . 58
6.2.6.1 Vcs-Browser . . . 58
6.2.6.2 Vcs-* . . . 58
6.3 Meilleures pratiques pour `debian/changelog` . . . 59
6.3.1 Entrées de journalisation utiles . . . 59
6.3.2 Sélection de l'urgence du téléversement . . . 59

6.3.3 Idées reçues sur les entrées de journalisation . . . 60
6.3.4 Erreurs usuelles dans les entrées de journalisation . . . 60
6.3.5 Complément des journaux de modifications dans les fichiers NEWS.Debian . . . 61
6.4 Best practices around security . . . 61
6.5 Meilleures pratiques pour les scripts du responsable . . . 61
6.5.1 Generic guidelines . . . 62
6.5.2 Expected behavior of complex postinst scripts . . . 62
6.6 Gestion de la configuration avec debconf . . . 62
6.6.1 Proscrire les abus de debconf . . . 63
6.6.2 Recommandations générales pour les auteurs et les traducteurs . . . 63
6.6.2.1 Utilisation d'un anglais correct . . . 63
6.6.2.2 Courtoisie avec les traducteurs . . . 63
6.6.2.3 Correction (« unfuzzy ») des traductions pour des erreurs typographiques ou de frappe . . . 64
6.6.2.4 Proscrire toute supposition sur les interfaces utilisateurs . . . 64
6.6.2.5 Proscrire l'utilisation de la première personne . . . 65
6.6.2.6 Neutralité en genre . . . 65
6.6.3 Définition des champs de modèles (« templates »). . . . 65
6.6.3.1 Type . . . 65
6.6.3.2 Description : descriptions courte et étendue . . . 66
6.6.3.3 Choices . . . 66
6.6.3.4 Default . . . 67
6.6.4 Guide de style spécifique à certains champs de modèle . . . 67
6.6.4.1 Champ Type . . . 67
6.6.4.2 Champ Description . . . 67
6.6.4.3 Champ Choices . . . 67
6.6.4.4 Champ Default . . . 68
6.7 Internationalisation . . . 68
6.7.1 Gestion des traductions debconf . . . 68
6.7.2 Documentation internationalisée . . . 69
6.8 Best practices for debian/patches . . . 69
6.9 Situations courantes de gestion de paquets . . . 69
6.9.1 Paquets utilisant autoconf ou automake . . . 69
6.9.2 Bibliothèques . . . 70
6.9.3 Documentation . . . 70
6.9.4 Catégories particulières de paquets . . . 70
6.9.5 Données indépendantes de l'architecture . . . 71
6.9.6 Besoin de paramètres régionaux spécifiques lors de la construction . . . 71
6.9.7 Paquets de transition conformes à deborphan . . . 71
6.9.8 Meilleures pratiques pour les fichiers .orig.tar.{gz,bz2,xz} . . . 71
6.9.8.1 Source originelle (« pristine ») . . . 72
6.9.8.2 Source amont reconstruite . . . 72
6.9.8.3 Modification de fichier binaire . . . 73
6.9.9 Meilleures pratiques pour les paquets de débogage . . . 73
6.9.9.1 Paquets de débogage créés automatiquement . . . 73
6.9.9.2 Paquets -dbg manuels . . . 73
6.9.10 Meilleures pratiques pour les métapaquets . . . 74

7 Au-delà de l'empaquetage **75**
7.1 Signalement de bogues . . . 75
7.1.1 Signalement d'un grand nombre de bogues en une fois (mass bug filing) . . . 76
7.1.1.1 Étiquettes d'utilisateur (Usertags) . . . 76
7.2 Effort d'assurance qualité . . . 76
7.2.1 Travail quotidien . . . 76
7.2.2 Chasses aux bogues . . . 77
7.3 Contact avec d'autres responsables . . . 77
7.4 Gestion des responsables non joignables . . . 77
7.5 Dealing with repeatedly negligent Debian developers . . . 78

7.6 Interaction avec de futurs développeurs Debian . . . 78
7.6.1 Parrainage de paquets . . . 79
7.6.1.1 Parrainage d'un nouveau paquet . . . 79
7.6.1.2 Parrainage de la mise à jour d'un paquet existant . . . 80
7.6.2 Granting upload permissions to DMs . . . 81
7.6.3 Recommandation d'un nouveau développeur . . . 81
7.6.4 Gestion des nouvelles candidatures . . . 81

8 Internationalisation et traduction **83**
8.1 Gestion des traductions au sein de Debian . . . 83
8.2 FAQ I18N et L10N pour les responsables . . . 84
8.2.1 Comment faire en sorte qu'un texte soit traduit . . . 84
8.2.2 Comment faire en sorte qu'une traduction donnée soit relue . . . 84
8.2.3 Comment faire en sorte qu'une traduction donnée soit mise à jour . . . 84
8.2.4 Comment gérer un rapport de bogue concernant une traduction . . . 84
8.3 FAQ I18N et L10N pour les traducteurs . . . 84
8.3.1 Comment aider l'effort de traduction . . . 85
8.3.2 Comment fournir une traduction pour inclusion dans un paquet . . . 85
8.4 Meilleures pratiques actuelles concernant la l10n . . . 85

9 Aperçu des outils du responsable Debian **87**
9.1 Outils de base . . . 87
9.1.1 `dpkg-dev` . . . 87
9.1.2 `debconf` . . . 87
9.1.3 `fakeroot` . . . 88
9.2 Contrôle de paquets (« `lint` ») . . . 88
9.2.1 `lintian` . . . 88
9.2.2 `lintian-brush` . . . 88
9.2.3 `piuparts` . . . 88
9.2.4 `debdiff` . . . 88
9.2.5 `diffoscope` . . . 89
9.2.6 `duck` . . . 89
9.2.7 `adequate` . . . 89
9.2.8 `i18nspector` . . . 89
9.2.9 `cme` . . . 89
9.2.10 `licensecheck` . . . 89
9.2.11 `blhc` . . . 89
9.3 Assistance pour `debian/rules` . . . 90
9.3.1 `debhelper` . . . 90
9.3.2 `dh-make` . . . 90
9.3.3 `equivs` . . . 90
9.4 Construction de paquets . . . 90
9.4.1 `git-buildpackage` . . . 90
9.4.2 `debootstrap` . . . 91
9.4.3 `pbuilder` . . . 91
9.4.4 `sbuild` . . . 91
9.5 Envoi de paquets . . . 91
9.5.1 `dupload` . . . 91
9.5.2 `dput` . . . 91
9.5.3 `dcut` . . . 91
9.6 Automatisation de la maintenance . . . 91
9.6.1 `devscripts` . . . 91
9.6.2 `reportbug` . . . 92
9.6.3 `autotools-dev` . . . 92
9.6.4 `dpkg-repack` . . . 92
9.6.5 `alien` . . . 92
9.6.6 `elpa-dpkg-dev-el` . . . 92
9.6.7 `dpkg-depcheck` . . . 92

9.6.8 debputy . . . 93
9.7 Outils de portage . . . 93
9.7.1 dpkg-cross . . . 93
9.8 Documentation et information . . . 93
9.8.1 debian-policy . . . 93
9.8.2 doc-debian . . . 94
9.8.3 developers-reference . . . 94
9.8.4 maint-guide . . . 94
9.8.5 debmake-doc . . . 94
9.8.6 packaging-tutorial . . . 94
9.8.7 how-can-i-help . . . 94
9.8.8 docbook-xml . . . 94
9.8.9 debiandoc-sgml . . . 95
9.8.10 debian-keyring . . . 95
9.8.11 elpa-debian-el . . . 95

Équipe du guide de référence pour Debian <developers-reference@packages.debian.org>

— Copyright © 2019 - 2026 Holger Levsen
— Copyright © 2015 - 2020 Hideki Yamane
— Copyright © 2008 - 2015 Lucas Nussbaum
— Copyright © 2004 - 2007 Andreas Barth
— Copyright © 2002 - 2009 Raphaël Hertzog
— Copyright © 1998 - 2003 Adam Di Carlo
— Copyright © 1997 - 1998 Christian Schwarz

Ce manuel est un logiciel libre ; vous pouvez le redistribuer et le modifier selon les termes de la Licence Publique Générale GNU telle que l'a publiée la Free Software Foundation (fondation pour le logiciel libre), soit selon la version2, soit (à votre choix) selon toute autre version ultérieure.

Il est distribué dans l'espoir qu'il sera utile, mais sans aucune garantie ; sans même une garantie implicite de qualité loyale et marchande ni aptitude à un quelconque usage. Voyez la Licence Publique Générale GNU pour plus de détails.

Une copie de la Licence Publique Générale GNU est disponible dans /usr/share/common-licenses/GPL-2 dans la distribution Debian ou sur le World Wide Web sur le site Web de GNU. Vous pouvez aussi l'obtenir en écrivant à la Free Software Foundation, Inc., 51Franklin Street, Fifth Floor, Boston, MA02110-1301, USA.

Ceci est la version `14.18`, publiée le `2026-11-02`, du guide de référence du développeur pour Debian.

Si vous souhaitez imprimer ce guide, vous devriez utiliser la version PDF. Ce manuel est aussi disponible dans d'autres langues.

CHAPITRE 1

Portée de ce document

Le but de ce document est de fournir une vue d'ensemble des procédures recommandées et des ressources disponibles pour les développeurs et les responsables de paquet de Debian.

Les procédures traitées dans ce document incluent comment devenir un membre du projet (*Candidater pour devenir membre du projet*), comment créer un nouveau paquet (*Nouveaux paquets*) et l'envoyer (*Envois de paquets*), comment manipuler un rapport de bogue (*Manipulation des bogues*), comment déplacer, supprimer ou abandonner un paquet (*Manipulation de paquet dans l'archive*), comment porter des paquets (*Portage*) et enfin, comment et quand effectuer une publication intermédiaire de paquets d'autres responsables (*Mises à jour indépendantes (NMU)*).

Ce manuel présente entre autres les listes de diffusion (*Listes de diffusion*) et les serveurs (*Serveurs Debian*), la structure de l'archive Debian (*Archive Debian*), des explications sur les serveurs qui acceptent l'envoi de paquets (*Envois sur ftp-master*) et une présentation des outils qui peuvent aider un responsable à améliorer la qualité de ses paquets (*Aperçu des outils du responsable Debian*).

Ce manuel de référence ne présente pas les aspects techniques liés aux paquets Debian, ni comment les créer. Il ne décrit pas non plus les règles que doivent respecter les paquets Debian. Ces informations sont disponibles dans la Charte Debian.

De plus ce document *n'est pas l'expression d'une politique officielle*. Il contient de la documentation sur le système Debian et des conseils pratiques largement suivis. Ce n'est donc pas une sorte de guide de normes.

CHAPITRE 2

Candidater pour devenir membre du projet

2.1 Entrée en matière

Vous avez lu toute la documentation, vous avez examiné le guide du nouveau responsable (ou son successeur, Guide for Debian Maintainers), vous comprenez l'intérêt de tout ce qui se trouve dans le paquet d'exemple `hello` et vous vous apprêtez à empaqueter votre logiciel préféré. Comment en pratique devenir responsable Debian et intégrer votre travail au projet ?

Si vous ne l'avez pas encore fait, commencez par vous inscrire à la liste `debian-devel@lists.debian.org`. Pour cela, envoyez un courriel à `debian-devel-REQUEST@lists.debian.org` avec le mot `subscribe` en objet (champ `Subject`). En cas de problème, contactez l'administrateur de la liste `listmaster@lists.debian.org`. Vous trouverez plus d'informations en *Listes de diffusion*. `debian-devel-announce@lists.debian.org` est une autre liste incontournable pour suivre les développements de Debian.

Vous devriez vous inscrire et suivre les discussions de cette liste (sans poster) pendant quelque temps avant de coder quoi que ce soit, puis vous informerez la liste de votre intention de travailler sur quelque chose pour éviter de dupliquer le travail d'un autre.

Une autre liste intéressante est `debian-mentors@lists.debian.org`. Voir *Mentors et parrains Debian* pour les détails. Le canal IRC `#debian` pourra aussi être utile ; voir *Canaux IRC*.

Une fois choisie une façon de contribuer au projet Debian, vous devriez entrer en contact avec les responsables Debian qui travaillent sur des tâches similaires. Ainsi, vous pourrez apprendre auprès de personnes expérimentées. Si, par exemple, vous voulez empaqueter des logiciels existants, trouvez-vous un parrain. Un parrain est une personne qui travaillera sur vos paquets avec vous et les enverra dans l'archive Debian une fois satisfait de l'empaquetage. Pour trouver un parrain, envoyez une demande de parrainage à la liste `debian-mentors@lists.debian.org` en vous présentant et en décrivant votre paquet (voir *Parrainage de paquets* et https://wiki.debian.org/DebianMentorsFaq pour en savoir plus sur le sujet). Si vous préférez porter Debian sur une architecture ou un noyau alternatif, abonnez-vous aux listes dédiées au portage et demandez-y comment démarrer. Finalement, si vous êtes intéressé par la documentation ou l'assurance qualité (`QA`), contactez les responsables qui travaillent déjà sur ces tâches et proposez des correctifs et des améliorations.

One pitfall could be a too-generic local part in your email address : Terms like mail, admin, root, master should be avoided, please see https://www.debian.org/MailingLists/ for details.

2.2 Mentors et parrains Debian

La liste de diffusion `debian-mentors@lists.debian.org` a été mise en place pour les responsables débutants recherchant de l'aide avec l'empaquetage initial et d'autres problèmes de développeur. Chaque nouveau développeur est invité à s'abonner à cette liste (voir *Listes de diffusion* pour les détails).

Ceux qui préfèrent recevoir une aide plus personnalisée (par exemple, par courrier privé) devraient également envoyer des messages à cette liste et un développeur expérimenté se proposera de les aider.

In addition, if you have some packages ready for inclusion in Debian, but are waiting for your new member application to go through, you might be able to find a sponsor to upload your package for you. Sponsors are people who are official Debian Developers, and who are willing to criticize and upload your packages for you. Please read the debian-mentors FAQ at https://wiki.debian.org/DebianMentorsFaqfirst.

Pour devenir mentor ou parrain, plus d'informations sont disponibles en *Interaction avec de futurs développeurs Debian*.

2.3 Enregistrement comme responsable Debian

Before you decide to register with Debian, you will need to read all the information available at the New Members Corner. It describes in detail the preparations you have to do before you can register to become a Debian member. For example, before you apply, you have to read the Debian Social Contract. Registering as a member means that you agree with and pledge to uphold the Debian Social Contract; it is very important that members are in accord with the essential ideas behind Debian. Reading the GNU Manifesto would also be a good idea.

Le processus d'enregistrement a pour but de vérifier votre identité, vos intentions et vos compétences. Comme le nombre de personnes travaillant pour Debian a dépassé 1000 et que notre système est utilisé dans plusieurs endroits très importants, nous devons rester vigilants pour éviter un acte malveillant. C'est pourquoi nous contrôlons les nouveaux responsables avant de leur donner un compte sur nos serveurs et de les autoriser à ajouter des paquets dans l'archive.

Pour devenir responsable, il faudra montrer que vous pouvez faire du bon travail et que vous serez un bon contributeur. Pour cela, vous pourrez proposer des correctifs par le système de suivi des bogues (BTS) et maintenir un paquet parrainé par un responsable Debian pendant un temps. Nous attendons aussi des contributeurs qu'ils soient intéressés par le projet dans son ensemble et pas uniquement par leurs propres paquets. Si vous pouvez aider d'autres responsables en fournissant des informations sur un bogue ou même avec un correctif, faites-le !

Registration requires that you are familiar with Debian's philosophy and technical documentation. Furthermore, you need an OpenPGP key which has been signed by an existing Debian maintainer. If your OpenPGP key is not signed yet, you should try to meet a Debian Developer in person to get your key signed. There's a Key Signing Coordination page which should help you find a Debian Developer close to you. (If there is no Debian Developer close to you, alternative ways to pass the ID check may be permitted as an absolute exception on a case-by-case-basis. See the identification page for more information.)

Si vous n'avez pas de clé OpenPGP, créez-la. Tout développeur a besoin d'une clé OpenPGP pour signer et vérifier les mises à jour de paquets. Vous lirez la documentation du logiciel de cryptographie que vous utiliserez car elle contient des informations indispensables pour la sécurité de votre clé. Les défaillances de sécurité sont bien plus souvent dues à des erreurs humaines qu'à des problèmes logiciels ou à des techniques d'espionnage avancées. Voir *Gestion de clé publique* pour plus d'informations sur la gestion de votre clé publique.

Debian uses the `GNU Privacy Guard` (package `gnupg` version 2 or better) as its baseline standard. You can use some other implementation of OpenPGP as well. Note that OpenPGP is an open standard based on RFC 9580.

La longueur de votre clé doit être d'au moins 2048 bits (de préférence 4096 bits). Il n'y a pas de raison d'utiliser une clé plus petite et le faire serait bien moins sûr.

Si votre clé publique n'est pas sur un serveur public tel que `subkeys.pgp.net`, reportez-vous à la documentation disponible à l'Étape 2 : Vérification d'identité. Cette documentation explique comment placer votre clé publique sur un serveur. L'équipe en charge des nouveaux responsables placera votre clé publique sur les serveurs de clés si elle n'y est pas déjà.

Certains pays limitent l'usage des logiciels de cryptographie. Cela ne devrait cependant pas avoir d'impact sur l'activité d'un responsable de paquet car il peut être tout à fait légal d'utiliser des logiciels de cryptographie pour

l'authentification plutôt que pour le chiffrement. Si vous vivez dans un pays où l'utilisation de la cryptographie pour l'authentification est interdite, contactez-nous pour que nous prenions des dispositions particulières.

Pour faire acte de candidature, il vous faut un responsable Debian qui soutienne votre candidature (un intercesseur ou `advocate` en anglais). Après avoir contribué au projet Debian pendant quelque temps, quand vous choisissez de devenir un responsable Debian officiel, un responsable déjà enregistré avec qui vous aurez travaillé dans les derniers mois devra exprimer que, d'après lui, vous pouvez contribuer avec succès au projet Debian.

When you have found an advocate, have your OpenPGP key signed and have already contributed to Debian for a while, you're ready to apply. You can simply register on our application page. After you have signed up, your advocate has to confirm your application. When your advocate has completed this step you will be assigned an Application Manager who will go with you through the necessary steps of the New Member process. You can always check your status on the applications status board.

Pour plus de détails, consultez le coin des nouveaux membres Debian sur le site web de Debian. Assurez-vous de bien connaître les différentes étapes nécessaires au processus du nouveau membre avant de vous porter candidat. Si vous êtes bien préparé, vous gagnerez beaucoup de temps plus tard.

CHAPITRE 3

Devoirs du développeur Debian

3.1 Devoirs du responsable de paquet

En tant que responsable de paquet, vous êtes censé fournir des paquets de haute qualité qui s'intégreront correctement dans le système et qui sont conformes à la Charte Debian.

3.1.1 Œuvrer pour la prochaine publication `stable`

Fournir des paquets de haute qualité dans `unstable` ne suffit pas, la plupart des utilisateurs ne profiteront de vos paquets que quand ils seront publiés avec la prochaine version `stable`. Vous êtes donc censé collaborer avec l'équipe en charge de la publication pour veiller à ce que vos paquets soient intégrés.

Plus concrètement, vous devriez surveiller si vos paquets migrent vers `testing` (consultez *La distribution testing*). Lorsque la migration n'a pas lieu après la période d'essai, vous devriez analyser pourquoi et œuvrer pour corriger cela. Votre paquet pourrait avoir besoin d'être corrigé (dans le cas de bogues critiques pour la publication ou d'échecs de construction sur certaines architectures) mais cela peut également signifier mettre à jour (ou corriger, ou supprimer de `testing`) d'autres paquets pour permettre de terminer une transition dans laquelle votre paquet est enchevêtré à cause de ses dépendances. L'équipe en charge de la publication devrait pouvoir vous renseigner sur ce qui bloque actuellement une transition donnée si vous ne parvenez pas à l'identifier.

3.1.2 Maintenance de paquets dans `stable`

La plupart du travail de responsable de paquet consiste à fournir des versions de paquets mis à jour dans `unstable`, mais son travail implique aussi de s'occuper des paquets dans la publication `stable` actuelle.

Même si les modifications dans `stable` sont déconseillées, elles sont possibles. Chaque fois qu'un problème de sécurité est signalé, vous devriez collaborer avec l'équipe en charge de la sécurité pour fournir une version corrigée (consultez *Gestion des bogues de sécurité*). Quand des bogues de sévérité `important` (ou plus) sont soumis sur la version `stable` de vos paquets, vous devriez envisager la possibilité de fournir une correction spécifique. Vous pouvez interroger l'équipe en charge de la publication `stable` pour savoir si elle accepterait une telle mise à jour puis préparer un envoi vers `stable` (consultez *Cas particulier : distributions stable et oldstable*).

3.1.3 Gestion des bogues critiques pour la publication

Habituellement, vous devriez traiter les rapports de bogue sur vos paquets tel que cela est décrit en *Manipulation des bogues*. Cependant, une catégorie spéciale de bogues nécessite particulièrement votre attention : les bogues critiques pour la publication (`release-critical` ou `RC`). Tous les rapports de bogue de gravité `critical`, `grave` ou `serious` rendent le paquet inapproprié pour être inclus dans la prochaine version `stable`. Ils peuvent donc

retarder la publication de Debian (quand ils concernent un paquet de `testing`) ou bloquer des migrations vers `testing` (quand ils concernent seulement le paquet d'`unstable`). Au pire, ils pourraient conduire à la suppression du paquet. C'est pourquoi ces bogues doivent être corrigés au plus tôt.

If, for any reason, you aren't able to fix an RC bug in a package of yours within 2 weeks (for example due to time constraints, or because it's difficult to fix), you should mention it clearly in the bug report and you should tag the bug `help` to invite other volunteers to chime in. Be aware that RC bugs are frequently the targets of Non-Maintainer Uploads (see *Mises à jour indépendantes (NMU)*) because they can block the `testing` migration of many packages.

Un manque d'attention aux bogues critiques pour la publication est souvent considéré par l'équipe d'assurance qualité comme un signe de disparition d'un responsable n'ayant pas abandonné correctement son paquet. L'équipe MIA pourrait aussi s'impliquer, avec comme éventuelle conséquence l'abandon de vos paquets (consultez *Gestion des responsables non joignables*).

3.1.4 Coordination avec les développeurs amont

A big part of your job as Debian maintainer will be to stay in contact with the upstream developers. Debian users will sometimes report bugs that are not specific to Debian to our bug tracking system. These bug reports should be forwarded to the upstream developers so that they can be fixed in a future upstream release. Usually it is best if you can do this, but alternatively, you may ask the bug submitter to do it. Ideally, you also get upstream to *subscribe* for Debian Package Tracker notifications for their software in Debian.

Bien qu'il ne soit pas de votre responsabilité de corriger les bogues non spécifiques à Debian, vous pouvez le faire si vous en êtes capable. Quand vous faites de telles corrections, assurez-vous de les envoyer également au développeur amont. Les utilisateurs et développeurs Debian proposent parfois un correctif pour les bogues amont, il vous faudra alors évaluer ce correctif puis le transmettre aux développeurs amont.

Dans le cas où un rapport de bogue est transmis au développeur amont, n'oubliez pas que le service `bts-link` peut aider à la synchronisation de l'état des bogues dans le système de suivi de bogues amont et celui de Debian.

Si vous avez besoin de modifier les sources d'un logiciel pour fabriquer un paquet conforme à la Charte Debian, vous devriez proposer un correctif aux développeurs amont pour qu'il soit inclus dans leur version. Ainsi, vous n'aurez plus besoin de modifier les sources lors des mises à jour amont suivantes. Quels que soient les changements dont vous avez besoin, il faut toujours essayer de rester dans la lignée des sources amont.

As most upstreams nowadays use git for version control, in most cases git-buildpackage offers the most convenient way to create and maintain patches in Debian that so they are submit upstream. For details, see git-buildpackage man pages about using `pq` to write and test `debian/patches` as git commits, and having git remote `upstreamvcs` to easily cherry-pick patches to and from upstream git branches.

Si vous estimez que les développeurs amont sont ou deviennent hostiles envers Debian ou la communauté du logiciel libre, vous pouvez vouloir reconsidérer le besoin d'inclure le logiciel dans Debian. Parfois, le coût social à la communauté Debian ne vaut pas le bénéfice que le logiciel peut apporter.

3.2 Devoirs administratifs

Un projet de la taille de Debian repose sur certaines structures administratives pour garder une trace de tout. En tant que membre du projet, vous avez quelques devoirs pour veiller à ce que tout se déroule sans problème.

3.2.1 Mise à jour des renseignements auprès de Debian

Une base de données LDAP contient des informations sur les développeurs Debian en https://db.debian.org/. Vous devriez y entrer vos informations et les mettre à jour quand elles changent. Le plus important est de vous assurer que l'adresse vers laquelle est renvoyée le courrier à destination de votre adresse debian.org est toujours à jour, de même que l'adresse à laquelle vous recevez votre abonnement à debian-private si vous choisissez d'être abonné à cette liste.

Pour plus d'informations sur cette base de données, veuillez consulter *Base de données des développeurs*.

3.2.2 Gestion de clé publique

Be very careful with your private keys. Do not place them on any public servers or multiuser machines, such as the Debian servers (see *Serveurs Debian*). Back your keys up ; keep a copy offline. Read the documentation that comes with your software ; read the PGP FAQ and OpenPGP Best Practices.

Assurez-vous que votre clé est non seulement à l'abri des vols, mais aussi d'une perte. Générez et faites une copie (c'est même mieux sur papier) de votre certificat de révocation ; il est nécessaire si votre clé est perdue ou volée.

Si vous ajoutez des signatures à votre clé publique, ou des identifiants d'utilisateurs, vous pouvez mettre à jour le porte-clés Debian en envoyant votre clé au serveur de clefs à `keyring.debian.org`.Les mises à jour sont traitées au moins une fois par mois par les responsables du paquet `debian-keyring`.

Pour ajouter une nouvelle clé ou supprimer une ancienne clé, vous devez faire signer la nouvelle clé par un autre développeur. Si l'ancienne clé est compromise ou non valable, vous devez également ajouter le certificat de révocation. S'il n'y pas de bonne raison pour une nouvelle clé, les responsables du trousseau peuvent rejeter la nouvelle clé. Vous trouverez plus de détails en https://keyring.debian.org/replacing_keys.html.

Les mêmes routines d'extraction de clé décrites en *Enregistrement comme responsable Debian* s'appliquent.

Une présentation approfondie de la gestion de clé Debian peut être trouvée dans la documentation du paquet `debian-keyring` et sur le site https://keyring.debian.org/.

3.2.3 Votes

Bien que Debian ne soit pas vraiment une démocratie, le projet utilise un processus démocratique pour élire les responsables de projet et approuver les résolutions générales. Ces procédures sont définies par la constitution Debian.

En dehors de l'élection annuelle du responsable de projet, les votes ne se tiennent pas régulièrement et ne sont pas entrepris à la légère. Chaque proposition est tout d'abord discutée sur la liste de diffusion `debian-vote@lists.debian.org` et a besoin de plusieurs approbations avant que le secrétaire du projet n'entame la procédure de vote.

You don't have to track the pre-vote discussions, as the secretary will issue several calls for votes on `debian-devel-announce@lists.debian.org` (and all developers are expected to be subscribed to that list). Democracy doesn't work well if people don't take part in the vote, which is why we encourage all developers to vote. Voting is conducted via OpenPGP-signed/encrypted email messages.

La liste de toutes les propositions (passées et présentes) est disponible sur la page des informations sur les votes Debian, ainsi que des informations complémentaires sur la procédure à suivre pour effectuer une proposition, la soutenir et voter pour elle.

3.2.4 Decision making

Besides the issues covered in *Votes*, Debian is closer to a do-ocracy, than a democracy. Generally speaking, this means that decisions are taken by the people who do the work, which in most cases is package maintainers and DPL delegates. Public discussion :

- while encouraged and may be appreciated as input to take a decision,
- is not required to take a decision,
- nor can overrule a decision (the latter is possible via a General Resolution or by escalating to the Debian Technical Committee).

In the spirit of do-ocracy, members of the Debian community are expected to contribute in concrete and constructive ways, rather than demand that others do any kind of work for them.

For more on the social aspects of contributing to Debian, you are strongly encouraged to read the Debian Community Guidelines at https://people.debian.org/~enrico/dcg/

3.2.5 Départ en vacances poli

Il est courant que les développeurs s'absentent, que ce soit pour des vacances prévues ou parce qu'ils sont submergés de travail. L'important est que les autres développeurs ont besoin de savoir si vous êtes indisponible pour pouvoir agir en conséquence si un problème se produit sur vos paquets ou autre pendant votre absence.

Habituellement, cela signifie que les autres développeurs peuvent faire des NMU (voir *Mises à jour indépendantes (NMU)*) sur votre paquet si un gros problème (bogue empêchant l'intégration dans la distribution, mise à jour de sécurité, etc.) se produit pendant que vous êtes en vacances. Parfois, ce n'est pas très important, mais il est de toute façon approprié d'indiquer aux autres que vous n'êtes pas disponible.

Il y a deux choses à faire pour informer les autres développeurs. Premièrement, envoyez un courrier électronique à `debian-private@lists.debian.org` en commençant le sujet de votre message par « [VAC] » [1] et donnez la période de vos vacances. Vous pouvez également donner quelques instructions pour indiquer comment agir si un problème survenait.

L'autre chose à faire est de vous signaler comme en vacances (`on vacation`) dans la *Base de données des développeurs* (l'accès à cette information est réservé aux développeurs). N'oubliez pas de retirer l'indicateur `on vacation` à votre retour !

Ideally, you should sign up at the OpenPGP coordination pages when booking a holiday and check if anyone there is looking for signing. This is especially important when people go to exotic places where we don't have any developers yet but where there are people who are interested in applying.

3.2.6 Démission

Si vous décidez de quitter le projet Debian, veillez à procéder comme suit :

- Abandonnez tous vos paquets comme décrit en *Abandon de paquet* ;
- Retirez-vous des téléverseurs des paquets dont vous êtes co-responsable ou responsable en équipe.
- Si vous recevez des courriels d'alias d'adresse @debian.org (p. ex. press@debian.org) et ne le désirez plus, ouvrez un ticket `RT` pour les administrateurs du système Debian (`Debian System Administrators`). Écrivez simplement à `admin@rt.debian.org` avec « Debian RT » dans le sujet et en déclarant de quel alias vous voulez être supprimé.
- N'oubliez pas de vous retirer des équipes, par exemple des pages wiki des équipes ou des groupes de sala.
- Login into https://nm.debian.org/ and then open https://nm.debian.org/process/emeritus to request emeritus status and write a goodbye message that will be automatically posted on debian-private.

  Salsa is used as authentication provider to the NM site, in case you run into problems opening the retirement process yourself, contact NM front desk using `nm@debian.org`

Le processus précédemment décrit devrait absolument être suivi, car trouver les développeurs inactifs et déclarer orphelins leurs paquets est une tâche longue et fastidieuse.

3.2.7 Revenir après démission

Le compte d'un développeur est marqué « emeritus » (honoraire) quand le processus précédent en *Démission* est suivi, et « removed » (supprimé) sinon. Un développeur ayant démissionné avec un compte « emeritus » peut réactiver son compte de la façon suivante :

- Log on onto nm.debian.org using your salsa credentials.
- Follow the Return from Emeritus Wizard.
- Passer un processus raccourci de nouveau responsable (pour s'assurer qu'il connaît toujours les parties importantes de « philosophie et procédures » et « tâches et compétences »).

In case you run into problems contact `nm@debian.org` for further instructions.

Les développeurs ayant démissionné avec un compte « removed » doivent repasser le processus complet de nouveau membre.

1. Ainsi, le message peut être facilement filtré par les personnes qui ne veulent pas lire ces annonces de vacances.

CHAPITRE 4

Ressources pour les membres de Debian

Ce chapitre décrit brièvement les listes de diffusion, les serveurs Debian à disposition des développeurs et les autres ressources utiles au travail de responsable.

4.1 Listes de diffusion

Une grande partie des discussions entre les développeurs Debian (et les utilisateurs) a lieu dans un large éventail de listes de diffusion hébergées sur `lists.debian.org`. Pour en savoir plus sur la façon de s'abonner ou se désabonner, d'utiliser les listes, de consulter les archives, de contacter leurs responsables, ainsi que diverses autres informations sur les listes de diffusion, veuillez lire https://www.debian.org/MailingLists/. Cette section ne détaille que les informations utiles aux développeurs.

4.1.1 Règles d'utilisation fondamentales

Une réponse sur une liste de diffusion ne doit pas être envoyée en copie (`CC`) à l'expéditeur initial, sauf s'il l'a explicitement demandé. Toute personne écrivant sur une liste de diffusion devrait la suivre pour voir les réponses.

L'envoi d'un même message à plusieurs listes (« `cross-post` ») est déconseillé. Conformément aux usages, veuillez réduire la citation des articles auxquels vous répondez. En règle générale, veuillez respecter les conventions habituelles d'envoi de messages.

Veuillez lire le code de conduite pour plus de renseignements. Les recommandations de la communauté Debian (« *Debian Community Guidelines* ») valent également la peine d'être lues.

4.1.2 Principales listes de diffusion pour les responsables

Les principales listes de diffusion que les développeurs devraient suivre sont :

- `debian-devel-announce@lists.debian.org`, pour les annonces importantes aux développeurs. Tous les responsables Debian sont censés être inscrits à cette liste ;
- `debian-devel@lists.debian.org`, pour les diverses questions techniques relatives au développement ;
- `debian-policy@lists.debian.org`, où la Charte Debian (« `Debian Policy` ») est discutée et votée ;
- `debian-project@lists.debian.org`, pour les questions diverses et non techniques relatives au projet.

D'autres listes de diffusion sont spécialisées dans différents thèmes ; voir une liste sur https://lists.debian.org/.

4.1.3 Listes particulières

debian-private@lists.debian.org est une liste de diffusion destinée aux échanges privés entre développeurs Debian. Elle sert aux messages qui, pour une raison ou une autre, ne devraient pas être rendus publics. De ce fait, c'est une liste à faible trafic. Il est déconseillé d'utiliser debian-private@lists.debian.org sauf en cas de réelle nécessité. En outre, il ne faut *jamais* faire suivre un message provenant de cette liste à qui que ce soit. Les archives de cette liste ne sont pas disponibles sur la toile pour des raisons évidentes, mais il est possible de les consulter dans le répertoire ~debian/archive/debian-private/ sur master.debian.org.

debian-email@lists.debian.org est une liste de diffusion fourre-tout. Elle est utilisée pour les correspondances relatives à Debian qu'il serait utile d'archiver, telles que des échanges avec les auteurs amont à propos de licences, de bogues ou encore des discussions sur le projet avec d'autres personnes.

4.1.4 Demander une nouvelle liste pour le développement

Avant de demander une liste de diffusion pour le développement d'un paquet (ou d'un petit groupe de paquets apparentés), veuillez envisager l'utilisation plus appropriée d'un alias (à l'aide d'un fichier .forward-nom-alias sur master.debian.org, qui se traduit en une adresse plutôt agréable *vous-nom-alias@debian.org*).

Si une liste de diffusion standard sur lists.debian.org est vraiment ce que vous voulez, lancez-vous et faites une demande en suivant le guide.

4.2 Canaux IRC

Plusieurs canaux IRC sont dédiés au développement de Debian. Ils sont principalement hébergés sur le réseau **`Open and Free Technology Community (OFTC)`<https ://www.oftc.net/>`__**. L'entrée DNS irc.debian.org est un alias vers irc.oftc.net.

Le principal canal pour Debian est #debian. Il s'agit d'un canal important, généraliste, où les utilisateurs peuvent trouver des nouvelles récentes dans le sujet et qui est administré par des robots. #debian est destiné aux anglophones ; il existe également #debian.de, #debian-fr, #debian-br et d'autres canaux avec des noms analogues pour les personnes parlant d'autres langues.

Le canal principal pour le développement de Debian est #debian-devel. C'est un canal très actif puisque plus de 150 personnes sont connectées en permanence. C'est un canal pour les personnes qui travaillent sur Debian, ce n'est pas un canal d'aide (il existe #debian pour cela). Il est cependant ouvert à tous ceux qui veulent écouter (et apprendre). Le sujet est généralement rempli d'informations intéressantes pour les développeurs.

Comme #debian-devel est un canal ouvert, vous ne devriez pas y parler de problèmes discutés sur debian-private@lists.debian.org. Il existe un autre canal dans ce but, appelé #debian-private et protégé par clé. La clé est disponible dans le fichier master.debian.org:~debian/misc/irc-password.

D'autres canaux sont dédiés à des sujets spécifiques. #debian-bugs est utilisé pour la coordination des chasses aux bogues (« bug squashing parties »). #debian-boot est utilisé pour la coordination du travail sur l'installateur Debian (« debian-installer »). #debian-doc est utilisé occasionnellement pour travailler sur la documentation comme celle que vous lisez actuellement. D'autres canaux sont dédiés à une architecture ou un ensemble de paquets : #debian-kde, #debian-dpkg, #debian-perl, #debian-python, etc.

Des canaux existent pour développeurs non anglophones, par exemple, #debian-devel-fr pour les francophones intéressés au développement de Debian.

Channels dedicated to Debian also exist on other IRC networks.

4.3 Documentation

Ce document contient beaucoup d'informations très utiles aux développeurs Debian, mais il ne peut pas tout contenir. La plupart des autres documents intéressants sont référencés dans le coin du développeur Debian. Prenez le temps de parcourir tous les liens, vous apprendrez encore plus de choses.

4.4 Serveurs Debian

Debian possède plusieurs ordinateurs employés comme serveurs, dont la plupart hébergent les fonctions critiques du projet Debian. La plupart des machines sont utilisées pour des activités de portage et elles ont toutes un accès permanent à Internet.

La plupart des machines peuvent être utilisées par les développeurs tant qu'ils respectent les règles définies dans la charte d'utilisation des machines Debian.

Ces machines peuvent être utilisées à votre discrétion pour des buts liés à Debian. Veuillez cependant, par égard aux administrateurs système, ne pas utiliser de grandes quantités d'espace disque, de ressource réseau ou processeur sans obtenir auparavant l'accord des administrateurs. Ces machines sont d'habitude administrées par des bénévoles.

Veuillez prendre soin de vos mots de passe Debian ainsi que des clés SSH installées sur les machines Debian. Évitez les méthodes de connexion ou d'envoi de données qui envoient les mots de passe en clair par Internet comme Telnet, FTP, POP, etc.

Veuillez ne pas déposer de données non relatives à Debian sur les serveurs Debian à moins d'avoir préalablement obtenu la permission de le faire.

La liste à jour des machines Debian est disponible sur la page : https://db.debian.org/machines.cgi. Cette page web contient les noms des machines et permet d'accéder aux informations suivantes : contact, qui peut s'y connecter, clés SSH, etc.

Si vous rencontrez un problème en utilisant un serveur Debian et si vous estimez que les administrateurs système devraient en être avertis, vous pouvez vérifier la liste des tickets ouverts dans la file d'attente relative aux `DSA` (administrateurs du système Debian « `Debian System Administrators` ») du gestionnaire de demandes (« `request tracker` ») sur https://rt.debian.org/ (avec comme identifiant « debian » dont le mot de passe est disponible en `master.debian.org:~debian/misc/rt-password`). Pour signaler un nouveau problème, il suffit d'envoyer un message à `admin@rt.debian.org` en s'assurant d'indiquer « Debian RT » dans le sujet. Pour contacter l'équipe DSA par courriel, utilisez `dsa@debian.org` pour toute chose privée ou confidentielle ne devant pas être publique et `debian-admin@lists.debian.org` pour tout autre chose. L'équipe DSA est aussi présente sur le canal IRC d'OFTC `#debian-admin`.

Si le problème est lié à un service particulier, non relatif à l'administration système (paquet à supprimer de l'archive ou suggestion pour le site web par exemple), il faudra en général ouvrir un rapport de bogue sur un « pseudopaquet ». Consultez *Signalement de bogues* pour connaître la procédure à suivre.

Certains serveurs de base sont à accès restreint, mais les informations de ceux-ci sont fournies par d'autres serveurs miroirs.

4.4.1 Serveur de suivi des bogues (`BTS`)

`bugs.debian.org` est le serveur maître du système de suivi des bogues (« `Bug Tracking System` » ou `BTS`).

Si vous envisagez de manipuler les bogues ou d'en faire une analyse statistique, ce sera le bon endroit pour le faire. Informez la liste `debian-devel@lists.debian.org` de votre intention avant d'implémenter quoi que ce soit afin d'éviter un travail en double ou un gaspillage de temps machine.

4.4.2 Serveur FTP principal `ftp-master`

Le serveur `ftp-master.debian.org` est le serveur maître de l'archive Debian. En général, les paquets envoyés à ftp.upload.debian.org aboutissent sur ce serveur, voir *Envois de paquets*.

Ce serveur est à accès restreint ; un miroir est disponible sur `mirror.ftp-master.debian.org`.

Les problèmes avec l'archive Debian FTP doivent généralement être rapportés comme bogues sur le pseudopaquet `ftp.debian.org` ou par courrier électronique à `ftpmaster@debian.org` ; voir *Manipulation de paquet dans l'archive* pour connaître la procédure à suivre.

4.4.3 Serveur web principal `www-master`

Le serveur web principal est `www-master.debian.org`. Il héberge les pages web officielles, la façade de Debian pour la plupart des débutants.

Si vous rencontrez un problème avec un serveur web Debian, vous devriez envoyer un rapport de bogue sur le pseudopaquet `www.debian.org`. Vérifiez d'abord sur le système de suivi des bogues que le problème n'a pas déjà été signalé.

4.4.4 Serveur web pour pages personnelles `people`

`people.debian.org` est le serveur utilisé par les développeurs pour leurs pages concernant Debian.

Si vous avez des informations spécifiques à Debian que vous voulez rendre disponibles sur le web, vous pouvez le faire en les plaçant dans le répertoire `public_html` de votre répertoire personnel sur `people.debian.org`. Elles seront accessibles à l'adresse `https://people.debian.org/~`*votre-identifiant*`/`.

Vous ne devriez utiliser que cet emplacement particulier, car il sera sauvegardé alors que sur les autres serveurs, ce ne sera pas le cas.

Normalement, la seule raison d'utiliser un serveur différent est pour publier des informations soumises aux restrictions d'exportation américaines. Dans ce cas, vous pouvez utiliser un autre serveur situé en dehors des États-Unis.

Veuillez envoyer toute question à `debian-devel@lists.debian.org`.

4.4.5 salsa.debian.org : dépôts Git et plateforme de développement collaborative

Si vous souhaitez utiliser un dépôt git pour un de vos projets pour Debian, vous pouvez utiliser à cet effet l'instance de GitLab de Debian nommée Salsa. Gitlab offre aussi la possibilité de faire des requêtes d'intégration, de créer des pages wiki, de bénéficier d'un suivi de bogues entre autres services, aussi bien qu'un réglage fin des droits d'accès, pour faciliter le travail collaboratif sur des projets.

Pour plus d'informations, veuillez consulter la documentation sur la page https://wiki.debian.org/Salsa/Doc.

Any Debian package hosted on Salsa has also access to the Salsa CI . The Salsa CI pipeline mimics the tests that are run after each upload to Debian, but instead of having to wait for results or risk the health of the Debian repositories, Salsa CI provides you with instant feedback about any problems the changes you made may have created or solved.

4.4.6 GitHub.com : Submitting pull requests to upstream repositories

If some upstream repository is hosted on GitHub.com, you can use the Debian organization to create repository forks and submit changed branches with pull requests to upstream maintainers.

The organization is open to all Debian Members. To request membership, open an issue in the Debian/.github meta repository.

4.5 Base de données des développeurs

La base de données des développeurs, sur https://db.debian.org/, est un annuaire LDAP regroupant des informations sur les développeurs Debian. Vous pouvez utiliser cette ressource pour rechercher la liste des développeurs Debian. Une partie de ces informations est également disponible avec `finger` sur les serveurs Debian, essayez `finger votreidentifiant@db.debian.org` pour voir ce qu'il indique.

Les développeurs peuvent se connecter à la base de données pour modifier différentes informations les concernant, comme :

- l'adresse de réacheminement pour leur adresse debian.org ainsi que la façon de gérer les pourriels. Consultez https://db.debian.org/forward.html pour une description de toutes les options ;
- l'abonnement à debian-private ;
- l'état en vacances ou non ;
- des informations personnelles comme les adresse, pays, latitude et longitude de l'endroit où ils vivent pour utilisation dans la carte mondiale des développeurs Debian, numéros de téléphone et de fax, surnom IRC et page web ;
- le mot de passe et l'interpréteur de commande préféré sur les machines du projet Debian.

La plupart des informations ne sont naturellement pas publiques. Pour plus d'informations, veuillez lire la documentation en ligne sur https://db.debian.org/doc-general.html.

Les développeurs peuvent également envoyer leurs clés SSH afin de les utiliser pour authentification sur les machines Debian officielles et même ajouter de nouvelles entrées DNS du type *.debian.net. Ces fonctionnalités sont documentées sur https://db.debian.org/doc-mail.html.

4.6 Archive Debian

La distribution Debian est composée d'un grand nombre de paquets (environ `40000`) et de quelques autres fichiers (comme la documentation et les images de disque d'installation).

Voici un exemple d'arborescence pour une archive Debian complète :

```
dists/stable/main/
dists/stable/main/binary-amd64/
dists/stable/main/binary-armel/
dists/stable/main/binary-i386/
     ...
dists/stable/main/source/
     ...
dists/stable/main/disks-amd64/
dists/stable/main/disks-armel/
dists/stable/main/disks-i386/
     ...

dists/stable/contrib/
dists/stable/contrib/binary-amd64/
dists/stable/contrib/binary-armel/
dists/stable/contrib/binary-i386/
     ...
dists/stable/contrib/source/

dists/stable/non-free/
dists/stable/non-free/binary-amd64/
dists/stable/non-free/binary-armel/
dists/stable/non-free/binary-i386/
     ...
dists/stable/non-free/source/

dists/stable/non-free-firmware/
dists/stable/non-free-firmware/binary-amd64/
dists/stable/non-free-firmware/binary-armel/
dists/stable/non-free-firmware/binary-i386/
     ...
dists/stable/non-free-firmware/source/

dists/testing/
dists/testing/main/
     ...
dists/testing/contrib/
     ...
dists/testing/non-free/
     ...
dists/testing/non-free-firmware/
     ...

dists/unstable
```

(suite sur la page suivante)

(suite de la page précédente)

```
dists/unstable/main/
     ...
dists/unstable/contrib/
     ...
dists/unstable/non-free/
     ...
dists/unstable/non-free-firmware/
     ...

pool/
pool/main/a/
pool/main/a/apt/
     ...
pool/main/b/
pool/main/b/bash/
     ...
pool/main/liba/
pool/main/liba/libalias-perl/
     ...
pool/main/m/
pool/main/m/mailx/
     ...
pool/non-free/d/
pool/non-free/d/doc-rfc/
     ...
pool/non-free-firmware/f/
pool/non-free-firmware/f/firmware-nonfree/
     ...
```

Le répertoire racine contient deux répertoires : `dists/` et `pool/`. Le second contient un ensemble de répertoires où sont stockés les paquets, gérés dans la base de données de l'archive, et les programmes d'accompagnement. Le premier répertoire contient les distributions `stable`, `testing` et `unstable`. Les fichiers `Packages` et `Sources` des sous-répertoires de distribution font référence aux fichiers du répertoire `pool/`. Le découpage en sous-répertoires est identique d'une distribution à l'autre. Ce qui est exposé ci-dessous pour la distribution `stable` est également valable pour les distributions `unstable` et `testing`.

`dists/stable` contains four directories, namely `main`, `contrib`, `non-free` and `non-free-firmware`.

Dans chacune de ces sections, se trouve un répertoire contenant les paquets source (`source/`) et un répertoire pour chaque architecture gérée (`binary-i386`, `binary-amd64`, etc.).

La section `main` contient d'autres répertoires destinés aux images de disque et à plusieurs documents essentiels pour installer la distribution Debian sur chaque architecture (`disks-i386`, `disks-amd64`, etc.).

4.6.1 Sections

La section `main` de l'archive constitue la **distribution Debian officielle**. La section `main` est officielle parce qu'elle est entièrement conforme à toutes nos recommandations. Les deux autres sections divergent de ces recommandations à différents degrés, elles ne font donc **pas** officiellement partie de Debian.

Chaque paquet de la section `main` doit être conforme aux directives Debian pour le logiciel libre (« `Debian Free Software Guidelines` » ou `DFSG`) et à toutes les autres recommandations décrites dans la Charte Debian (« `Debian Policy Manual` »). Les DFSG constituent la définition du « logiciel libre » selon Debian. Reportez-vous à la Charte Debian pour en savoir plus.

Les paquets de la section `contrib` doivent être conformes aux DFSG, mais ne respectent pas d'autres contraintes. Ils peuvent, par exemple, dépendre de paquets de la section `non-free`.

Packages which do not conform to the DFSG are placed in the `non-free` or `non-free-firmware` sections. These

packages are not considered as part of the Debian distribution, though we enable their use, and we provide infrastructure (such as our bug-tracking system and mailing lists) for these non-free software packages.

The Debian Policy Manual contains a more exact definition of the four sections. The above discussion is just an introduction.

The separation of the four sections at the top-level of the archive is important for all people who want to distribute Debian, either via FTP servers on the Internet or on CD-ROMs : by distributing only the `main` and `contrib` sections, one can avoid any legal risks. Some packages in the `non-free` section do not allow commercial distribution, for example.

D'un autre côté, un distributeur de CD-ROM pourra facilement vérifier la licence de chacun des paquets de la section `non-free` et les intégrer si cela lui est autorisé (dans la mesure où cela varie énormément d'un distributeur à l'autre, ce travail ne peut être fait par les développeurs Debian).

Le terme « section » est également utilisé pour faire référence aux catégories (par exemple `admin`, `net`, `utils`, etc.), ce qui simplifie l'organisation des paquets disponibles et leur recherche. Il fut un temps où ces sections (ou plutôt sous-sections) existaient sous forme de sous-répertoires dans l'archive Debian. Maintenant, elles n'existent plus que dans le champ d'en-tête `Section` des paquets.

4.6.2 Architectures

À ses débuts, le noyau Linux existait seulement pour les architectures Intel i386 (ou postérieures) ; il en était de même pour Debian. Linux devenant de plus en plus populaire, le noyau a été porté vers d'autres architectures et Debian a commencé à les gérer. Comme si la gestion de nombreuses nouvelles architectures ne suffisait pas, Debian a décidé de construire des portages sur d'autres noyaux de type `Unix`, comme `hurd` et `kfreebsd`.

Debian GNU/Linux 1.3 était disponible uniquement pour `i386`. Debian 2.0 gérait les architectures `i386` et `m68k`. Debian 2.1 gérait les architectures `i386`, `m68k`, `alpha` et `sparc`. Depuis, Debian a considérablement évolué. Debian 9 gère un total de dix architectures Linux (`amd64`, `arm64`, `armel`, `armhf`, `i386`, `mips`, `mips64el`, `mipsel`, `ppc64el`, `s390` et deux architectures kFreeBSD (`kfreebsd-i386` et `kfreebsd-amd64`).

Pour chaque portage, des informations destinées aux développeurs et utilisateurs sont disponibles sur les pages de portages Debian.

4.6.3 Paquets

Il existe deux types de paquets Debian : les paquets `sources` et les paquets `binaires`.

Suivant son format, le paquet source peut être constitué d'un ou plusieurs fichiers en plus du fichier obligatoire `.dsc` :

- soit un fichier `.tar.gz`, soit un fichier `.orig.tar.gz` et un fichier `.diff.gz` pour le format « 1.0 » ;
- obligatoirement l'archive amont `.orig.tar.{gz,bz2,xz}`, éventuellement plusieurs archives amont supplémentaires `.orig-`*composant*`.tar.{gz,bz2,xz}` et l'archive debian obligatoire `debian.tar.{gz,bz2,xz}` pour le format « 3.0 (quilt) » ;
- une seule archive `.tar.{gz,bz2,xz}` pour le format « 3.0 (native) ».

Si un paquet est développé spécifiquement pour le projet Debian et n'est pas distribué en dehors, il n'y a qu'un fichier `.tar.{gz,bz2,xz}` qui contient les sources du programme, il est appelé paquet source « natif » (« `native` »). Si un paquet est distribué ailleurs aussi, le fichier `.orig.tar.{gz,bz2,xz}` contient ce que l'on appelle le `code source amont`, c'est-à-dire, le code source distribué par le `responsable amont` (il s'agit souvent de l'auteur du logiciel). Dans ce cas, le fichier `.diff.gz` ou `debian.tar.{gz,bz2,xz}` contient les modifications faites par le responsable Debian.

Le fichier `.dsc` liste tous les fichiers sources avec leurs sommes de contrôle (`md5sums`, `sha1sums`, `sha256sums`) et quelques informations supplémentaires concernant le paquet (responsable, version, etc.).

4.6.4 Distributions

L'organisation des répertoires présentée précédemment est elle-même contenue dans les `répertoires de distributions`. Chaque distribution est en fait incluse dans le répertoire `pool` à la racine de l'archive Debian.

Pour résumer, une archive Debian a un répertoire racine sur un serveur FTP. Par exemple, sur le site miroir `ftp.fr.debian.org`, l'archive Debian se trouve dans /debian qui est un emplacement courant (un autre emplacement courant est `/pub/debian`).

Une distribution est composée de paquets source et binaires, et des fichiers `Sources` et `Packages` correspondants, qui contiennent toutes les méta-informations sur les paquets. Les premiers sont dans le répertoire `pool/` tandis que les seconds sont dans le répertoire `dists/` de l'archive (pour rétrocompatibilité).

4.6.4.1 `Stable`, `testing`, et `unstable`

Il existe toujours une distribution appelée `stable` (dans le répertoire `dists/stable`), une distribution appelée `testing` (dans le répertoire `dists/testing`) et une distribution appelée `unstable` (dans le répertoire `dists/unstable`). Cela reflète le processus de développement du projet Debian.

Les développements se font sur la distribution `unstable` (c'est pourquoi elle est aussi appelée `distribution de développement`). Chaque développeur Debian peut modifier ses paquets à tout moment dans cette distribution. Ainsi son contenu change tous les jours. Comme aucun effort particulier n'est fait pour s'assurer que tout fonctionne correctement dans cette distribution, elle est parfois littéralement « instable ».

La distribution testing est générée automatiquement en prenant les paquets d'`unstable` s'ils satisfont à certains critères. Ces critères devraient garantir la bonne qualité des paquets de `testing`. La mise à jour de `testing` est effectuée deux fois par jour après l'installation des nouveaux paquets. Voir *La distribution testing*.

Après une période de développement, quand l'équipe de publication (« `release team` ») le juge opportun, la distribution `testing` est gelée, ce qui signifie que les conditions à remplir pour qu'un paquet passe d'`unstable` à `testing` sont durcies. Les paquets trop bogués sont supprimés et les seules mises à jours autorisées concernent les corrections de bogues. Après quelque temps, selon l'avancement, la distribution `testing` est gelée encore plus. Les détails de la gestion de la distribution `testing` sont publiés par l'équipe de publication sur la liste debian-devel-announce. Une fois les derniers problèmes résolus de façon satisfaisante pour l'équipe de publication, la distribution est publiée. La publication signifie que `testing` est renommée en `stable`, une nouvelle copie est créée pour la nouvelle `testing`, et l'ancienne `stable` est renommée en `oldstable` et y reste jusqu'à ce qu'elle soit finalement archivée. Lors de l'archivage, son contenu est déplacé sur `archive.debian.org`.

Ce cycle de développement est basé sur l'idée que la distribution `unstable` devient `stable` après une période de test dans `testing`. Une distribution contient inévitablement des bogues, même si elle est classée stable. C'est pourquoi la distribution stable est mise à jour de temps en temps. Les corrections introduites sont testées avec une grande attention et sont ajoutées une à une à l'archive pour diminuer les risques d'introduire de nouveaux bogues. Vous pouvez trouver les paquets proposés pour la prochaine mise à jour de `stable` dans le répertoire `proposed-updates`. De temps en temps, ces paquets du répertoire `proposed-updates` qui n'introduisent pas de régression sont installés ensemble dans la distribution `stable` et le numéro de révision de cette distribution est incrémenté (« 6.0 » devient « 6.0.1 », « 5.0.7 » devient « 5.0.8 » et ainsi de suite). Veuillez vous référer aux *Cas particulier : distributions stable et oldstable* pour plus de détails.

Note that development in `unstable` during the freeze should not be continued as usual, as packages are still build in `unstable`, before they migrate to `testing`, thus `unstable` should only contain packages meant for `testing`. Thus only upload to `unstable` during freezes, if you are planning to request an unblock (or if the package is not in `testing`).

Si vous souhaitez développer de nouvelles choses pour après le gel, envoyez les plutôt dans `experimental`.

4.6.4.2 Informations complémentaires sur la distribution `testing`

Les paquets sont habituellement installés dans la distribution `testing` après avoir subi suffisamment de tests dans `unstable`.

Pour plus de détails, veuillez consulter la section sur *La distribution testing*.

4.6.4.3 `Experimental`

La distribution `experimental` est particulière. Ce n'est pas une distribution à part entière comme le sont `stable`, `testing` et `unstable`. Elle sert de plate-forme de développement pour les projets expérimentaux qui risquent vraiment de détruire le système ou pour des logiciels vraiment trop instables pour être inclus dans la distribution

unstable (mais pour lesquels une mise en paquet est justifiée). Les utilisateurs qui téléchargent et installent des paquets d'experimental sont prévenus : on ne peut pas faire confiance à la distribution experimental.

Voici les lignes de sources.list 5 pour experimental :

```
deb http://deb.debian.org/debian/ experimental main
deb-src http://deb.debian.org/debian/ experimental main
```

Si un logiciel peut causer des dégâts importants, il sera sûrement préférable de le mettre dans la distribution experimental. Un système de fichiers compressé expérimental, par exemple, devrait probablement aller dans experimental.

Une nouvelle version amont de paquet qui introduit de nouvelles fonctions tout en supprimant de nombreuses autres ne devra pas être ajoutée à l'archive Debian, elle pourra cependant être ajoutée à experimental. Une nouvelle version non finalisée d'un logiciel qui utilise une méthode de configuration complètement différente pourrait aller dans experimental au gré du responsable. Si vous travaillez sur un cas de mise à niveau complexe ou incompatible, vous pouvez aussi utiliser experimental comme plate-forme d'intégration et ainsi fournir un accès aux testeurs.

Quelques logiciels expérimentaux peuvent cependant aller dans unstable, avec un avertissement dans la description, mais ce n'est pas recommandé, car les paquets d'unstable se propagent dans testing et aboutissent dans stable. Vous ne devriez pas avoir peur d'utiliser experimental, car cela ne cause aucun souci aux responsables de l'archive (« ftpmasters »), les paquets expérimentaux sont périodiquement enlevés quand vous envoyez le paquet dans unstable avec un numéro de version supérieur.

Un nouveau logiciel qui ne risque pas d'endommager le système ira directement dans unstable.

Une solution de rechange à experimental consiste à utiliser vos pages personnelles sur le serveur people.debian.org.

4.6.5 Noms de code des distributions

Chaque distribution Debian diffusée a un nom de code : Debian 11 s'appelle bullseye ; Debian 12, bookworm ; Debian 13, trixie. La prochaine publication Debian 14, s'appellera forky et 15 s'appellera duke. Il existe aussi une « pseudodistribution » nommée Sid ; il s'agit de la distribution unstable. Comme les paquets sont déplacés d'unstable vers testing quand ils sont suffisamment stables, la distribution Sid n'est jamais publiée. En plus du contenu habituel d'une distribution Debian, Sid contient des paquets pour des architectures qui ne sont pas encore officiellement prises en charge ou pour lesquelles la distribution n'a pas encore été publiée. Ces architectures seront intégrées ultérieurement à la distribution principale. Les noms de code et les versions des prochaines publications sont listés sur le site web.

Comme Debian est un projet de développement ouvert (où tout le monde peut participer et suivre les développements), même les distributions unstable et testing sont disponibles sur les serveurs HTTP et FTP de Debian. Si nous avions nommé le répertoire qui contient la future distribution « testing », il aurait fallu changer son nom en « stable » au moment de la publication, ce qui aurait forcé les miroirs FTP à télécharger de nouveau la distribution complète (qui est plutôt volumineuse).

D'un autre côté, si une distribution s'appelait Debian-x.y dès le départ, des personnes pourraient s'imaginer que la version x.y de Debian est disponible. (Cela s'est produit par le passé : un distributeur avait gravé un CD-ROM Debian 1.0 en utilisant une version de développement pré-1.0. C'est pour cette raison que la première version officielle était la version 1.1 et non la 1.0.)

En conséquence, les noms de répertoire de distribution dans l'archive sont déterminés par leur nom de code plutôt que par leur état de publication (« trixie » par exemple). Ces noms sont identiques pendant la période de développement et une fois la distribution diffusée. Des liens symboliques, qui peuvent être modifiés facilement, indiquent la distribution stable actuelle. Tout cela explique pourquoi les répertoires des distributions sont nommés à partir des noms de code des distributions alors que stable, testing et unstable sont des liens symboliques qui pointent vers les répertoires appropriés.

4.7 Miroirs Debian

Les différentes archives de téléchargement et le site web disposent de plusieurs miroirs pour soulager les serveurs principaux d'une charge importante. En fait, certains serveurs principaux ne sont pas publics — la charge est répartie sur une première série de serveurs. De cette façon, les utilisateurs ont toujours accès aux miroirs et s'y habituent, ce qui permet à Debian de mieux répartir les besoins en bande passante sur plusieurs serveurs et réseaux, et évite aux utilisateurs de surcharger l'emplacement primaire. Dans cette première série, les serveurs sont aussi à jour que possible, car la mise à jour est déclenchée par les sites maîtres internes.

Toutes les informations sur les miroirs Debian peuvent être trouvées sur https://www.debian.org/mirror/, y compris une liste des miroirs publics disponibles par FTP et HTTP. Cette page utile inclut également des informations et des outils pour créer son propre miroir, en interne ou pour un accès public.

Les miroirs sont souvent mis en œuvre par des tiers qui veulent aider Debian. C'est pourquoi les développeurs n'ont en général pas de compte sur ces machines.

4.8 Système « `Incoming` »

Le système « `Incoming` » est responsable de la collecte des paquets mis à jour et de leur installation dans l'archive Debian. Il est constitué d'un ensemble de répertoires et de scripts sur `ftp-master.debian.org`.

Les paquets sont envoyés par tous les responsables Debian dans un répertoire nommé `UploadQueue`. Ce répertoire est parcouru toutes les quelques minutes par un démon appelé `queued`, les fichiers `*.command` sont exécutés et les fichiers `*.changes` restants et correctement signés sont déplacés avec leurs fichiers correspondants dans le répertoire `unchecked`. Ce répertoire n'est pas visible pour la plupart des développeurs, car ftp-master est à accès restreint ; il est parcouru toutes les 15 minutes par le script `dak process-upload` qui vérifie l'intégrité des paquets envoyés et leurs signatures numériques. Si le paquet est considéré comme prêt à être installé, il est déplacé dans le répertoire `done`. S'il s'agit du premier envoi du paquet (ou s'il a de nouveaux paquets binaires), il est déplacé dans le répertoire `new` où il attend l'approbation des responsables de l'archive. Si le paquet contient des fichiers devant être installés manuellement, il est déplacé dans le répertoire `byhand` où il attend une installation manuelle par les responsables de l'archive. Sinon, quand une erreur a été détectée, le paquet est refusé et déplacé dans le répertoire `reject`.

Une fois le paquet accepté, le système envoie une confirmation par courrier au responsable et ferme les bogues corrigés. Ensuite, les compilateurs automatiques peuvent commencer leur travail. À ce moment, le paquet est accessible sur https://incoming.debian.org/ avant d'être vraiment installé dans l'archive Debian. Cette opération se produit quatre fois par jour (elle est aussi appelée « `dinstall run` » pour des raisons historiques) ; le paquet est alors supprimé de `incoming` et installé dans le `pool` avec les autres paquets. Une fois toutes les autres mises à jour (fabrication des nouveaux fichiers d'index `Packages` et `Sources` par exemple) effectuées, un script spécifique déclenche la mise à jour les miroirs primaires.

The archive maintenance software will also send the OpenPGP signed `.changes` file that you uploaded to the appropriate mailing lists. If a package is released with the `Distribution` set to `stable`, the announcement is sent to `debian-changes@lists.debian.org`. If a package is released with `Distribution` set to `unstable` or `experimental`, the announcement will be posted to `debian-devel-changes@lists.debian.org` or `debian-experimental-changes@lists.debian.org` instead.

Bien que ftp-master soit à accès restreint, une copie de l'installation est disponible à tous les développeurs sur `mirror.ftp-master.debian.org`.

4.9 Informations sur un paquet

4.9.1 Sur le web

Chaque paquet a plusieurs pages web dédiées. `https://packages.debian.org/`*nom-de-paquet* affiche chaque version du paquet disponible dans les différentes distributions. Chaque version fait un lien vers une page qui fournit des informations détaillées comme la description du paquet, les dépendances et des liens pour télécharger le paquet.

Le système de suivi des bogues trie les bogues par paquet. Les bogues de chaque paquet sont disponibles sur `https://bugs.debian.org/`*nom-de-paquet*.

4.9.2 Utilitaire `dak ls`

`dak ls` fait partie de la suite `dak` (« `Debian Archive Kit` ») et liste les versions de paquet disponibles pour toutes les distributions et architectures connues. L'outil `dak` est disponible sur `ftp-master.debian.org` et sur le miroir `mirror.ftp-master.debian.org`. Il utilise un seul paramètre qui correspond au nom du paquet. Un exemple vaut mieux qu'un long discours :

```
$ dak ls evince
evince     | 3.22.1-3+deb11u2 | oldstable           | source, amd64, arm64, armel,␣
↪armhf, i386, mips, mips64el, mipsel, ppc64el, s390x
evince     | 3.22.1-3+deb11u2 | oldstable-debug     | source
evince     | 3.30.2-3+deb12u1 | stable              | source, amd64, arm64, armel,␣
↪armhf, i386, mips, mips64el, mipsel, ppc64el, s390x
evince     | 3.30.2-3+deb12u1 | stable-debug        | source
evince     | 3.38.2-1         | testing             | source, amd64, arm64, armel,␣
↪armhf, i386, mips64el, mipsel, ppc64el, s390x
evince     | 3.38.2-1         | unstable            | source, amd64, arm64, armel,␣
↪armhf, i386, mips64el, mipsel, ppc64el, s390x
evince     | 3.38.2-1         | unstable-debug      | source
evince     | 40.4-1           | buildd-experimental | source, amd64, arm64, armel,␣
↪armhf, i386, mips64el, mipsel, ppc64el, s390x
evince     | 40.4-1           | experimental        | source, amd64, arm64, armel,␣
↪armhf, i386, mips64el, mipsel, ppc64el, s390x
evince     | 40.4-1           | experimental-debug  | source
```

Dans cet exemple, on peut voir que la version dans `unstable` n'est pas la même que dans `testing` où seul le binaire a été mis à jour indépendamment (« `binary-only NMU` ») pour toutes les architectures. Chaque version du paquet a été recompilée sur toutes les architectures.

You do not need an account on `ftp-master.debian.org` to get this information. The `rmadison` command, from the `devscripts` package (see *devscripts*), queries the same data over HTTPS; for example, `rmadison --suite=unstable --architecture=amd64 evince` restricts the output to one suite and one architecture. The data is also available directly from the web at `https://api.ftp-master.debian.org/madison?package=evince`.

4.10 Suivi de paquets Debian (Debian Package Tracker)

The Debian Package Tracker is an email and web-based tool to track the activity of a source package. You can get the same emails that the package maintainer gets, simply by *subscribing* to the package in the Debian Package Tracker.

Le suivi de paquets possède une interface web sur https://tracker.debian.org/ qui réunit beaucoup d'informations pour chaque paquet source. Plusieurs liens utiles sont proposés (BTS, statistiques QA, informations de contact, état de traduction DDTP, journaux de compilation automatique) et beaucoup d'autres informations provenant de différents endroits sont regroupés (les 30 dernières entrées de changelog, l'état dans `testing`, etc.). C'est un outil très pratique pour connaître ce qu'il en est d'un paquet source spécifique. De plus, une fois authentifié, il possible de s'inscrire ou désinscrire de n'importe quel paquet d'un simple clic.

Il est possible d'aller directement à la page web concernant un paquet source avec une URL comme `https://tracker.debian.org/pkg/`*paquet-source*.

Pour une information plus approfondie, vous devriez regarder dans sa documentation. Entre autres choses, elle explique comment interagir avec lui par courriel, comment filtrer les courriels qu'il transmet, comment configurer vos notifications de commit VCS, comment exploiter ses caractéristiques pour les équipes de responsables, etc.

4.11 Vue d'ensemble des paquets d'un développeur

Un portail web pour l'assurance qualité (« `quality assurance` » ou `QA`) sur https://qa.debian.org/developer.php affiche un tableau de tous les paquets d'un développeur (y compris ceux pour lesquels il est co-responsable). Le tableau donne un bon résumé sur les paquets d'un développeur : nombre de bogues par gravité, liste des versions disponibles, état des tests et des liens vers d'autres informations utiles.

C'est une bonne idée de vérifier régulièrement vos données pour ne pas oublier de bogues ouverts et quels paquets sont sous votre responsabilité.

4.12 `FusionForge` pour Debian : `Alioth`

Until Alioth was deprecated and eventually turned off in June 2018, it was a Debian service based on a slightly modified version of the FusionForge software (which evolved from SourceForge and GForge). This software offered developers access to easy-to-use tools such as bug trackers, patch managers, project/task managers, file hosting services, mailing lists, VCS repositories, etc.

Il existe des solutions de remplacement pour beaucoup des services offerts précédemment. Il est important de savoir qu'il reste de nombreuses références à alioth à corriger. Si vous rencontrez certaines de ces références, merci de prendre le temps d'essayer de les corriger, par exemple en remplissant des rapports de bogue, ou quand cela est possible, en corrigeant la référence.

4.13 Cadeaux pour les membres de Debian

Les avantages pour les membres de Debian sont documentés sur la page https://wiki.debian.org/MemberBenefits.

CHAPITRE 5

Gestion des paquets

Ce chapitre contient des informations relatives à la création, l'envoi, la maintenance et le portage des paquets.

5.1 Nouveaux paquets

Si vous voulez créer un nouveau paquet pour la distribution Debian, vous devriez commencer par consulter la liste des paquets en souffrance et paquets souhaités (« `Work-Needing and Prospective Packages` » ou `WNPP`). Vous pourrez ainsi vérifier que personne ne travaille déjà sur ce paquet et éviter un travail en double. Consultez aussi cette page si vous voulez en savoir plus.

Supposons que personne ne travaille sur le paquet que vous visez, vous devez alors envoyer un rapport de bogue (voir *Signalement de bogues*) concernant le pseudopaquet `wnpp`. Ce courrier devra décrire le paquet que vous projetez de créer, la licence de ce paquet et l'URL à laquelle le code source peut être téléchargé. Cette liste n'est pas limitative.

You should set the subject of the bug to `ITP:` *foo* `--` *short description*, substituting the *name of the new package* for *foo*. The severity of the bug report must be set to `wishlist`. Please send a copy to `debian-devel@lists.debian.org` by using the X-Debbugs-CC header (don't use CC :, because that way the message's subject won't indicate the bug number). If you are packaging so many new packages (>10) that notifying the mailing list in separate messages is too disruptive, send a summary after filing the bugs to the debian-devel list instead. This will inform the other developers about upcoming packages and will allow a review of your description and package name.

Veuillez ajouter « `Closes:` *#nnnnn* » au journal de modification (`changelog`) du nouveau paquet. Cette indication provoquera la fermeture automatique du rapport de bogue à l'installation du nouveau paquet dans l'archive (voir *Fermeture des rapports de bogue lors des mises à jour*).

Si vous jugez nécessaire d'ajouter des explications pour les administrateurs de la file d'attente de nouveaux paquets (`NEW`), veuillez les ajouter au fichier `changelog`, envoyer à `ftpmaster@debian.org` une réponse au message reçu en tant que responsable suite à votre envoi de paquet, ou une réponse au message de rejet si vous envoyez à nouveau le paquet.

Lors de la fermeture de bogues de sécurité, indiquez les numéros CVE en plus de « `Closes:` *#nnnnn* ». Cela permet à l'équipe de sécurité de suivre les failles. Si un envoi est effectué pour corriger le bogue avant que l'identifiant d'alerte soit connu, il est conseillé de modifier la mention existante du fichier `changelog` lors d'un envoi suivant. Même dans ce cas, veuillez inclure toutes les indications disponibles sur les origines de la situation dans la première entrée de `changelog`.

Les responsables sont priés d'annoncer leurs intentions pour plusieurs raisons :

— afin d'être informés si quelqu'un travaille déjà sur le paquet et pour permettre à d'autres membres de la liste de partager leur expérience ;
— si d'autres personnes envisagent de travailler sur le même paquet, elles apprendront qu'il existe un volontaire et pourront proposer de partager le travail ;
— cela permet aux autres responsables d'en apprendre plus sur le nouveau paquet que la description courte et la formule consacrée du journal de modification « `Initial release` » (publication initiale) envoyée sur `debian-devel-changes@lists.debian.org` ;
— cette information est utile aux utilisateurs d'`unstable` qui sont les premiers testeurs. Ces personnes devraient être incitées à essayer le nouveau paquet ;
— ces annonces donnent aux responsables et autres personnes intéressées une meilleure idée des évolutions et des nouveautés du projet.

5.2 Enregistrement des modifications

Les modifications apportées au paquet doivent être consignées dans le fichier `debian/changelog` pour être lisibles et compréhensibles par les humains. Ces notes doivent donner une description concise des changements, expliquer les raisons de ceux-ci (si ce n'est pas clair) et indiquer quels rapports de bogue ont été clos. Il faut aussi indiquer quand le paquet a été terminé. Ce fichier sera installé dans `/usr/share/doc/`*paquet*`/changelog.Debian.gz` ou `/usr/share/doc/`*paquet*`/changelog.gz` pour un paquet natif.

Le fichier `debian/changelog` a une structure précise comportant différents champs. Le champ `distribution` est décrit en *Choix de distribution*. Plus d'informations sur la structure de ce fichier sont disponibles dans la section « `debian/changelog` » de la Charte Debian (« `Debian Policy` »).

Certaines indications du fichier `changelog` peuvent provoquer la fermeture automatique des rapports de bogue au moment où le paquet est installé dans l'archive. Voir *Fermeture des rapports de bogue lors des mises à jour*.

Par convention, quand un paquet contient une nouvelle version amont, le fichier `changelog` comporte une ligne qui ressemble à :

```
* New upstream release.
```

There are tools to help you create entries and finalize the `changelog` for release — see *devscripts* (command `dch`), `git-buildpackage` (command `gbp dch`) and *elpa-dpkg-dev-el*.

Voir aussi *Meilleures pratiques pour debian/changelog*.

5.3 Tests du paquet

Avant d'envoyer un paquet, il faut effectuer quelques tests essentiels. Les opérations suivantes (une ancienne version du paquet est parfois nécessaire) devraient au moins être effectuées :

— Run `lintian` over the package. You can run `lintian` as follows : `lintian -v`*package-version*`.changes`. This will check the source package as well as the binary package. If you don't understand the output that `lintian` generates, try adding the `-i` switch, which will cause `lintian` to output a very verbose description of the problem.

  En principe, un paquet pour lequel `lintian` renvoie des erreurs (elles commencent par `E`) ne devrait *jamais* être envoyé.

  Pour en savoir plus sur `lintian`, voir *lintian* ;
— facultativement exécuter `debdiff` (voir *debdiff*) pour analyser les modifications depuis une ancienne version si celle-ci existe ;
— installer le paquet et s'assurer que le logiciel fonctionne sur un système `unstable` à jour ;
— mettre à niveau le paquet à partir d'une version plus ancienne vers la nouvelle version ;'
— retirer le paquet et l'installer à nouveau ;
— l'installation, la mise à niveau et le retrait des paquets peuvent être testés manuellement ou en utilisant l'outil `piuparts` ;
— With the package installed, run `adequate <package name>` to test its installed state for policy violations. Ideally, you should automate this step using `autopkgtest` (see *adequate*).

— copier le paquet source dans un répertoire différent puis tenter de le décompresser et de le reconstruire. Le but est de vérifier que la construction n'utilise pas de fichiers en dehors de ceux du paquet ou des permissions non préservées sur les fichiers contenus dans le fichier `.diff.gz`.

5.4 Agencement du paquet source

Il existe deux types de paquets source Debian :

— les paquets natifs (« `native` ») pour lesquels il n'y a pas de distinction entre les sources d'origine et les correctifs appliqués pour Debian ;
— les paquets (plus courants) avec au moins une archive, contenant les sources d'origine, accompagnée d'un fichier, contenant les modifications pour Debian.

Pour les paquets natifs, le paquet source comprend un fichier de contrôle source Debian (`.dsc`) et l'archive source (`.tar.{gz,bz2,xz}`). Un paquet source d'un paquet non natif comprend un fichier de contrôle source Debian, l'archive source d'origine (`.orig.tar.{gz,bz2,xz}`) et les modifications Debian (`.diff.gz` pour le format source « 1.0 » ou `.debian.tar.{gz,bz2,xz}` pour le format source « 3.0 (quilt) »).

Avec le format « 1.0 », le paquet est soit natif, soit non déterminé par `dpkg-source` au moment de la construction. Il est dorénavant recommandé de déterminer explicitement le format source en écrivant « 3.0 (quilt) » ou « 3.0 (native) » dans `debian/source/format`. La suite de cette partie ne traite que les paquets non natifs.

La première fois qu'un paquet est installé dans l'archive pour une version amont donnée, le fichier `tar` de cette version amont doit être envoyé et mentionné dans le fichier `.changes`. Par la suite, ce même fichier `tar` sera utilisé pour générer les fichiers `diff` et `.dsc` et il ne sera pas nécessaire de l'envoyer à nouveau.

Par défaut, `dpkg-genchanges` et `dpkg-buildpackage` incluront le fichier `tar` amont si et seulement si la précédente modification de `changelog` mentionne une version amont différente de la précédente. Ce comportement peut être modifié en utilisant `-sa` pour l'inclure systématiquement ou `-sd` pour ne jamais l'inclure.

Si la mise à jour ne contient pas le fichier `tar` des sources d'origine, `dpkg-source` *doit* utiliser le même fichier `tar` que celui déjà présent dans l'archive pour construire les fichiers `.dsc` et `diff` envoyés.

Dans des paquets non natifs, les permissions des fichiers non présents dans l'archive `*.orig.tar.{gz,bz2,xz}` ne seront pas préservées, car `diff` ne stocke pas les permissions dans le correctif. Néanmoins, en utilisant le format « 3.0 (quilt) », les permissions des fichiers du répertoire `debian` seront préservées puisqu'ils seront contenus dans une archive `tar`.

5.5 Choix de distribution

Chaque envoi doit indiquer à quelle distribution le paquet est destiné. Le processus de construction de paquet extrait cette information à partir de la première ligne du fichier `debian/changelog` et la place dans le champ `Distribution` du fichier `.changes`.

Les paquets sont généralement téléversés dans `unstable`. Les envois dans `unstable` ou `experimental` doivent utiliser ces noms de publication dans le fichier `changelog`. Les envois pour les autres publications doivent aussi utiliser les noms de code des publications, car ils évitent toute ambiguïté.

En fait, il y a d'autres distributions possibles : *nomdecode*`-security`, consultez *Gestion des bogues de sécurité* pour plus d'informations sur celles-ci.

Il n'est pas possible d'envoyer un paquet dans plusieurs distributions en même temps.

5.5.1 Cas particulier : distributions `stable` et `oldstable`

Envoyer un paquet pour la distribution `stable` signifie que le paquet sera dirigé vers la file d'attente `proposed-updates-new` pour être revu par les responsables de la publication `stable`. Une fois accepté, le paquet sera installé dans le répertoire `stable-proposed-updates` de l'archive Debian. Il sera ensuite ajouté à `stable` lors de la prochaine mise à jour de la distribution.

Uploads to a supported `stable` release should target their suite name in the changelog, i.e. `trixie` or `bookworm`. You should normally use `reportbug` and the `release.debian.org` pseudo-package to send a *source* `debdiff`,

rationale and associated bug numbers to the stable release managers, and await a request to upload or further information.

Si vous êtes certain que l'envoi sera accepté sans changement, n'hésitez pas à faire l'envoi en même temps que vous remplissez le bogue sur `release.debian.org`. Néanmoins, si vous êtes peu familier avec le processus, nous vous recommandons d'attendre l'approbation avant de faire l'envoi pour avoir la possibilité de voir si vos exigences correspondent à celle des responsables de publication.

Either way, there must be an accompanying bug for tracking, and your upload must comply with these acceptance criteria defined by the stable release managers. These criteria are designed to help the process be as smooth and frustration-free as possible.

- Le bogue que vous voulez corriger dans `stable` doit être déjà corrigé dans `unstable` (et pas en attente dans `NEW` ou dans la file d'attente `DELAYED`).
- Le bogue devrait être de sévérité « important » ou plus haute.
- Les métadonnées du bogue — les versions particulièrement affectées — doivent être à jour.
- Les correctifs doivent être minimaux et pertinents et inclure une entrée de journal de modification suffisamment détaillée.
- Un `debdiff` source des modifications projetées doit être inclus dans la requête (pas uniquement les correctifs bruts ou la mention « `a debdiff can be found at $URL` »).
- The proposed package must have a correct version number (e.g. `...+deb13u1/...~deb13u1` for `trixie` or `+deb12u1/~deb12u1` for `bookworm`) and you should be able to explain what testing it has had. See the Debian Policy for the version number : https://www.debian.org/doc/debian-policy/ch-controlfields.html#special-version-conventions
- La mise à jour doit être construite dans un environnement ou un chroot `stable` (ou `oldstable` si cette distribution est visée).
- Les corrections de problèmes de sécurité devraient être coordonnées avec l'équipe de sécurité à moins qu'elle n'ait déclaré explicitement qu'elle n'émettrait pas de `DSA` pour le bogue (par exemple, au moyen d'un indicateur « no-dsa » dans le *Gestionnaire de sécurité Debian (« Debian Security Tracker »)*).
- Do not close `release.debian.org` bugs in debian/changelog. They will be closed by the release team once the package has reached the respective point release.

Il est recommandé d'utiliser `reportbug`, car cela facilite la création de bogues avec des métadonnées correctes. L'équipe de publication fait un usage extensif d'étiquettes pour trier et gérer les requêtes, et les rapports mal étiquetés prennent plus de temps pour être remarqués et traités.

Les mises à jour de la distribution `oldstable` sont possibles tant qu'elle n'a pas été archivée. Les mêmes règles que pour `stable` s'appliquent.

Par le passé, les envois vers `stable` étaient également utilisés pour corriger les problèmes de sécurité. Cependant, cette pratique est déconseillée, car les mises à jour pour les avis de sécurité Debian (« `Debian security advisory` » ou `DSA`) sont automatiquement copiées dans l'archive `proposed-updates` appropriée quand l'avis est publié. Reportez-vous en *Gestion des bogues de sécurité* pour des informations plus détaillées sur la gestion des problèmes de sécurité. Si l'équipe en charge de la sécurité estime le problème trop insignifiant pour justifier un `DSA`, les responsables de la publication `stable` seront cependant plus facilement disposés à intégrer votre correctif par un envoi ordinaire vers `stable`.

5.5.2 Cas particulier : la publication `stable-updates`

Parfois les responsables de la publication `stable` décideront qu'une mise à jour vers `stable` devrait être mise à disposition des utilisateurs plus tôt que la prochaine mise à jour intermédiaire programmée. Dans ce cas, ils peuvent copier la mise à jour vers la publication `stable-updates`, dont l'utilisation est activée par défaut par l'installateur.

À l'origine, le processus décrit dans le *Cas particulier : distributions stable et oldstable* devrait être suivi comme d'habitude. Si vous pensez que l'envoi devrait être publié à travers `stable-updates`, mentionnez-le dans votre requête. Voici des exemples de circonstances pour lesquelles l'envoi peut être éligible pour ce type de traitement :

- La mise à jour est urgente, mais n'a pas de caractère de sécurité. Les mises à jour de sécurité continueront à être publiées dans l'archive de sécurité. Comme exemple on citera les paquets cassés par le fil du temps (cf. `spamassassin` et le problème de l'année 2010) et les correctifs des bogues introduits par les versions intermédiaires.
- Le paquet en question est un paquet de données qui doivent être mises à jour dans un délai raisonnable (par exemple, `tzdata`).

— La correction de paquets en bout de chaîne qui ont été cassés par des modifications externes (par exemple des utilitaires de téléchargement de vidéos et `tor`).
— Les paquets qui nécessitent d'être à jour pour être utiles (par exemple `clamav`).
— Uploads to `stable-updates` should target their suite name in the changelog as usual, e.g. `trixie`.

Une fois que l'envoi a été accepté dans `proposed-updates` et est prêt pour la publication, les responsables de la publication `stable` le copieront vers la publication *stable-updates* ` et publieront une annonce de mise à jour de `stable` (Stable Update Announcement ou `SUA`) sur la liste de diffusion `debian-stable-announce`.

Toutes les mises à jour publiées par `stable-updates` seront incluses dans `stable` avec la prochaine version intermédiaire comme d'habitude.

5.5.3 Cas particulier : `testing/testing-proposed-updates`

Veuillez consulter les informations des *Mises à jour directes dans testing* pour plus de détails.

5.6 Envois de paquets

5.6.1 Source and binary uploads

Each upload to Debian consists of a signed `.changes` file describing the requested change to the archive, plus the source and binary package files that are referenced by the `.changes` file.

If possible, the version of a package that is uploaded should be a source-only changes file. These are typically named `*_source.changes`, and reference the source package, but no binary `.deb` or `.udeb` packages. All of the corresponding architecture-dependent and architecture-independent binary packages, for all architectures, will be built automatically by the build daemons in a controlled and predictable environment (see *wanna-build* for more details).

For many source-only uploads you can use tag2upload instead, which means that you only need to push a signed Git tag to Salsa, instead of generating and signing `.dsc` and `.changes` files yourself. See https://wiki.debian.org/tag2upload.

There are several situations where a source-only upload is not possible.

The first upload of a new source package (see *Nouveaux paquets*) must include binary packages, so that they can be reviewed by the archive administrators before they are added to Debian.

If new binary packages are added to an existing source package, then the first upload that lists the new binary packages in `debian/control` must include binary packages, again so that they can be reviewed by the archive administrators before they are added to Debian. It is preferred for these uploads to be done via the `experimental` suite.

Uploads that will be held for review in other queues, such as packages being added to the `*-backports` suites, might also require inclusion of binary packages.

The build daemons will automatically attempt to build any `main` or `contrib` package for which the build-dependencies are available. Packages in `non-free` and `non-free-firmware` will not be built by the build daemons unless the package has been marked as suitable for auto-building (see *Paquets non libres pouvant être automatiquement construits*).

The build daemons only install build-dependencies from the `main` archive area. This means that if a source package has build-dependencies that are in the `contrib`, `non-free` or `non-free-firmware` archive areas, then uploads of that package need to include prebuilt binary packages for every architecture that will be supported. By definition this can only be the case for source packages that are themselves in the `contrib`, `non-free` or `non-free-firmware` archive areas.

Bootstrapping a new architecture, or a new version of a package with circular dependencies (such as a self-hosting compiler), will sometimes also require an upload that includes binary packages.

5.6.2 Envois sur `ftp-master`

Pour envoyer un paquet, il faut envoyer les fichiers (y compris les fichiers `changes` et `dsc` signés) par FTP anonyme sur `ftp.upload.debian.org` dans le répertoire /pub/UploadQueue/. Pour que les fichiers y soient traités, ils doivent être signés avec une clé du porte-clés (`keyring`) des développeurs ou des responsables Debian (voir https://wiki.debian.org/DebianMaintainer).

Attention, il est préférable de transférer le fichier `changes` en dernier. Dans le cas contraire, votre envoi pourrait être rejeté, car l'outil de maintenance de l'archive pourrait lire le fichier `changes` et constater que les fichiers ne sont pas tous présents.

Les paquets *dupload* ou *dput* pourront vous faciliter le travail lors du téléchargement. Ces programmes bien pratiques aident à automatiser le processus d'envoi de paquets vers Debian.

For removing packages or cancelling an upload, please see ftp://ftp.upload.debian.org/pub/UploadQueue/README and the Debian package *dcut*.

Enfin, vous devriez réfléchir au statut de votre paquet par rapport à `testing` avant l'envoi vers `unstable`. S'il a une version en attente de migration dans `unstable`, c'est généralement une bonne idée d'attendre sa migration avant d'envoyer une nouvelle version. Vous devriez aussi rechercher dans le *Suivi de paquets Debian (Debian Package Tracker)* les avertissements detransition pour éviter de procéder à des envois qui perturbent des transitions en cours.

5.6.3 Envois différés

It is sometimes useful to upload a package immediately, but to want this package to arrive in the archive only a few days later. For example, when preparing a *Mises à jour indépendantes (NMU)*, you might want to give the maintainer a few days to react, while at the same time having the certainty that your upload will eventually take effect if the maintainer does not. A one-shot approach of (i) file bug with a patch (or Merge Request) and (ii) at once upload to DELAYED, is much more efficient than having to wait for days between the two steps.

Les envois vers le répertoire différé sont gardés dans la file d'attente différée. Une fois le temps d'attente indiqué terminé, le paquet est déplacé dans le répertoire `incoming` normal pour être traité. Cela est réalisé par une mise à jour automatique en envoyant dans le répertoire `DELAYED/`*X*`-day` (*X* compris entre 0 et 15) de `ftp.upload.debian.org`. Le contenu de `0-day` est envoyé plusieurs fois par jour vers `ftp.upload.debian.org`.

With dput, you can use the `--delayed` *DELAY* parameter to put the package into one of the queues.

Sometimes you'll need to remove a previously uploaded package from the DELAYED queue (e.g. because the package maintainer has asked you so). dcut(1) has examples of how to do so.

5.6.4 Envois de sécurité

N'envoyez **jamais** un paquet vers la file d'envoi de sécurité (sur `*.security.upload.debian.org`) sans avoir auparavant obtenu l'autorisation de l'équipe de sécurité. Si le paquet ne correspond pas tout à fait aux besoins de cette équipe, il entraînera beaucoup de problèmes et de retards dans la gestion de cet envoi non désiré. Pour plus de précisions, consultez *Gestion des bogues de sécurité*.

5.6.5 Les autres files d'envoi

Une file d'attente alternative en Europe est disponible sur ftp://ftp.eu.upload.debian.org/pub/UploadQueue/. Son fonctionnement est similaire à `ftp.upload.debian.org`, mais devrait être plus rapide pour les responsables européens.

Les paquets peuvent également être envoyés à l'aide de ssh sur `ssh.upload.debian.org`; les fichiers doivent être placés dans `/srv/upload.debian.org/UploadQueue`. Cette file d'attente ne permet pas les *Envois différés*.

5.6.6 Notifications

Les administrateurs de l'archive Debian sont responsables de l'installation des mises à jour. La plupart des mises à jour sont gérées quotidiennement par le logiciel de gestion de l'archive `dak process-upload`. Les mises à jour de paquets sur la distribution `unstable` sont ainsi installées automatiquement. Dans les autres cas et notamment dans le cas d'un nouveau paquet, celui-ci sera installé manuellement. Il peut s'écouler un peu de temps entre l'envoi d'un paquet vers un serveur et son installation effective. Veuillez être patient.

Dans tous les cas, vous recevrez un accusé de réception par courrier électronique indiquant que votre paquet a été installé et quels rapports de bogue ont été clos. Veuillez lire attentivement ce courrier et vérifier que tous les rapports de bogue que vous vouliez clore sont bien dans cette liste.

L'accusé de réception indique aussi la section dans laquelle le paquet a été installé. S'il ne s'agit pas de votre choix, vous recevrez un second courrier qui vous informera de cette différence (voir ci-dessous).

Notez que si vous envoyez en utilisant les files d'attente, le démon vous enverra également une notification par courrier électronique.

Notez aussi que les nouveaux envois sont annoncés sur le *Canaux IRC* `#debian-devel-changes`. Si le vôtre échoue sans écho, cela peut être dû à ce que votre paquet n'est pas signé correctement, auquel cas vous pouvez trouver plus d'explications dans `ssh.upload.debian.org:/srv/upload.debian.org/queued/run/log`.

5.7 Section, sous-section et priorité de paquet

Les champs `Section` et `Priority` du fichier `debian/control` ne précisent pas vraiment l'endroit où le fichier sera placé dans l'archive, ni sa priorité. Afin de conserver l'intégrité globale de l'archive, ce sont les administrateurs de l'archive qui contrôlent ces champs. Les valeurs dans le fichier `debian/control` sont seulement indicatives.

Les administrateurs de l'archive indiquent les sections et priorités des paquets dans le fichier `override`. Si ce fichier `override` et le fichier `debian/control` du paquet diffèrent, vous en serez informé par courrier électronique quand le paquet sera installé dans l'archive. Vous pouvez corriger votre fichier `debian/control` avant votre prochain envoi ou alors vous pouvez modifier le fichier `override`.

Pour modifier la section dans laquelle un paquet est archivé, vous devez d'abord vérifier que le fichier `debian/control` est correct. Ensuite, envoyez un rapport de bogue sur le pseudopaquet `ftp.debian.org` demandant la modification de la section ou de la priorité de votre paquet. Utilisez un sujet comme `override: PACKAGE1:section/priorité, [...], PACKAGEX: section/priorité`, et exposez bien les raisons qui vous amènent à demander ces changements dans le corps de texte.

Pour en savoir plus sur les fichiers `override`, reportez-vous à dpkg-scanpackages 1 et https://www.debian.org/Bugs/Developer#maintincorrect.

Notez que le champ `Section` décrit à la fois la section et la sous-section, comme décrit en *Sections*. Si la section est `main`, elle devrait être omise. La liste des sous-sections autorisées peut être trouvée en https://www.debian.org/doc/debian-policy/ch-archive.html#s-subsections.

5.8 New upstream versions

To update a package for a new upstream release :

1. Read the upstream changelog, NEWS, and whatever other documentation they may have released with the new version.
2. If possible, inspect the full diff between the old and new upstream sources, potentially filtering out uninteresting parts using filterdiff (e.g. `filterdiff -x '*.po'`). If it's too big to review thoroughly, you can use `diffstat` to get a feel for the scope and nature of the changes (and thus where new bugs may appear), and to keep an eye for anything suspicious (e.g. unexpected use of network or the appearance of dubious binary blobs).
3. Port the old Debian packaging to the new version. This basically involves incrementing the `debian/changelog` and merging `debian/patches` from the old package to the new one.
4. Check to see if any bugs have been fixed that are currently open in the BTS. If they have been, close them in the `debian/changelog`.
5. If the patch/merge did not apply cleanly, figure out why. A patch may fail to apply if it's already been applied in the new upstream release, or if the upstream files the patch applies to have been substantially modified (or deleted).
6. If any changes were made to the build system (you'd know from steps 1 and 2), update the `debian/rules` and `debian/control` build dependencies if necessary.
7. Build the new package in an isolated environment, e.g. using `sbuild` or `pbuilder`. This ensures that all required build dependencies are listed in `debian/control` and eliminates the possibility of interference of any third party packages in your system.

8. Verify that the new package builds correctly and if so carry out the checks in *Tests du paquet*.

5.9 Manipulation des bogues

Chaque développeur doit être capable de travailler avec le système de suivi des bogues de Debian (« ``bubogue correctement (voir *Signalement de bogues*), comment les mettre à jour, les réordonner, les traiter et les fermer.

Les fonctionnalités du système de suivi des bogues sont décrites dans la documentation du BTS pour les développeurs : fermeture de bogues, envoi de messages de suivi, assignation de niveaux de gravité et de marques, indication que les bogues ont été transmis aux développeurs amonts, etc.

Des opérations comme réassigner des bogues à d'autres paquets, réunir des rapports de bogues séparés traitant du même problème ou rouvrir des bogues quand ils ont été prématurément fermés, sont gérées en utilisant le serveur de contrôle par courrier. Toutes les commandes disponibles pour ce serveur sont décrites dans la documentation du serveur de contrôle du BTS.

5.9.1 Supervision des bogues

Être un bon responsable implique de consulter régulièrement la page du système de suivi des bogues (BTS) de vos paquets. Le système de suivi des bogues contient tous les rapports de bogue qui concernent vos paquets. Vous pouvez les vérifier en consultant cette page : `https://bugs.debian.org/`*votrecompte*`@debian.org`.

Les responsables interagissent avec le système de suivi des bogues en utilisant l'adresse électronique `bugs.debian.org`. Vous trouverez une documentation sur les commandes disponibles à l'adresse https://www.debian.org/Bugs/ ou, si vous avez installé le paquet `doc-debian`, dans les fichiers locaux `/usr/share/doc/debian/bug-*`.

Certains trouvent utile de recevoir régulièrement une synthèse des rapports de bogue ouverts. Si vous voulez recevoir une synthèse hebdomadaire relevant tous les rapports de bogue ouverts pour vos paquets, vous pouvez configurer `cron` comme suit :

```
# ask for weekly reports of bugs in my packages
0 17 * * fri   echo "index maint address" | mail request@bugs.debian.org
```

Remplacez *address* par votre adresse officielle de responsable Debian.

5.9.2 Réponses aux bogues

Lors d'une réponse à un bogue assurez-vous que toutes les discussions concernant le bogue sont envoyées au rapporteur original du bogue, au bogue lui-même et (si vous n'êtes pas le responsable du paquet) au responsable. L'envoi d'un message à *123*`@bugs.debian.org` enverra le message au responsable du paquet et l'enregistrera dans le journal du bogue. Si vous ne vous souvenez pas de l'adresse courriel du rapporteur, vous pouvez aussi employer *123*`-submitter@bugs.debian.org` pour contacter le rapporteur du bogue. Cette adresse enregistre aussi le message dans le journal du bogue, ainsi, si vous êtes le responsable du paquet en question, il suffit de répondre à *123*`-submitter@bugs.debian.org`. Sinon, vous devrez aussi ajouter l'adresse *123*`@bugs.debian.org` afin d'atteindre également le responsable du paquet.

Si vous recevez un rapport de bogue mentionnant « FTBFS », cela signifie une erreur de construction à partir du paquet source (« `Fails To Build From Source` »). Les porteurs emploient fréquemment cet acronyme.

Une fois un bogue traité (c'est-à-dire qu'il est corrigé), marquez-le comme `done` (il sera fermé) en envoyant un message d'explication à *123*`-done@bugs.debian.org`. Si vous corrigez un bogue en changeant et en envoyant une nouvelle version du paquet, vous pouvez fermer le bogue automatiquement comme décrit en *Fermeture des rapports de bogue lors des mises à jour*.

Vous ne devez *jamais* fermer un rapport de bogue en envoyant la commande `close` à l'adresse `control@bugs.debian.org`. Si vous le faites, le rapporteur n'aura aucune information sur la clôture de son rapport.

5.9.3 Gestion des bogues

En tant que responsable de paquet, vous trouverez fréquemment des bogues dans d'autres paquets et recevrez des rapports de bogue sur vos paquets qui sont en fait relatifs à d'autres paquets. Les fonctions intéressantes du système de suivi des bogues sont décrites dans la documentation du BTS pour les développeurs Debian. Les instructions du serveur de contrôle du BTS documentent les opérations techniques du BTS, telles que comment remplir, réassigner, regrouper et marquer des bogues. Cette section contient des lignes directrices pour gérer vos propres bogues, définies à partir de l'expérience collective des développeurs Debian.

Remplir des rapports de bogue pour des problèmes que vous trouvez dans d'autres paquets est l'une des « obligations civiques » du responsable, voir *Signalement de bogues* pour les détails. Cependant, gérer les bogues de vos propres paquets est encore plus important.

Voici une liste des étapes que vous pouvez suivre pour traiter un rapport de bogue :

1. décider si le rapport correspond à un bogue réel ou non. Parfois, les utilisateurs utilisent simplement un programme d'une mauvaise façon, car ils n'ont pas lu la documentation. Si c'est votre diagnostic, fermez simplement le bogue avec assez d'informations pour laisser l'utilisateur corriger son problème (donnez des indications vers la bonne documentation et ainsi de suite). Si le rapport de bogue revient régulièrement, vous devriez vous demander si la documentation est assez bonne ou si le programme ne devrait pas détecter une mauvaise utilisation pour donner un message d'erreur informatif. Il s'agit d'un problème qui peut être discuté avec l'auteur amont.

   Si le rapporteur de bogue n'est pas d'accord avec votre décision de fermeture du bogue, il peut le rouvrir jusqu'à ce que vous trouviez un accord sur la façon de le gérer. Si vous n'en trouvez pas, vous pouvez marquer le bogue `wontfix` pour indiquer à tout le monde que le bogue existe, mais ne sera pas corrigé. Assurez-vous que le rapporteur du bogue comprend les raisons de votre décision en ajoutant une explication au message qui ajoute la marque `wontfix`.

   Si cette situation n'est pas acceptable, vous (ou le rapporteur) pouvez vouloir demander une décision par le comité technique en ouvrant un nouveau bogue sur le pseudopaquet `tech-ctte` avec un résumé de la situation. Avant de faire cela, veuillez lire la procédure recommandée ;
2. si le bogue est réel, mais causé par un autre paquet, réassignez simplement le bogue à l'autre paquet. Si vous ne savez pas à quel paquet il doit être réassigné, vous pouvez demander de l'aide sur *Canaux IRC* ou sur `debian-devel@lists.debian.org`. Veuillez informer le ou les responsables du paquet sur lequel est réassigné le bogue, par exemple en envoyant une copie du message de réassignation à *nomdupaquet*`@packages.debian.org`, en expliquant vos raisons. Attention, une simple réassignation n'envoie *pas* de courrier aux mainteneurs du paquet auquel le bogue est réassigné, de ce fait ils n'apprendraient l'existence du bogue qu'en regardant la vue d'ensemble des bogues relatifs à leurs paquets.

   Si le bogue affecte le fonctionnement de votre paquet, veuillez envisager de cloner le bogue avant de le réassigner au paquet qui provoque vraiment le comportement. Si vous procédez autrement, le bogue ne sera pas vu dans la liste des bogues sur votre paquet, au risque que d'autres utilisateurs signalent le même bogue de nouveau. Vous devriez marquer « votre » bogue bloqué par le clone réassigné afin de documenter la relation entre les deux bogues ;
3. parfois, vous devez également ajuster la gravité du bogue pour qu'elle corresponde à la définition de gravité des bogues. C'est dû au fait que les gens tendent à augmenter la gravité des bogues pour s'assurer que leurs bogues seront corrigés rapidement. La gravité de certains bogues peut même être rétrogradée en `wishlist` (souhait) quand le changement demandé est seulement superficiel ;
4. si le bogue est réel, mais que le problème a déjà été rapporté auparavant, alors les deux rapports devraient être rassemblés en un seul à l'aide de la commande `merge` du BTS. De cette façon, quand un bogue sera corrigé, tous les rapporteurs en seront informés (veuillez notez, néanmoins, qu'un courrier envoyé au rapporteur d'un des bogues ne sera pas automatiquement envoyé aux autres rapporteurs) ;
5. le rapporteur de bogue peut avoir oublié de fournir certaines informations. Dans ce cas, vous devez lui demander les informations nécessaires. Vous pouvez utiliser la marque `moreinfo` (plus d'information) sur le bogue. De plus, si vous ne pouvez pas reproduire le bogue, vous pouvez le marquer comme `unreproducible` (non reproductible). Une personne qui arriverait à reproduire le bogue est alors invitée à fournir plus d'informations sur la façon de le reproduire. Après quelques mois, si cette information n'a été envoyée par personne, le bogue peut être fermé ;
6. If the bug is related to the packaging, you just fix it. If you are not able to fix it yourself, then tag the bug as `help`. You can also ask for help on `debian-devel@lists.debian.org` or `debian-qa@lists.debian.`

org. See *Coordination avec les développeurs amont* if it's an upstream problem. If you have the required skills you can prepare a patch that fixes the bug and send it to the author at the same time. Make sure to send the patch to the BTS and to tag the bug as patch.

7. si un bogue a été corrigé sur la copie locale ou sur le système de gestion de versions, il peut être marqué pending (en attente) pour signaler qu'il est corrigé, et sera fermé à la prochaine mise à jour (ajouter « closes: » dans changelog). C'est d'autant plus utile si plusieurs développeurs travaillent sur le même paquet ;
8. une fois le paquet corrigé disponible dans l'archive, le bogue devrait être fermé en précisant dans quelle version du paquet il a été réglé. Cela peut être fait automatiquement, voir *Fermeture des rapports de bogue lors des mises à jour*.

5.9.4 Fermeture des rapports de bogue lors des mises à jour

Au fur et à mesure que les bogues et problèmes sont corrigés dans vos paquets, il est de votre responsabilité en tant que responsable du paquet de fermer les rapports de bogue associés. Cependant, vous ne devez pas les fermer avant que le paquet n'ait été accepté dans l'archive Debian. C'est pourquoi, vous pouvez et devriez clore les rapports dans le système de suivi des bogues une fois que vous avez reçu l'avis indiquant que votre nouveau paquet a été installé dans l'archive. Le bogue devrait être fermé avec la bonne version.

Cependant, il est possible de fermer automatiquement les bogues après l'envoi — indiquez simplement les bogues corrigés dans le fichier debian/changelog en suivant une syntaxe précise et le logiciel de maintenance de l'archive s'occupera de le fermer pour vous. Par exemple :

```
acme-cannon (3.1415) unstable; urgency=low

  * Frobbed with options (closes: Bug#98339)
  * Added safety to prevent operator dismemberment, closes: bug#98765,
    bug#98713, #98714.
  * Added man page. Closes: #98725.
```

D'un point de vue technique, l'expression rationnelle Perl suivante décrit comment sont identifiées les fermetures de bogue dans les lignes de changelog :

```
/closes:\s*(?:bug)?\#?\s?\d+(?:,\s*(?:bug)?\#?\s?\d+)*/ig
```

La syntaxe « closes: #*XXX* » est à préférer, car c'est la plus concise et facile à intégrer au texte de changelog. À moins de spécifier un comportement différent avec l'option -v de dpkg-buildpackage, seuls les bogues ainsi marqués dans l'entrée la plus récente de changelog seront fermés (de fait, seuls les bogues signalés dans la partie relative au journal de modification du fichier .changes sont fermés).

Historiquement, les envois identifiés comme *Mises à jour indépendantes (NMU)* étaient marqués comme fixed au lieu d'être fermés, mais cette pratique a cessé avec l'ajout du suivi des versions. Le même raisonnement s'applique à l'étiquette fixed-in-experimental.

If you happen to mistype a bug number or forget a bug in the changelog entries, don't hesitate to undo any damage the error caused. To reopen wrongly closed bugs, send a reopen *XXX* command to the bug tracking system's control address, control@bugs.debian.org. To close any remaining bugs that were fixed by your upload, email the .changes file to *XXX*-done@bugs.debian.org, where *XXX* is the bug number, and put Version : *YYY* and an empty line as the first two lines of the body of the email, where *YYY* is the first version where the bug has been fixed.

Rappelez-vous qu'il n'est pas obligatoire de fermer les bogues en utilisant le changelog tel que décrit ci-dessus. Si vous désirez simplement fermer les bogues qui n'ont rien à voir avec l'un de vos envois, faites-le simplement en envoyant une explication à *XXX*-done@bugs.debian.org. Vous ne devez **jamais** fermer des bogues dans une entrée du journal de modification (changelog) si les changements dans cette version n'ont rien à voir avec le bogue.

Pour une information plus générale sur ce qu'il faut mettre dans les entrées du journal de modification (changelog), voir *Meilleures pratiques pour debian/changelog*.

5.9.5 Gestion des bogues de sécurité

À cause de leur nature sensible, les bogues liés à la sécurité doivent être soigneusement traités. L'équipe de sécurité de Debian est là pour coordonner cette activité, pour faire le suivi des problèmes de sécurité en cours, pour aider les responsables ayant des problèmes de sécurité ou pour les corriger elle-même, pour envoyer les annonces de sécurité et pour maintenir `security.debian.org`.

Si vous prenez connaissance d'un bogue lié à un problème de sécurité sur un paquet Debian, que vous soyez ou non le responsable, regroupez les informations pertinentes sur le problème et contactez rapidement l'équipe de sécurité par courriel à `team@security.debian.org`. Si vous le désirez, vous pouvez chiffrer votre courriel avec la clé de contact de l'équipe de sécurité ; consultez https://www.debian.org/security/faq#contact pour plus de détails. **N'envoyez pas** de paquet pour `stable` sans contacter l'équipe de sécurité. Les informations utiles comprennent, par exemple :

- si le bogue a déjà été rendu public on non ;
- les versions du paquet affectées par le bogue. Vérifiez chaque version présente dans les distributions maintenues par Debian ainsi que dans `testing` et `unstable` ;
- la nature d'une solution si elle existe (les correctifs sont particulièrement utiles) ;
- tout paquet corrigé préparé par vous-même (envoyez seulement le fichier debdiff résultant ou autrement uniquement les fichiers `.diff.gz` et `.dsc` et lisez d'abord *Préparation de paquets pour les problèmes de sécurité*) ;
- toute assistance possible pour aider à tester (exploitation de faille, tests de régression, etc.) ;
- toute information utile pour l'annonce de sécurité (voir *Annonces de sécurité*).

En tant que responsable d'un paquet, il est de votre devoir de le maintenir, même dans la distribution stable. Vous êtes le mieux placé pour apprécier les correctifs et tester les paquets mis à jour, donc merci de vous référer aux sections suivantes sur la façon de préparer les paquets pour l'équipe en charge de la sécurité.

5.9.5.1 Gestionnaire de sécurité Debian (« `Debian Security Tracker` »)

L'équipe en charge de la sécurité gère une base de donnée centralisée, le gestionnaire de sécurité Debian (« `Debian Security Tracker` »). Il contient tous les renseignements publics à propos des problèmes de sécurité connus : quels sont les paquets et versions affectés et non affectés, et par conséquent si `stable`, `testing` ou `unstable` sont vulnérables. Les informations encore confidentielles ne sont pas ajoutées à la base de données.

Il est possible de rechercher un problème particulier, mais aussi un paquet. Cherchez parmi vos paquets afin de prendre connaissance de problèmes non encore résolus. Si vous le pouvez, veuillez fournir plus d'informations sur ces problèmes, ou aidez à les corriger dans vos paquets. Le mode d'emploi est disponible sur les pages web du gestionnaire.

5.9.5.2 Confidentialité

À la différence de la plupart des autres activités de Debian, les problèmes de sécurité doivent parfois être tenus secrets un certain temps. Cela permet aux distributeurs de logiciels de coordonner leur divulgation afin de minimiser l'exposition de leurs utilisateurs. Cette décision dépend de la nature du problème, de l'existence d'une solution correspondante, et de sa publicité.

Il existe plusieurs façons pour un développeur de prendre connaissance d'un problème de sécurité :

- il le remarque sur un forum public (liste de diffusion, site web, etc.) ;
- quelqu'un soumet un rapport de bogue ;
- quelqu'un l'informe en privé.

Dans les deux premiers cas, l'information est publique et il est important de régler le problème au plus vite. Dans le dernier cas, cependant, l'information n'est pas forcément publique. Il existe alors différentes possibilités pour traiter le problème :

- si l'exposition est mineure, il n'y a parfois pas besoin de garder le secret sur le problème et une correction devrait être mise en œuvre et diffusée ;
- si le problème est grave, il vaut mieux partager cette information avec d'autres distributeurs et de coordonner une publication. L'équipe de sécurité est en contact avec les différentes organisations et individus et peut s'en occuper.

Dans tous les cas, si la personne ayant indiqué le problème demande à ce que l'information ne soit pas diffusée, cela devrait être respecté, avec l'évidente exception d'informer l'équipe de sécurité pour préparer un correctif de la

version `stable` de Debian. Quand vous envoyez des informations confidentielles à l'équipe de sécurité, assurez-vous de bien le préciser.

Si le secret est nécessaire, vous ne pourrez pas envoyer de correctif vers `unstable` (ou ailleurs, comme un système de gestion de version public). Il ne suffit pas d'occulter les détails des modifications : puisque le code lui même est public, il peut être (et sera) étudié.

Il existe deux raisons de diffuser l'information même si le secret est demandé : le problème est connu depuis un certain temps, ou le problème ou son exploitation est devenu public.

L'équipe de sécurité dispose d'une clé PGP pour permettre de chiffrer tout échange d'informations pour les problèmes sensibles. Voir la FAQ de l'équipe Debian sur la sécurité pour plus de détails.

5.9.5.3 Annonces de sécurité

Les annonces de sécurité ne sont émises que pour la distribution actuellement `stable`, mais *pas* pour `testing` ou `unstable`. Une fois diffusée, l'annonce est envoyée à la liste `debian-security-announce@lists.debian.org` et mise en ligne sur la page d'informations de sécurité. Les annonces de sécurité sont écrites et mises en ligne par les membres de l'équipe en charge de la sécurité. Cependant, ils ne verront aucun inconvénient à ce qu'un responsable leur apporte des informations ou écrive une partie du texte. Les informations d'une annonce devraient comporter :

- une description du problème et de sa portée, y compris :
  - le type du problème (élévation de privilège, déni de service, etc.),
  - quels sont les privilèges obtenus et par quels utilisateurs (si c'est le cas),
  - comment il peut être exploité,
  - si le problème peut être exploité à distance ou localement,
  - comment le problème a été corrigé,

  ces informations permettent aux utilisateurs d'estimer la menace pesant sur leurs systèmes ;
- les numéros de version des paquets affectés ;
- les numéros de version des paquets corrigés ;
- une information sur la façon de récupérer les paquets mis à jour (habituellement l'archive de sécurité Debian) ;
- References to upstream advisories, CVE identifiers, and any other information useful in cross-referencing the vulnerability

5.9.5.4 Préparation de paquets pour les problèmes de sécurité

Une façon d'aider l'équipe de sécurité dans sa tâche est de lui fournir des paquets corrigés adéquats pour une annonce de sécurité de la version `stable` de Debian.

Quand une mise à jour de la version `stable` est effectuée, un soin particulier doit être apporté pour éviter de modifier le comportement du système ou d'introduire de nouveaux bogues. Pour cela, faites le moins de changements possibles pour corriger le bogue. Les utilisateurs et les administrateurs s'attendent à conserver un fonctionnement convenu dans une distribution lorsque celle-ci est publiée, donc toute modification est susceptible de casser le système de quelqu'un. Cela est spécialement vrai pour les bibliothèques : assurez-vous de ne jamais changer l'API (interface de programmation applicative) ou l'ABI (interface binaire-programme), aussi minime que soit le changement.

Cela signifie que passer à une version amont supérieure n'est pas une bonne solution. À la place, les changements pertinents devraient être rétroportés vers la version actuelle de la distribution `stable` de Debian. Habituellement, les développeurs amont veulent bien aider. Sinon, l'équipe de sécurité Debian peut le faire.

Dans certains cas, il n'est pas possible de rétroporter un correctif de sécurité, par exemple, quand de grandes quantités de code source doivent être modifiées ou réécrites. Si cela se produit, il peut être nécessaire de passer à une nouvelle version amont. Cependant, cela n'est fait que dans des situations extrêmes et vous devez toujours coordonner cela avec l'équipe de sécurité auparavant.

Une autre règle importante découle de ce qui précède : testez toujours vos changements. Si une exploitation du problème existe, essayez-la et vérifiez qu'elle réussit sur le paquet non corrigé et échoue sur le paquet corrigé. Testez aussi les autres actions normales, car un correctif de sécurité peut parfois casser de manière subtile des fonctionnalités apparemment découplées.

N'ajoutez **pas** de modifications au paquet qui ne soient pas directement liées à la correction de la vulnérabilité. Celles-ci devraient alors être enlevées, ce qui ne représentera qu'une perte de temps. S'il y a d'autres bogues dans

votre paquet que vous aimeriez corriger, faites un envoi vers proposed-updates de la façon habituelle, après l'envoi de l'alerte de sécurité. Le mécanisme de mise à jour de sécurité n'est pas un moyen d'introduire des changements dans votre paquet qui serait sinon rejeté de la distribution stable, veuillez donc ne pas essayer de le faire.

Examinez et testez autant que possible vos changements. Vérifiez les différences avec la version précédente de manière répétée (`interdiff` du paquet `patchutils` et `debdiff` du paquet `devscripts` sont des outils pratiques pour cela, voir *debdiff*).

Assurez-vous de garder les points suivants à l'esprit :

- **ciblez la bonne distribution** dans le fichier `debian/changelog` : *nomdecode*`-security` (par exemple `trixie-security`). Ne ciblez ni *distribution*`-proposed-updates`, ni `stable` !
- fournissez des entrées de `changelog` descriptives et complètes. D'autres personnes se baseront dessus pour déterminer si un bogue particulier a été corrigé. Ajoutez la déclaration `closes:` pour tout **bogue Debian**. Intégrez toujours une référence externe, de préférence un **identifiant CVE**, pour qu'elle puisse être recoupée. Néanmoins, si aucun identifiant CVE n'a encore été assigné, ne l'attendez pas et continuez le processus. L'identifiant pourra être référencé plus tard ;
- assurez-vous que le **numéro de version** est correct. Il doit être plus élevé que celui du paquet actuel, mais moins que celui des paquets des distributions suivantes. En cas de doute, testez-le avec `dpkg --compare-versions`. Soyez attentif à ne pas réutiliser un numéro de version déjà utilisé pour un précédent envoi, ou qui entrerait en conflit avec une mise à jour indépendante binaire (`binNMU`). Par convention, ajoutez `+debXu1` (où *X* est le numéro de publication majeure), par exemple `1:2.4.3-4+deb13u1`, bien sûr, incrémentez le nombre final (1 ici) lors des mises à jour suivantes ;
- à moins que l'archive source amont n'ait déjà été envoyée à `security.debian.org` (lors d'une précédente mise à jour de sécurité), construisez le paquet en incluant l'archive **source amont complète** (`dpkg-buildpackage -sa`). Si l'archive source amont a déjà été envoyée à `security.debian.org`, vous pouvez préparer le paquet en l'excluant (`dpkg-buildpackage -sd`) ;
- assurez-vous d'utiliser **exactement le même ``*.orig.tar.{gz,bz2,xz}``** que celui utilisé dans l'archive normale, sinon il ne sera pas possible de déplacer plus tard le correctif de sécurité dans l'archive principale ;
- compilez le paquet sur un **système propre**, où tous les paquets appartiennent à la distribution pour laquelle vous construisez le paquet. Si vous ne disposez pas d'un tel système, vous pouvez utiliser l'une des machines de debian.org (voir *Serveurs Debian*) ou mettre en place un chroot (voir *pbuilder* et *debootstrap*).

5.9.5.5 Mise à jour du paquet corrigé

N'envoyez **jamais** un paquet vers la file d'envoi de sécurité (sur `*.security.upload.debian.org`) sans avoir auparavant obtenu l'autorisation de l'équipe de sécurité. Si le paquet ne correspond pas tout à fait aux besoins de cette équipe, il entraînera beaucoup de problèmes et de retards dans la gestion de cet envoi non désiré.

Vous ne devez **jamais** envoyer votre correction dans `proposed-updates` sans vous coordonner avec l'équipe de sécurité. Les paquets seront copiés de `security.debian.org` au répertoire `proposed-updates` automatiquement. Si un paquet avec le même numéro de version ou un numéro plus grand est déjà installé dans l'archive, la mise à jour de sécurité sera rejetée par le système d'archive. Ainsi, la distribution `stable` se retrouvera plutôt sans la mise à jour de sécurité de ce paquet.

Une fois le nouveau paquet créé et testé, et qu'il a été approuvé par l'équipe de sécurité, il doit être envoyé pour être installé dans les archives. Pour les envois de sécurité, l'adresse d'envoi est `ftp://ftp.security.upload.debian.org/pub/SecurityUploadQueue/`.

Une fois l'envoi vers la file d'attente de sécurité accepté, le paquet sera automatiquement recompilé pour toutes les architectures et stocké pour vérification par l'équipe de sécurité.

Les envois en attente d'acceptation ou de vérification ne sont accessibles que par l'équipe de sécurité. C'est obligatoire, car il pourrait y avoir des correctifs pour des problèmes de sécurité qui ne peuvent pas encore être diffusés.

Si une personne de l'équipe de sécurité accepte un paquet, il sera installé sur `security.debian.org` et proposé pour le répertoire *distribution*`-proposed-updates` adéquat sur `ftp-master.debian.org`.

5.10 Subscribing to package updates

As a maintainer of a package, you get email notifications for various kinds of events (uploads, BTS messages etc). You can subscribe to receive such notifications also for packages you are not a maintainer of (or packages you maintain collaboratively), by mailing `control@tracker.debian.org` with `subscribe <pkgname>` either in the subject or the body of the email (see https://qa.pages.debian.net/distro-tracker/usage/messages.html#email-messages for details).

5.11 Manipulation de paquet dans l'archive

Certaines manipulations de l'archive ne sont pas possibles avec le processus de mise à jour automatisé. Elles sont effectuées manuellement par les responsables. Cette partie décrit la marche à suivre dans ces situations.

5.11.1 Déplacement de paquet

Il arrive parfois qu'un paquet change de section. Un paquet de la section `non-free` pourrait, par exemple, être distribué sous licence GNU GPL dans une nouvelle version ; dans ce cas, le paquet devrait être déplacé vers la section `main` ou `contrib`. [1]

Pour changer la section d'un paquet, modifiez les informations de contrôle pour placer le paquet dans la section voulue et envoyez-le à nouveau dans l'archive (voir la Charte Debian pour plus d'informations). Assurez-vous d'inclure le fichier `.orig.tar.{gz,bz2,xz}` dans l'envoi (même si vous n'envoyez pas de nouvelle version amont) sinon il n'apparaîtra pas dans la nouvelle section avec le reste du paquet. Si la nouvelle section est valable, il sera déplacé automatiquement. Si ce n'est pas le cas, contactez les responsables de l'archive (« `ftpmasters` ») pour comprendre ce qui s'est passé.

Pour changer la sous-section d'un paquet (`devel` ou `admin` par exemple), la procédure est légèrement différente. Modifiez la sous-section dans le fichier de contrôle du paquet et renvoyez-le. Il vous faudra ensuite demander la modification du fichier `override` comme décrit en *Section, sous-section et priorité de paquet*.

5.11.2 Suppression de paquet

If for some reason you want to completely remove a package (say, if it is an old compatibility library which is no longer required), you need to file a bug against `ftp.debian.org` asking that the package be removed ; as with all bugs, this bug should normally have normal severity. The bug title should be in the form `RM:` *package [architecture list]* `--` *reason*, where *package* is the package to be removed and *reason* is a short summary of the reason for the removal request. *[architecture list]* is optional and only needed if the removal request only applies to some architectures, not all. Note that the `reportbug` will create a title conforming to these rules when you use it to report a bug against the `ftp.debian.org` pseudo-package.

Si vous êtes responsable du paquet à supprimer, il faudrait le préciser dans le titre du rapport en commençant celui-ci par la mention `ROM` (« `Request Of Maintainer` », demande du responsable). De nombreux autres acronymes sont utilisés pour justifier la suppression d'un paquet, voir la liste complète sur https://ftp-master.debian.org/removals.html. Cette page fournit également une vue d'ensemble des requêtes en cours.

Seuls les paquets d'`unstable`, `experimental` ou `stable` peuvent être supprimés. Les paquets de `testing` ne sont pas supprimés directement. Ils sont plutôt enlevés automatiquement après suppression d'`unstable` et si aucun paquet de `testing` n'en dépend. (Les suppressions de `testing` sont possibles aussi en soumettant un bogue de suppression sur le pseudopaquet `release.debian.org`. Consultez la section *Suppression de testing*.)

Il existe une exception pour laquelle il n'est pas nécessaire de faire une demande explicite de suppression : si un paquet (source ou binaire) ne se construit plus depuis le source, il sera supprimé de façon semi-automatique. Pour un paquet binaire, cela veut dire qu'il n'y a plus de paquet source produisant ce paquet binaire ; si le paquet binaire n'est simplement plus produit pour certaines architectures, une demande de suppression est toujours nécessaire. Pour un paquet source, cela veut dire que tous les paquets binaires auxquels il se réfère ont été récupérés par un autre paquet source.

Il faut détailler dans la demande de suppression les raisons de cette demande. Cela a pour but d'éviter les suppressions indésirables et de garder une trace de la raison pour laquelle un paquet a été supprimé. Par exemple, vous

1. Reportez-vous à la Charte Debian (« Debian Policy Manual ») pour savoir dans quelle section un paquet doit être classé.

pouvez fournir le nom du paquet qui remplace celui à supprimer.

Normalement, vous ne devriez demander la suppression d'un paquet que si vous en êtes le responsable. Si vous voulez supprimer un autre paquet, vous devez obtenir l'accord de son responsable. Dans le cas d'un paquet orphelin, qui n'a donc pas de responsable, vous devriez discuter la demande de suppression sur `debian-qa@lists.debian.org`. S'il existe un consensus sur la suppression du paquet, vous devriez changer le titre et réassigner le bogue `O:` au paquet `wnpp` plutôt que d'en ouvrir un autre.

Further information relating to these and other package removal related topics may be found at https://wiki.debian.org/ftpmaster_Removalsand https://qa.debian.org/howto-remove.html.

If in doubt concerning whether a package is disposable, email `debian-devel@lists.debian.org` asking for opinions. Also of interest is the `apt-cache` program from the `apt` package. When invoked as `apt-cache showpkg` *package*, the program will show details for *package*, including reverse depends. Other useful programs include `apt-cache rdepends`, `apt-rdepends`, `build-rdeps` (in the `devscripts` package) and `grep-dctrl`. Removal of orphaned packages is discussed on `debian-qa@lists.debian.org`.

Une fois le paquet supprimé, les bogues du paquet doivent être gérés. Soit ils sont réassignés dans le cas où le code a évolué vers un autre paquet (par exemple, `libfoo12` a été supprimé parce que `libfoo13` le remplace), soit ils sont fermés si le logiciel ne fait simplement plus partie de Debian. Lors de la fermeture des bogues, pour éviter d'être marqués corrigés dans des versions du paquet disponibles dans des distributions précédentes de Debian, ils devraient être marqués corrigés dans la version `<dernière-version-existant-dans-Debian>+rm`.

5.11.2.1 Suppression de paquet d'`Incoming`

Par le passé, il était possible de supprimer un paquet d'`incoming`. Cependant, ce n'est plus possible depuis la mise en place du nouveau système. [4] À la place, il faut envoyer une nouvelle version du paquet avec un numéro de version plus élevé que celui à remplacer. Les deux versions seront installées dans l'archive, mais seule la plus récente sera réellement disponible dans `unstable`, car la précédente sera immédiatement remplacée par la nouvelle. Toutefois, si vous testez correctement vos paquets, vous ne devriez pas avoir besoin de les remplacer trop souvent.

5.11.3 Remplacement et changement de nom de paquet

Quand les responsables amont d'un de vos paquet décident de renommer leur logiciel (ou si vous vous trompez en nommant un paquet), vous devrez intervenir en deux étapes pour changer son nom. D'abord, modifiez le fichier `debian/control` pour que le nouveau paquet remplace (`Replaces`), fournisse (`Provides`) et entre en conflit avec (`Conflicts`) le paquet mal nommé (reportez-vous à la Charte Debian pour les détails). Vous ne devriez ajouter une relation `Provides` que si tous les paquets dépendants du paquet mal nommé continuent de fonctionner après le changement de nom. Une fois le paquet installé dans l'archive, faites un rapport de bogue concernant le pseudopaquet `ftp.debian.org` et demandez la suppression du paquet mal nommé (voir *Suppression de paquet*). N'oubliez pas de réassigner correctement les bogues du paquet en même temps.

Vous pourriez aussi commettre une erreur en construisant le paquet et voulant le remplacer. La seule façon de faire est d'incrémenter le numéro de version et d'envoyer une nouvelle version. L'ancienne version expirera de la façon habituelle. Notez que cela s'applique à chaque partie de votre paquet, y compris les sources : pour remplacer l'archive source amont de votre paquet, envoyez-la avec un numéro de version différent. Une possibilité simple est de remplacer `foo_1.00.orig.tar.gz` par `foo_1.00+0.orig.tar.gz` ou `foo_1.00.orig.tar.bz2`. Cette restriction permet à chaque fichier de l'archive d'avoir un nom unique, ce qui aide à garantir la cohérence dans le réseau des miroirs.

5.11.4 Abandon de paquet

If you can no longer maintain a package, you need to inform others, and see that the package is marked as orphaned. You should set the package maintainer to `Debian QA Group <packages@qa.debian.org>` and submit a bug report against the pseudo package `wnpp`. The bug report should be titled `O:` *package* `--` *short description* indicating that the package is now orphaned. The severity of the bug should be set to `normal`; if the package has a priority of standard or higher, it should be set to important. If you feel it's necessary, send a copy to `debian-devel@lists.debian.org` by putting the address in the X-Debbugs-CC : header of the message (no, don't use CC :, because that way the message's subject won't indicate the bug number).

4. Though, if a package still is in the upload queue and hasn't been moved to Incoming yet, it can be removed. (see *Envois sur ftp-master*)

If you just intend to give the package away, but you can keep maintainership for the moment, then you should instead submit a bug against `wnpp` and title it `RFA:` *package* `--` *short description*. `RFA` stands for `Request For Adoption`.

Vous pouvez trouver plus d'informations sur les pages web WNPP (« `Work-Needing and Prospective Packages` » : paquets en souffrance et paquets souhaités).

5.11.5 Adoption de paquet

Une liste des paquets en attente de nouveau responsable est disponible dans la liste des paquets en souffrance et paquets souhaités (WNPP). Afin de prendre en charge un paquet de cette liste, reportez-vous à la page mentionnée précédemment pour plus d'informations et les procédures à suivre.

Prendre un paquet sans l'accord du responsable actuel n'est pas correct — ce serait un détournement de paquet. Vous pouvez, bien sûr, prendre contact avec le responsable actuel et lui demander si vous pouvez prendre en charge ce paquet.

Cependant, si un paquet est abandonné par son responsable, vous pouvez prendre la responsabilité de ce paquet en suivant le processus de récupération décrit dans *Sauvetage de paquets*. Si vous avez une raison de croire que le responsable n'est plus du tout actif, consultez *Gestion des responsables non joignables*.

Les plaintes à propos des responsables devraient être portées sur la liste de diffusion des développeurs. Si la discussion ne se termine pas par une conclusion positive et que le problème est de nature technique, envisagez de porter le cas à l'attention du comité technique (voir la page web du comité technique pour plus d'informations).

Si vous reprenez un vieux paquet, vous voudrez sûrement que le système de suivi des bogues indique que vous êtes le responsable du paquet. Cela se produira automatiquement une fois installé une nouvelle version du paquet dans l'archive avec le champ `Maintainer` à jour. Cela peut prendre quelques heures après l'envoi. Si vous ne pensez pas faire de mise à jour avant un moment, vous pouvez utiliser le *Suivi de paquets Debian (Debian Package Tracker)* pour recevoir les rapports de bogue. Cependant, assurez-vous que l'ancien responsable n'est pas embêté de recevoir les rapports de bogue en attendant.

To change the maintainer of a pseudo-package, e-mail `owner@lists.debian.org` and ask them to update their `pseudo-packages.maintainers` file. Include the package name, the name and e-mail address for the old and new maintainer, and a reason for the change. Remember to CC the old maintainer on the e-mail.

5.11.6 Réintroduction de paquet

Les paquets sont souvent supprimés à cause de bogues critiques pour la publication, de mainteneurs absents, de trop peu d'utilisateurs ou d'une médiocre qualité générale. Alors que le processus de réintroduction de paquet est similaire au processus d'empaquetage initial, certains pièges peuvent être évités en commençant par un peu de recherche historique.

Vous devriez d'abord vérifier la raison pour laquelle le paquet a été supprimé. Ces renseignements sont disponibles dans le paragraphe suppression (« removal ») de la section de nouvelles du PTS pour le paquet ou en consultant le journal des suppressions. Le bogue de suppression indiquera la raison pour laquelle le paquet a été supprimé et donnera quelques indications sur les points à travailler avant de réintroduire le paquet. Cela pourrait indiquer que la meilleure solution est d'utiliser un autre programme au lieu de réintroduire le paquet.

Contacter les précédents responsables peut être utile pour savoir s'ils ont commencé à réintroduire le paquet et s'ils sont intéressés à être coresponsables du paquet ou à parrainer le paquet si nécessaire.

Toutes les étapes nécessaires à l'introduction d'un nouveau paquet (*Nouveaux paquets*) devraient être suivies.

Le travail devrait être basé sur la dernière version pertinente du paquet disponible. Ce pourrait être la dernière version d'`unstable`, toujours disponible dans l'archive d'instantanés.

Le système de gestion de versions utilisé par les précédents mainteneurs pourrait contenir des modifications utiles, y jeter un œil pourrait donc être une bonne idée. Vérifiez si le fichier `control` du précédent paquet contient des en-têtes pointant vers le système de gestion de versions du paquet et s'il existe encore.

Les suppressions de paquet d'`unstable` (pas de `testing`, `stable` ou `oldstable`) déclenchent la fermeture de tous les bogues relatifs au paquet. Vous devriez passer en revue tous les bogues fermés (y compris les bogues archivés) puis extraire et rouvrir tous ceux qui ont été fermés avec une version se finissant par `+rm` et toujours

d'actualité. Tous ceux qui ne s'appliquent plus devraient être marqués comme corrigés dans la version adéquate si elle est connue.

Les suppressions de paquet d'`unstable` déclenchent aussi le marquage du paquet comme supprimé dans le *Gestionnaire de sécurité Debian (« Debian Security Tracker »)*.Les membres de Debian devraient marquer les problèmes supprimés comme non corrigés dans le dépôt du suivi de sécurité et tous les autres devraient contacter l'équipe de sécurité pour rapporter les paquets réintroduits.

5.12 Portage

Debian gère un nombre croissant d'architectures. Même si vous n'êtes pas un porteur et que vous utilisez une seule architecture, il est de votre responsabilité de développeur d'être attentif aux questions de portabilité. C'est pourquoi il est important de lire ce chapitre même si vous n'êtes pas un porteur.

Porter un paquet consiste à compiler un paquet binaire pour des architectures différentes de celle du paquet binaire du responsable du paquet. C'est une activité remarquable et essentielle. En fait, les porteurs sont à l'origine de la plupart des compilations de paquets Debian. Par exemple, quand un paquet source (portable) est envoyé avec les paquets binaires `i386`, il faut compter une compilation pour chaque autre architecture, soit un total de 8 compilations.

5.12.1 Courtoisie avec les porteurs

Les porteurs ont une tâche remarquable et difficile, car ils doivent gérer un grand nombre de paquets. Idéalement, tout paquet source devrait compiler sans modification. Malheureusement, c'est rarement le cas. Cette section contient une liste d'erreurs régulièrement commises par les responsables Debian — problèmes courants qui bloquent souvent les porteurs et compliquent inutilement leur travail.

Ici, le premier et plus important point est de répondre rapidement aux rapports de bogue et problèmes soulevés par les porteurs. Traitez-les courtoisement, comme s'ils étaient coresponsables de vos paquets (ce qu'ils sont d'une certaine manière). Merci pour votre indulgence envers des rapports de bogue succincts ou peu clairs ; faites de votre mieux pour éliminer le problème.

Les problèmes les plus couramment rencontrés par les porteurs sont causés par des *erreurs d'empaquetage* dans le paquet source. Voici un pense-bête pour les points auxquels vous devez être attentif :

1. vérifiez que les champs `Build-Depends` et `Build-Depends-Indep` du fichier `debian/control` sont corrects. Le meilleur moyen de le vérifier est d'utiliser le paquet `debootstrap` pour créer un environnement `unstable chrooté` (voir *debootstrap*). Dans cet environnement `chrooté`, installez le paquet `build-essential` et tous les paquets mentionnés dans les champs `Build-Depends` ou `Build-Depends-Indep`. Ensuite, essayez de fabriquer le paquet dans cet environnement. Ces étapes peuvent être automatisées en utilisant le programme `pbuilder` fourni par le paquet de même nom (voir *pbuilder*).

   En cas de difficultés pour configurer un environnement `chrooté`, `dpkg-depcheck` pourra peut-être vous aider (voir *dpkg-depcheck*).

   Consultez la Charte Debian pour en savoir plus sur les dépendances de construction ;
2. ne choisissez pas d'autres valeurs que `all` ou `any` pour le champ architecture sans avoir de bonnes raisons. Trop souvent, les développeurs ne respectent pas les instructions de la Charte Debian. Choisir la valeur `i386` ou `amd64` est généralement incorrect ;
3. Make sure your source package is correct. Do `dpkg-source -x` *package*`.dsc` to make sure your source package unpacks properly. Then, in there, try building your package from scratch with `dpkg-buildpackage`.
4. vérifiez que les fichiers `debian/files` ou `debian/substvars` ne sont pas dans votre paquet source. Ils doivent être effacés par la cible `clean` de `debian/rules` ;
5. assurez-vous de ne pas dépendre d'éléments de configuration, ou de logiciels installés ou modifiés localement. Par exemple, vous ne devriez jamais appeler des programmes du répertoire `/usr/local/bin` ou de répertoires équivalents. Essayez de ne pas dépendre de logiciels configurés de manière spéciale. Essayez de construire votre paquet sur une autre machine, même s'il s'agit de la même architecture ;
6. ne vous appuyez pas sur une installation préexistante du paquet (un sous-cas de la remarque précédente). Il existe, bien sûr, des exceptions à cette règle, mais soyez conscient que chaque cas comme celui-ci de-

mande une mise en place (« bootstrapping ») manuelle et ne peut être automatisé par les services d'empaquetage ;

7. si possible, ne dépendez pas d'une version particulière d'un compilateur. Si vous ne pouvez pas faire autrement, assurez-vous que les dépendances de construction reflètent cette restriction, bien que vous cherchiez sûrement les problèmes, puisque certaines architectures s'uniformisent pour différents compilateurs.
8. vérifiez que le fichier `debian/rules` distingue les cibles `binary-arch` et `binary-indep` comme l'exige la Charte Debian. Vérifiez que ces cibles sont indépendantes l'une de l'autre, c'est-à-dire qu'il n'est pas nécessaire d'invoquer l'une de ces cibles avant d'invoquer l'autre. Pour vérifier cela, essayez d'exécuter `dpkg-buildpackage -B`.
9. When you can't support your package on a particular architecture, you shouldn't use the Architecture field to reflect that (it's also a pain to maintain correctly). If the package fails to build from source, you can just let it be and interested people can take a look at the build logs. If the package would actually build, the trick is to add a `Build-Depends` on `unsupported-architecture [the-not-supported-arch]`. The buildds will not build the package as the build dependencies are not fulfilled on that arch. To prevent building on 32-bits architectures, the `architecture-is-64-bit` build dependency can be used, as `architecture-is-little-endian` can be used to prevent building on big endian systems.

5.12.2 Conseils aux porteurs pour les mises à jour

Si le paquet se construit tel quel sur l'architecture visée, vous avez de la chance et votre travail est facile. Cette section s'applique dans ce cas ; elle décrit comment construire et installer correctement le paquet binaire dans l'archive Debian. Si vous devez modifier le paquet pour le rendre compilable sur la nouvelle architecture, il faudra faire une NMU sources, consultez plutôt *NMU : quand et comment*.

Pour un envoi de portage, ne faites pas de changement dans les sources. Vous n'avez pas besoin de modifier les fichiers du paquet source, y compris le fichier `debian/changelog`.

The way to invoke `dpkg-buildpackage` is as `dpkg-buildpackage -B -m`*porter-email*. Of course, set *porter-email* to your email address. This will do a binary-only build of only the architecture-dependent portions of the package, using the `binary-arch` target in `debian/rules`.

If you are working on a Debian machine for your porting efforts and you need to sign your upload locally for its acceptance in the archive, you can run `debsign` on your `.changes` file to have it signed conveniently.

5.12.2.1 Recompilation ou mise à jour indépendante binaire (`binNMU`)

Parfois, l'envoi du porteur initial pose problème, car l'environnement dans lequel le paquet a été construit n'était pas bon (bibliothèques périmées ou obsolètes, mauvais compilateur, etc.). Il se peut que vous ayez à le recompiler dans un environnement mis à jour. Cependant, dans ce cas, vous devez changer le numéro de version pour que les mauvais anciens paquets soient remplacés dans l'archive Debian (`dak` refuse d'installer un nouveau paquet s'il n'a pas un numéro de version supérieur à celui actuellement disponible).

Vous devez vous assurer que votre binNMU ne rend pas le paquet non installable. Cela peut arriver si un paquet source génère des paquets dépendants et indépendants de l'architecture qui ont des interdépendances créées par l'utilisation de la substitution de variable de dpkg `$(Source-Version)`.

Malgré la modification nécessaire du journal de modification (`changelog`), ce type de mise à jour est appelé binNMU — il n'est pas nécessaire de reconsidérer le statut des paquets binaires des autres architectures pour les marquer périmés ou à recompiler.

Ces recompilations nécessitent des numéros de version « magiques » pour que le système de maintenance de l'archive comprenne que, bien qu'il y ait une nouvelle version, il n'y a pas eu de modification des sources. Si vous ne faites pas cela correctement, les administrateurs de l'archive rejetteront votre mise à jour (car il n'y aura pas de code source associé).

Le « numéro magique » d'une NMU pour une recompilation particulière est obtenu en utilisant un suffixe ajouté au numéro de version du paquet, de la forme b*numéro*. Par exemple, si la dernière version recompilée était la version `2.9-3`, la binNMU aura pour version `2.9-3+b1`. Si la dernière version était `3.4+b1` (c'est-à-dire un paquet natif avec une précédente NMU par recompilation), la binNMU aura le numéro de version `3.4+b2`.[2]

2. Par le passé, ces NMU utilisaient le numéro de troisième niveau de la partie Debian de la révision pour indiquer l'état de leur recompilation particulière ; cependant, cette syntaxe était ambiguë pour les paquets natifs et ne permettait pas d'ordonner correctement les NMU de

De manière similaire aux envois du porteur initial, la façon correcte d'invoquer `dpkg-buildpackage` est `dpkg-buildpackage -B` pour ne construire que les parties dépendant de l'architecture du paquet.

5.12.2.2 Quand utiliser une NMU source pour un portage

Les porteurs faisant des NMU source suivent normalement les instructions de *Mises à jour indépendantes (NMU)*, tout comme les non-porteurs. Les délais d'attente sont cependant réduits, car les porteurs doivent manipuler un grand nombre de paquets. À nouveau, la situation diffère selon la distribution visée. Elle varie également si l'architecture est candidate pour la prochaine version stable ; les responsables de publication décident et annoncent quelles sont les architectures candidates.

Si vous êtes porteur et faites une NMU pour `unstable`, les instructions précédentes sont applicables à deux différences près. Tout d'abord, le temps d'attente raisonnable — délai entre le moment où vous envoyez un rapport au BTS et le moment où vous pouvez faire une NMU — est de sept jours. Ce délai peut être réduit si le problème est crucial et met l'effort de portage en difficulté : c'est à la discrétion de l'équipe de portage. (Souvenez-vous, il ne s'agit pas d'un règlement, mais de recommandations communément acceptées). Pour les envois de `stable` ou `testing`, veuillez d'abord vous coordonner avec l'équipe de publication concernée.

Deuxième différence, les porteurs faisant des NMU source doivent choisir une gravité `serious` (sérieuse) ou supérieure quand ils envoient leur rapport au BTS. Cela assure qu'un paquet source unique permet de produire un paquet binaire pour chaque architecture maintenue au moment de la sortie de la distribution. Il est très important d'avoir un paquet source et un paquet binaire pour toutes les architectures pour être conforme à plusieurs licences.

Les porteurs doivent éviter les correctifs qui contournent les bogues des actuelles versions de l'environnement de compilation, du noyau ou de la `libc`. Parfois, ces contournements ne peuvent être améliorés. Si vous devez faire quelque chose de ce genre, marquez proprement vos modifications avec des `#ifdef` et documentez votre contournement pour pouvoir le retirer une fois le problème externe disparu.

Les porteurs peuvent aussi avoir un dépôt non officiel pour publier le résultat de leur travail pendant le délai d'attente. Ainsi, d'autres personnes peuvent bénéficier du travail du porteur même pendant ce délai. Bien sûr, ces dépôts n'ont rien d'officiel, donc soyez sur vos gardes si vous les utilisez.

5.12.3 Infrastructure de portage et automatisation

Une infrastructure et plusieurs outils sont disponibles pour faciliter l'automatisation du portage des paquets. Cette section contient un bref aperçu de cette automatisation et de ces outils ; veuillez vous reporter à la documentation des paquets ou les références pour des informations complètes.

5.12.3.1 Listes de diffusion et pages web

Les pages web contenant l'état de chaque portage peuvent être trouvées à https://www.debian.org/ports/.

Chaque portage de Debian possède sa propre liste de diffusion. La liste des listes de diffusion de portage peut être trouvée à https://lists.debian.org/ports.html. Ces listes sont utilisées pour coordonner les porteurs et pour mettre en relation les utilisateurs d'un portage donné avec les porteurs.

5.12.3.2 Outils pour les porteurs

Les descriptions de plusieurs outils de portage peuvent être trouvées en *Outils de portage*.

5.12.3.3 `wanna-build`

Le système `wanna-build` est un système distribué pour la compilation d'une distribution. Il est habituellement utilisé en conjonction avec des automates de compilation faisant fonctionner le programme `buildd`. Les automates de compilation (« `build daemons` ») sont des machines « esclaves » qui récupèrent la liste des paquets à compiler du système principal `wanna-build`.

`wanna-build` n'est pas encore disponible sous forme de paquet ; pourtant, tous les efforts de portage l'utilisent pour automatiser la compilation de paquets. L'outil de compilation vraiment utilisé est dans le paquet `sbuild`, voir la

recompilation particulière, les NMU source et les NMU de sécurité sur le même paquet, elle a donc été abandonnée en faveur de cette nouvelle syntaxe.

description en *sbuild*. La version empaquetée n'est pas la même que celle utilisée sur les automates de compilation, mais suffisamment proche pour reproduire les problèmes.

La plupart des informations produites par `wanna-build`, souvent utiles pour les porteurs, sont disponibles sur la toile à l'adresse https://buildd.debian.org/. Les données disponibles sont entre autres les statistiques mises à jour chaque nuit, les informations de file d'attente et les journaux de tentatives de compilation.

Ce système est une fierté de Debian, car il a de nombreux usages potentiels. Il peut être utilisé par des groupes de développeurs indépendants pour créer différentes variantes de Debian d'intérêt général ou non (par exemple, une variante de Debian compilée avec des vérifications de limites (« `bounds checking` ») de `gcc`). Ce système permettra aussi de recompiler rapidement toute une distribution.

L'équipe de `wanna-build`, en charge des empaqueteurs (« `buildd` »), peut être contactée à l'adresse électronique `debian-wb-team@lists.debian.org`. Pour savoir qui (équipe de `wanna-build`, équipe de publication) et comment (courrier électronique, BTS) contacter, se reporter à https://lists.debian.org/debian-project/2009/03/msg00096.html.

Lors d'une demande de mise à jour indépendante binaire (`binNMU`) ou de « rendu » (« `give-back` » : nouvel essai suite à une compilation échouée), veuillez utiliser le format décrit en https://release.debian.org/wanna-build.txt.

5.12.4 Paquet *non* portable

Certains paquets ont encore des problèmes pour être construits ou pour fonctionner sur certaines architectures prises en charge par Debian, et ne peuvent pas du tout être portés, ou pas dans un délai raisonnable. Par exemple, un paquet spécifique à SVGA (disponible uniquement sur `i386` et `amd64`), ou qui utilise des fonctionnalités spécifiques au matériel non gérées sur toutes les architectures.

Pour éviter que des paquets cassés soient envoyés dans l'archive et qu'ils fassent perdre le temps des empaqueteurs (« `buildd` »), vous devez faire plusieurs choses :

— tout d'abord, assurez-vous que votre paquet *échoue* à la compilation sur les architectures qu'il ne gère pas. Il y a plusieurs façons de faire cela. Le meilleur moyen est d'avoir une petite suite de tests pendant la construction qui vérifiera la fonctionnalité et échouera si cela ne fonctionne pas. C'est de toute façon une bonne idée et empêchera (certains) des envois cassés pour toutes les architectures, cela permettra également au paquet d'être construit dès que la fonctionnalité nécessaire sera disponible.

  De plus, si vous croyez que la liste des architectures gérées est plutôt constante, vous devriez changer `any` en une liste des architectures gérées dans le fichier `debian/control`. Ainsi, la construction échouera également et l'indiquera à un lecteur humain sans vraiment essayer ;

— pour empêcher les compilateurs automatiques de tenter sans raison de construire votre paquet, il doit être inclus dans `Packages-arch-specific`, une liste utilisée par le script `wanna-build`. La version actuelle est disponible en https://wiki.debian.org/PackagesArchSpecific ; veuillez consulter le début du fichier pour savoir qui contacter pour le modifier.

Il ne suffit pas d'ajouter votre paquet à `Packages-arch-specific` sans le faire échouer lors de compilation sur les architectures non gérées : un porteur ou toute autre personne essayant de construire votre paquet peut accidentellement l'envoyer sans remarquer qu'il ne fonctionne pas. Si par le passé, certains paquets binaires ont été envoyés pour des architectures non gérées, demandez leur suppression en remplissant un bogue sur `ftp.debian.org`.

5.12.5 Paquets non libres pouvant être automatiquement construits

By default packages from the `non-free` and `non-free-firmware` sections are not built by the autobuilder network (mostly because the license of the packages could disapprove). To enable a package to be built, you need to perform the following steps :

1. vérifier s'il est légalement permis et techniquement possible de construire automatiquement le paquet ;
2. ajouter `XS-Autobuild: yes` dans la partie en-tête de `debian/control` ;
3. envoyer un courrier à `non-free@buildd.debian.org` et expliquer pourquoi le paquet peut légalement et automatiquement être construit.

5.13 Mises à jour indépendantes (NMU)

Chaque paquet est géré par un ou plusieurs responsables. Normalement, ce sont eux qui travaillent sur les paquets et s'occupent de les mettre à jour. Dans certains cas, il est utile que d'autres développeurs puissent aussi envoyer une nouvelle version, par exemple pour résoudre un bogue dans un paquet dont ils ne sont pas responsables, lorsque le responsable a besoin d'aide pour réagir aux problèmes. De tels envois sont appelés *mises à jour indépendantes (« ``Non-Maintainer Uploads`` » ou ``NMU``).*

5.13.1 NMU : quand et comment

Avant de procéder à une NMU, veuillez prendre en considération les questions suivantes.

- Have you geared the NMU towards helping the maintainer ? As there might be disagreement on the notion of whether the maintainer actually needs help or not, the DELAYED queue exists to give time to the maintainer to react and has the beneficial side-effect of allowing for independent reviews of the NMU diff.
- Does your NMU really fix bugs ? ("Bugs" means any kind of bugs, e.g. wishlist bugs for packaging a new upstream version, but care should be taken to minimize the impact to the maintainer.) Using NMUs to make changes that are likely to be non-consensual is discouraged.
- As more specific examples, the following changes are generally considered acceptable, unless there are good reasons for not following those practices in a particular package : using the latest released debhelper compatibility level ; using dh ; using 3.0 (quilt) ; using lintian-brush.
- Avez-vous laissé suffisamment de temps au responsable ? Quand le bogue a-t-il été reporté au BTS ? Être occupé pendant une semaine ou deux n'est pas exceptionnel. Le bogue est-il si grave qu'il doive être corrigé immédiatement, ou cela peut-il attendre encore quelques jours ?
- Quelle confiance avez-vous dans vos modifications ? Souvenez-vous du serment d'Hippocrate : « je m'abstiendrai de tout mal ». Il est préférable de laisser un paquet avec un bogue ouvert grave plutôt qu'appliquer un correctif non fonctionnel ou un correctif qui cache le bogue sans le résoudre. Si vous n'êtes pas absolument sûr de vous, il pourrait être judicieux de chercher des conseils autour de vous. Rappelez-vous que si quelque chose est cassé par votre NMU, de nombreuses personnes seront mécontentes.
- Avez-vous clairement manifesté votre intention de procéder à une NMU, au moins dans le BTS ? En l'absence de réaction, une bonne idée serait d'essayer de contacter le responsable par d'autres moyens (courriel aux responsables ou personnel, IRC).
- Si le responsable est habituellement actif et réactif, avez-vous tenté de le contacter ? En général il est préférable que le responsable prenne en charge lui-même un problème et qu'il lui soit offert une chance d'examiner et corriger votre correctif, car il est censé être mieux placé pour découvrir d'éventuels problèmes que vous pourriez rater. C'est souvent un gain de temps pour tout le monde si le responsable a la possibilité d'envoyer lui-même une correction.

Lors d'une NMU, vous devez d'abord vous assurer que votre intention est sans ambiguïté. Ensuite, vous devez envoyer un correctif contenant les différences entre le paquet actuel et votre proposition de NMU au BTS. Le script `nmudiff` du paquet `devscripts` pourrait être utile.

While preparing the patch, you had better be aware of any package-specific practices that the maintainer might be using. Taking them into account reduces the burden of integrating your changes into the normal package workflow and thus increases the chances that integration will happen. A good place to look for possible package-specific practices is `debian/README.source`.

À moins d'avoir une excellente raison de ne pas le faire, vous devez laisser du temps au responsable pour réagir (par exemple en envoyant le paquet dans la file `DELAYED`). Voici quelques valeurs recommandées pour les délais :

- envoi corrigeant seulement un bogue critique pour la publication ouvert il y a plus de sept jours, sans action du responsable sur le bogue pendant sept jours, et sans indication qu'un correctif est en cours : zéro jour ;
- envoi corrigeant seulement un bogue critique pour la publication ouvert il y a plus de sept jours : deux jours ;
- envoi corrigeant seulement un bogue critique pour la publication et important : cinq jours ;
- Other NMUs : 15 days

Ces délais sont simplement donnés à titre indicatifs. Dans certains cas, comme des envois corrigeant des problèmes de sécurité, ou corrigeant des bogues insignifiants qui bloquent une transition, il est préférable que le paquet atteigne `unstable` au plus tôt.

Parfois, les responsables de publication peuvent décider d'encourager des délais plus courts pour les NMU corrigeant un sous-ensemble de bogues (par exemple les bogues critiques pour la publication ouverts il y a plus de sept jours). Certains responsables s'inscrivent d'eux-mêmes à la *liste permissive de NMU (« ``Low Threshold NMU*

list ` ») <https://wiki.debian.org/LowThresholdNmu>`__, et acceptent que les NMU soient effectuées sans délai. Mais même dans ce cas, il est toujours préférable de laisser quelques jours au responsable pour réagir avant votre envoi, d'autant plus si le correctif n'était pas disponible auparavant dans le BTS, ou si vous savez que le responsable est habituellement actif.

After you upload an NMU, you are responsible for the possible problems that you might have introduced. You must keep an eye on the package (*Subscribing to package updates* is a good way to achieve this).

Il ne s'agit pas d'un permis pour faire des NMU irréfléchies. Si vous procédez à une NMU alors que le responsable est clairement actif et aurait pris en considération un correctif de façon opportune, ou si vous passez outre les recommandations de ce document, votre envoi risque d'être une cause de conflit avec le responsable. Vous devriez toujours être prêt à défendre le bien-fondé de toute NMU effectuée.

5.13.2 NMU et `debian/changelog`

Comme tout autre envoi (de paquet source), les NMU doivent comporter une nouvelle entrée dans le fichier `debian/changelog`, expliquant les modifications effectuées dans cet envoi. La première ligne de cette entrée doit signaler explicitement qu'il s'agit d'une NMU, par exemple :

```
* Non-maintainer upload.
```

La façon de numéroter les versions lors d'une NMU est différente s'il s'agit d'un paquet natif ou non.

Si le paquet est natif (sans partie révision Debian dans le numéro de version du paquet), la version doit être celle du dernier envoi du responsable, suivi de `+nmu`*X*, où *X* est un compteur commençant à `1`. Si le dernier envoi était également une NMU, le compteur devrait être augmenté. Par exemple, si la version actuelle est `1.5`, alors une NMU devrait prendre la version `1.5+nmu1`.

Si le paquet n'est pas natif, vous devriez ajouter un numéro de version mineure à la partie révision Debian du numéro de version (la partie après le dernier tiret). Ce numéro supplémentaire devrait commencer à `1`. Par exemple si la version actuelle est `1.5-2`, alors une NMU devrait prendre la version `1.5-2.1`. Si une nouvelle version amont est empaquetée lors de la NMU, la révision Debian est configurée à `0`, par exemple `1.6-0.1`.

Dans les deux cas, si le dernier envoi était également une NMU, le compteur devrait être augmenté. Par exemple, si la version actuelle est `1.5+nmu3` (un paquet natif déjà mis à jour indépendamment), la NMU devrait prendre la version `1.5+nmu4`.

Une numérotation de version spécifique est nécessaire pour éviter de perturber le travail du responsable, car l'utilisation d'un entier dans la révision Debian risque d'entrer en conflit avec un envoi déjà en préparation lors de la NMU, ou même déjà dans la file d'attente de nouveaux paquets (`NEW`), Cela présente également l'avantage d'indiquer clairement que le paquet dans l'archive n'a pas été préparé par le responsable officiel.

Lors d'un envoi de paquet vers `testing` ou `stable`, il est parfois nécessaire de créer une branche (« `fork` ») dans l'arbre de numérotation des versions. C'est par exemple le cas pour les mises à jour de sécurité. Pour cela, une version de la forme `+deb`*X*`u`*Y* devrait être utilisée, *X* étant le numéro de publication majeure et *Y* étant un compteur qui commence à 1. Par exemple, alors que `trixie` (Debian 13) est stable, une NMU de sécurité pour un paquet dont la version est `1.5-3` devrait avoir la version `1.5-3+deb13u1`, alors qu'une NMU de sécurité vers `forky` devrait prendre la version `1.5-3+deb14u1`.

5.13.3 Utilisation de la file d'attente `DELAYED/`

Attendre une réponse après avoir demandé la permission de procéder à une NMU est inefficace, car cela coûte au demandeur un changement de contexte (« `context switch` ») pour revenir sur le problème. La file d'attente `DELAYED/` (voir *Envois différés*) permet au développeur préparant une NMU d'accomplir toutes les tâches nécessaire en même temps. Par exemple, plutôt que dire au responsable que vous allez envoyer le nouveau paquet dans sept jours, vous devriez envoyer le paquet vers `DELAYED/7` et dire au responsable qu'il a sept jours pour réagir. Pendant ce temps, le responsable peut vous demander de retarder un peu plus votre envoi, ou l'annuler.

You can cancel your upload using *dcut*. In case you uploaded `foo_1.2-1.1_all.changes` to a `DELAYED` queue, you can run `dcut cancel foo_1.2-1.1_all.changes` to cancel your upload. The `.changes` file does not need to be present locally as you instruct *dcut* to upload a command file removing a remote filename. The `.changes` file name is the same that you used when uploading.

La file d'attente DELAYED ne devrait pas être utilisée pour augmenter la pression sur le responsable. Notamment, il est important d'être disponible pour annuler ou retarder l'envoi avant la fin du délai, car le responsable ne peut pas le faire lui-même.

Si vous procédez à une NMU vers DELAYED et que le responsable envoie son paquet avant la fin du délai, votre envoi sera rejeté, car une nouvelle version sera alors disponible dans l'archive. Dans l'idéal, le responsable se chargera d'intégrer votre proposition (ou du moins une solution pour le problème en question) dans son envoi.

5.13.4 NMU d'un point de vue du responsable

Quand quelqu'un réalise une NMU sur votre paquet, c'est pour vous aider à le garder en bon état. Cela permet aux utilisateurs d'obtenir un paquet corrigé au plus vite. Vous pouvez envisager de proposer à l'auteur de la NMU de devenir coresponsable du paquet. Recevoir une NMU sur un paquet n'est pas une mauvaise chose : cela signifie simplement que le paquet est suffisamment intéressant pour que d'autres personnes veuillent travailler dessus.

Pour prendre en compte une NMU, intégrez ses modifications et l'entrée de journal de modification (changelog) lors de votre envoi suivant. Si vous ne prenez pas en compte la NMU en conservant l'entrée de changelog correspondante, le bogue restera fermé dans le BTS, mais sera listé comme affectant votre version du paquet.

Notez que si vous avez besoin d'annuler une NMU qui a empaqueté une nouvelle version amont, il est recommandé d'utiliser une version amont factice telle que *ACTUEL*+really*ANCIEN* jusqu'à ce que quelqu'un puisse téléverser à nouveau la dernière version. Plus d'informations sont disponibles dans https://www.debian.org/doc/debian-policy/ch-controlfields.html#epochs-should-be-used-sparingly.

Note that the easiest way to both check if your package has been NMUed, and also automatically download and commit the changes into a git-buildpackage maintained git repository is to run `gbp import-dsc --verbose --pristine-tar apt:<package>/sid`. This example command assumes you are working on the `debian/latest` branch preparing the next upload to Debian unstable, and it assumes your `apt` has the `deb-src` line active for Debian unstable.

5.13.5 Mise à jour indépendante source (NMU) vs binaire (binNMU)

Le nom complet pour une NMU est *mise à jour indépendante source (« ``source NMU`` »)*. Il en existe aussi d'un autre type, appelé *mise à jour indépendante binaire (« ``binary-only NMU`` » ou « ``binNMU`` »)*. Une binNMU est aussi un paquet envoyé par quelqu'un d'autre que le responsable du paquet. Cependant, seul le paquet binaire est mis à jour.

Lorsqu'une bibliothèque (ou toute autre dépendance) est mise à jour, les paquets l'utilisant risquent de devoir être reconstruits. Puisque le code source n'a pas besoin d'être modifié, le même paquet source est utilisé.

Les binNMU sont généralement déclenchées sur les empaqueteurs (« buildd ») par wanna-build. Une entrée est ajoutée à debian/changelog expliquant pourquoi un envoi était requis et le numéro de version est augmenté tel que décrit en *Recompilation ou mise à jour indépendante binaire (binNMU)*. Cette entrée ne devrait pas être gardée lors de l'envoi suivant.

Les empaqueteurs (« buildd ») envoient les paquets de leur architecture comme des mises à jour binaires. Au sens strict, ce sont des binNMU. Cependant, elles ne sont généralement pas appelées NMU, et aucune entrée n'est ajoutée à debian/changelog.

5.13.6 NMU et envoi de QA

Les NMU sont des envois effectués par quelqu'un d'autre que le responsable attitré. Il existe un autre type d'envoi où le paquet mis à jour ne vous appartient pas : les envois de QA, qui sont des envois pour les paquets orphelins.

Les envois de QA ressemblent beaucoup à des envois normaux de responsable : ils peuvent corriger quelque chose, même un problème mineur ; la numérotation de version est normale, et il n'est pas nécessaire d'utiliser d'envoi retardé. La différence est que vous ne faites pas partie des responsables (Maintainer ou Uploader) du paquet. Ainsi, l'entrée du journal de modification (changelog) d'un envoi de QA commence par la ligne :

```
* QA upload.
```

Si vous voulez faire une NMU, et que le responsable ne semble pas actif, il est judicieux de vérifier le paquet pour voir s'il est orphelin (cette information est disponible sur la page du système de suivi relative au paquet).

Lors d'un premier envoi de QA sur un paquet orphelin, veuillez positionner le responsable à « Debian QA Group <packages@qa.debian.org> ». La liste actuelle des paquets orphelins dont le responsable n'a pas encore été modifié est disponible en https://qa.debian.org/orphaned.html.

Plutôt que faire un envoi de QA, vous pouvez envisager l'adoption du paquet en devenant son responsable. Vous n'avez besoin de la permission de personne pour adopter un paquet orphelin, il suffit de vous configurer comme responsable et d'envoyer la nouvelle version (voir *Adoption de paquet*).

5.13.7 NMU et envoi d'équipe

Parfois, vous corrigez ou envoyez un paquet, car vous êtes membre d'une équipe de responsables (qui utilise une liste de diffusion comme responsable (Maintainer ou Uploader), voir *Maintenance collective*), mais vous ne voulez pas vous ajouter comme coresponsable (Uploaders), car vous n'avez pas l'intention de participer régulièrement à ce paquet. Si cela est conforme avec la politique de votre équipe, vous pouvez procéder à un envoi normal sans être listé parmi les responsables (Maintainer ou Uploader). Dans ce cas, vous devriez commencer l'entrée du journal de modification (changelog) par la ligne suivante :

```
* Team upload.
```

5.14 Sauvetage de paquets

Le sauvetage de paquet est le processus permettant de sauver un paquet qui, tout en n'étant pas officiellement orphelin, semble peu ou complètement non entretenu. C'est une procédure moindre et plus rapide que de déclarer un paquet orphelin officiellement à l'aide des moyens de l'équipe MIA. Sauver un paquet n'est pas destiné à remplacer la gestion MIA et diffère en ce qu'elle n'implique aucunement l'activité complète de responsable. Cela gère plutôt la transition de responsabilité pour un seul paquet, n'affectant pas tout autre paquet, affiliation à Debian ou droits de téléversement (lorsque pertinents).

Il est a remarquer que le processus est seulement destiné à prendre fermement la responsabilité. Ne démarrez pas le processus de sauvetage de paquet sans la ferme intention d'entretenir le paquet pendant une période prolongée. Si vous voulez corriger certaines choses sans devenir responsable, vous devez utiliser le processus NMU, même si le paquet est susceptible d'être sauvé. Le processus NMU est expliqué dans *Mises à jour indépendantes (NMU)*.

Une autre chose à se souvenir est qu'il n'est pas acceptable de détourner d'autres paquets. S'il est suivi, le processus de sauvetage vous aide à être sûr que votre initiative n'est pas un détournement, mais une procédure (officielle) de sauvetage, et vous pourrez réfuter toute allégation de détournement en faisant référence à ce processus. Grâce à ce processus, les nouveaux contributeurs ne devraient pas être effrayés de s'accaparer des paquets délaissés ou complètement abandonnés.

Le processus se déroule en deux phases. Premièrement, vous déterminez si le paquet en question est *éligible* au processus de sauvetage. Seulement après que l'éligibilité a été déterminée, vous pouvez démarrer la deuxième phase, le sauvetage *effectif* du paquet.

Pour des informations complémentaires, des principes et des FAQ sur le sauvetage de paquet, veuillez consulter la page Salvaging Packages sur le wiki de Debian.

5.14.1 Quand un paquet devient-il éligible au sauvetage de paquet ?

Un paquet devient éligible au sauvetage lorsqu'il est abandonné par le responsable actuel. Pour savoir qu'un paquet est réellement abandonné par son responsable, les indicateurs suivants donnent une idée sommaire sur ce qu'il faut rechercher :

- les NMU, spécialement s'il y a eu plusieurs NMU consécutifs ;
- les bogues concernant le paquet, sans réponse du responsable ;
- l'amont a publié plusieurs versions du paquet, mais malgré un dépôt de bogue en faisant la demande, elle n'a pas été empaquetée ;
- Il existe des problèmes de QA avec le paquet.

You will have to use your judgement as to whether a given combination of factors constitutes neglect ; in case the maintainer disagrees they have only to say so (see below). If you're not sure about your judgement or simply want to be on the safe side, there is a more precise (and conservative) set of conditions in the Package Salvaging wiki

page. These conditions represent a current Debian consensus on salvaging criteria. In any case you should explain your reasons for thinking the package is neglected when you file an Intent to Salvage bug later.

5.14.2 Comment sauver un paquet ?

Si, et *seulement* si, un paquet a été déterminé comme éligible au sauvetage, n'importe quel responsable potentiel peut démarrer la procédure de sauvetage qui suit.

1. Ouvrez un bogue avec une sévérité « important » pour le paquet en question, exprimant votre intention de prendre la responsabilité du paquet. Pour cela, le titre du rapport doit débuter par `ITS: package-name`[3]. Sinon vous pouvez proposer de prendre la coresponsabilité du paquet. Lorsque vous ouvrez le bogue, vous devez informer tous les responsables, téléverseurs et si approprié, l'équipe d'empaquetage, explicitement en les ajoutant dans `X-Debbugs-CC`. De plus, si le ou les responsables semblent être généralement inactifs, veuillez informer l'équipe MIA en ajoutant également `mia@qa.debian.org` à `X-Debbugs-CC`. De même que l'expression explicite de l'intention de sauvetage, veuillez aussi prendre le temps de documenter votre évaluation de l'éligibilité dans le rapport de bogue, par exemple, en listant les critères utilisés et en ajoutant quelques données pour faciliter l'évaluation de la situation par d'autres ;
2. À ce stade, vous devez attendre au cas où des objections seraient soulevées. Le responsable, n'importe quel téléverseur actuel ou n'importe quel membre de l'équipe d'empaquetage concernée par le paquet en question peut objecter publiquement en réponse au bogue déposé sous `21 jours`, et cela met fin au processus de sauvetage.

   The current maintainers may also agree to your intent to salvage by filing a (signed) public response to the bug. They might propose that you become a co-maintainer instead of the sole maintainer. On team maintained packages, a member of the associated team can accept your salvaging proposal by sending out a signed agreement notice to the ITS bug, alternatively inviting you to become a new co-maintainer of the package. The team may require you to keep the package under the team's umbrella, but then may ask or invite you to join the team. In any of these cases where you have received the OK to proceed, you can upload the new package immediately as the new (co-)maintainer, without the need to utilise the `DELAYED` queue as described in the next step.
3. Après 21 jours d'attente, si aucune réponse n'a été envoyée au bogue par le responsable, un téléverseur ou une équipe, vous pouvez envoyez la nouvelle publication du paquet dans la file d'attente `DELAYED` avec un délai minimal de `sept jours`. Vous devrez clore le bogue dans le changelog et vous devez aussi envoyer un nmudiff au bogue veillant que des copies soient envoyées au responsable et à tous les téléverseurs (y compris les équipes) du paquet en les mettant en `CC` (Copie conforme) dans le courriel au BTS.

   Durant le temps d'attente de la file `DELAYED`, les responsables peuvent accepter le sauvetage, faire un envoi eux-mêmes ou (demander à) annuler l'envoi. Ces deux dernières actions stopperont le processus de sauvetage, mais les responsables doivent répondre au bogue de sauvetage avec plus d'informations sur le pourquoi.

5.15 Maintenance collective

« Maintenance collective » est un terme décrivant le partage des devoirs de la maintenance d'un paquet Debian par plusieurs personnes. Cette collaboration est presque toujours une bonne idée, car il en résulte généralement une meilleure qualité et un temps de correction de bogues plus court. Il est fortement recommandé que les paquets de priorité `standard` ou qui font partie de la base aient des coresponsables.

Habituellement, il y a un responsable principal et un ou plusieurs coresponsables. Le responsable principal est la personne dont le nom est indiqué dans le champ `Maintainer` du fichier `debian/control`. Les coresponsables sont tous les autres responsables, normalement listés dans le champ `Uploaders` du fichier `debian/control`.

Dans sa forme la plus simple, ajouter un nouveau coresponsable est assez facile :

— configurer le coresponsable avec droit d'accès aux sources à partir desquelles vous construisez le paquet. En général, cela implique que vous utilisez un système de gestion de versions en réseau, tel que `Git`. Salsa (voir *salsa.debian.org : dépôts Git et plateforme de développement collaborative*) fournit des dépôts Git parmi d'autres outils collaboratifs.
— ajouter les nom et adresse correctes du coresponsable au champ `Uploaders` dans le premier paragraphe du fichier `debian/control` ;

3. ITS est une abréviation pour *"Intend to Salvage"*.

```
Uploaders: John Buzz <jbuzz@debian.org>, Adam Rex <arex@debian.org>
```

— The co-maintainers should subscribe themselves to the appropriate source package (see *Subscribing to package updates*).

Une autre forme de maintenance collective est une maintenance en équipe, recommandée si vous gérez plusieurs paquets avec le même groupe de développeurs. Dans ce cas, les champs de responsable (`Maintainer` et `Uploaders`) de chaque paquet doivent être gérés avec attention. Il est conseillé de choisir entre les deux possibilités suivantes :

1. placer un membre de l'équipe comme responsable principal du paquet dans le champ `Maintainer`. En `Uploaders`, placer l'adresse de la liste de diffusion et les membres de l'équipe qui s'occupent du paquet ;
2. placer l'adresse de la liste de diffusion dans le champ `Maintainer`. En `Uploaders`, placer les membres de l'équipe qui s'occupent du paquet. Dans ce cas, vous devez vous assurer que la liste de diffusion peut recevoir les rapports de bogue sans interaction humaine (modération pour les non inscrits par exemple).

En tout cas, il faut éviter de placer automatiquement tous les membres de l'équipe dans le champ `Uploaders`. Cela encombre la vue d'ensemble des paquets d'un développeur (voir *Vue d'ensemble des paquets d'un développeur*) avec des paquets dont il ne s'occupe pas vraiment et donne la fausse impression d'un bon suivi. De même, les membres de l'équipe n'ont pas besoin de s'ajouter dans le champ `Uploaders` pour faire un envoi ponctuel, ils peuvent le faire en « envoi d'équipe » (voir *NMU et envoi d'équipe*). En revanche, c'est une mauvaise idée de garder un paquet avec seulement l'adresse de la liste de diffusion dans le champ `Maintainer` et sans `Uploaders`.

5.16 La distribution `testing`

5.16.1 Bases

Les paquets sont habituellement installés dans la distribution `testing` après avoir subi suffisamment de tests dans `unstable`.

Ils doivent être en synchronisation pour toutes les architectures et ne doivent pas avoir de dépendances qui les rendraient non installables ; ils doivent également être exempts de bogue critique pour la publication (« `release-critical` ») au moment où ils sont installés dans `testing`. Ainsi, `testing` devrait toujours être prête à devenir une version candidate pour la publication. Veuillez voir ci-dessous pour les détails.

5.16.2 Mise à jour depuis `unstable`

Les scripts de mise à jour de la distribution `testing` sont exécutés deux fois par jour, juste après l'installation des paquets mis à jour ; ces scripts sont appelés `britney`. Ils fabriquent les fichiers `Packages` pour la distribution `testing`, mais ils le font d'une manière intelligente pour éviter toute incohérence et essayer de n'utiliser que des paquets sans bogue.

L'inclusion d'un paquet d'`unstable` est soumise aux conditions suivantes :

— The package must have been available in `unstable` for a certain number of days, see *Sélection de l'urgence du téléversement*. Please note that the urgency is sticky, meaning that the highest urgency uploaded since the previous `testing` transition is taken into account ;
— il ne doit pas introduire de nouveau bogue critique pour la publication (« `RC bug` » affectant la version disponible dans `unstable`, mais pas celle de `testing`) ;
— il doit être disponible pour toutes les architectures pour lesquelles il a déjà été construit dans `unstable`. *Utilitaire dak ls* permet de vérifier cette information ;
— il ne doit pas casser les dépendances d'un paquet déjà disponible dans `testing` ;
— les paquets dont il dépend doivent soit être déjà disponibles dans `testing`, soit être acceptés dans `testing` au même moment (et ils doivent remplir tous les critères nécessaires) ;
— l'état du projet. C'est-à-dire que les transitions automatiques sont arrêtées pendant le *freeze* (gel) de la distribution `testing`.

Pour savoir si un paquet a progressé ou non dans `testing`, veuillez voir la sortie du script de `testing` sur la *page web de la distribution `testing* <https://www.debian.org/devel/testing>`__ ou utilisez le programme `grep-excuses` du paquet `devscripts`. Pour rester informé de la progression de vos paquets dans `testing`, vous pouvez facilement le mettre dans une crontab 5.

Le fichier `update_excuses` ne donne pas toujours la raison précise pour laquelle un paquet est refusé ; il peut être nécessaire de la chercher soi-même en regardant ce qui serait cassé avec l'inclusion du paquet. La page web de la distribution testing donne plus d'informations sur les problèmes courants pouvant occasionner cela.

Parfois, certains paquets n'entrent jamais dans `testing` parce que le jeu des interrelations est trop compliqué et ne peut être résolu par le script. Voir ci-dessous pour des détails.

Des analyses de dépendances plus avancées sont présentées sur https://release.debian.org/migration/ — mais, attention, cette page affiche également des dépendances de construction qui ne sont pas prises en compte par `britney`.

5.16.2.1 Désynchronisation

Pour le script de migration de `testing`, périmé signifie : il y a différentes versions dans `unstable` pour les architectures publiées (sauf les architectures dans outofsync_arches ; outofsync_arches est la liste des architectures qui sont abandonnées (dans `britney.py`), mais actuellement la liste est vide). Le caractère périmé n'a absolument rien à voir avec les architectures de ce paquet dans `testing`.

Considérons cet exemple :

	alpha	arm
testing	1	-
unstable	1	2

Le paquet est désynchronisé pour `alpha` dans `unstable` et n'entrera pas dans `testing`. Supprimer le paquet de `testing` n'aiderait en rien, le paquet serait toujours désynchronisé pour `alpha` et ne se propagerait pas dans `testing`.

Cependant, si `ftp-master` supprime un paquet d'`unstable` (ici pour `arm`) :

	alpha	arm	hurd-i386
testing	1	1	-
unstable	2	-	1

Dans ce cas, le paquet est synchronisé pour toutes les architectures de version dans `unstable` (et l'architecture supplémentaire `hurd-i386` ne compte pas, car ce n'est pas une architecture de publication).

Quelquefois, la question est soulevée de savoir s'il est possible de permettre à des paquets de passer dans `testing` alors qu'ils ne sont pas encore construits pour toutes les architectures : non. Simplement non. (Excepté si vous êtes responsable de glibc ou équivalent).

5.16.2.2 Suppression de `testing`

Parfois, un paquet est supprimé pour permettre l'inclusion d'un autre paquet : cela ne se produit que pour permettre à un *autre* paquet d'entrer, ce dernier doit être prêt pour tous les autres critères. Par exemple, si un paquet `a` ne peut pas être installé avec la nouvelle version de `b`, alors `a` peut être supprimé pour permettre l'entrée de `b`.

Bien sûr, il existe une autre raison pour supprimer un paquet de `testing` : le paquet est trop bogué (et avoir un seul bogue critique pour la publication est suffisant pour être dans cet état).

De plus, si un paquet a été supprimé d'`unstable` et que plus un seul paquet de `testing` n'en dépend, il sera alors automatiquement supprimé.

5.16.2.3 Dépendances circulaires

Une situation mal gérée par `britney` est si un paquet `a` dépend de la nouvelle version d'un paquet `b` et vice versa.

Voici un exemple :

	testing	unstable
a	1 ; dépend de : b=1	2 ; dépend de : b=2
b	2 ; dépend de : a=1	2 ; dépend de : a=2

Aucun des paquets `a` et `b` ne sera considéré pour une mise à jour.

Actuellement, cela nécessite un coup de pouce manuel de l'équipe de publication. Veuillez les contacter à l'adresse `debian-release@lists.debian.org` si cela se produit pour l'un de vos paquets.

5.16.2.4 Influence d'un paquet dans `testing`

En général, rien n'est indiqué par l'état d'un paquet dans `testing` pour une transition de la versions suivante d'`unstable` vers `testing`, avec deux exceptions : la première si le nombre de bogues critiques pour la publication diminue, il peut entrer même s'il comporte encore des bogues critiques. La seconde exception, c'est si la version du paquet dans `testing` est désynchronisée sur les différentes architectures : alors une architecture pourrait seulement être mise à niveau vers la version du paquet source ; néanmoins, cela peut se produire seulement si auparavant le passage du paquet a été forcé, si l'architecture est dans outofsync_arches ou s'il n'y avait aucun paquet binaire de cette architecture présent dans `unstable` durant la migration de `testing`.

En résumé, cela signifie : la seule influence qu'un paquet de `testing` a sur la nouvelle version du même paquet est que la nouvelle version peut entrer plus facilement.

5.16.2.5 Détails

Si vous êtes intéressés par les détails, voici quelques informations sur le fonctionnement de `britney`.

Les paquets sont examinés pour savoir si ce sont des candidats valables. Cela génère les dispenses de mise à jour (« `update excuses` »). Les raisons habituelles pour lesquelles un paquet n'est pas considéré sont la jeunesse du paquet, le nombre de bogues critiques pour la publication et la désynchronisation pour certaines architectures. Pour cette partie de `britney`, les responsables de publication ont des marteaux de toute taille, appelés coups de pouce (« hints », voir ci-dessous) pour forcer `britney` à examiner un paquet.

Maintenant, la partie la plus complexe se produit : `britney` tente de mettre à jour `testing` avec des candidats valables. Pour ce faire, `britney` essaye d'ajouter chaque candidat valable à la distribution `testing`. Si le nombre de paquets non installables dans `testing` n'augmente pas, le paquet est accepté. À partir de là, le paquet accepté est considéré comme partie de `testing`, de telle sorte qu'il sera considéré dans les tests suivants d'installabilité. Avant et après cette partie, certains coups de pouce (« hints ») de l'équipe de publication sont traités.

Pour obtenir plus de précisions, vous pouvez consulter https://release.debian.org/britney/update_output/.

Les coups de pouce (« hints ») sont disponibles sur https://release.debian.org/britney/hints/, qui contient aussi une description. Avec les coups de pouce, l'équipe en charge de la publication de Debian peut bloquer ou débloquer des paquets, faciliter ou forcer le passage de paquets dans `testing`, enlever des paquets de `testing`, approuver des envois vers les *Mises à jour directes dans testing* ou outrepasser l'urgence.

5.16.3 Mises à jour directes dans `testing`

La distribution `testing` est remplie de paquets en provenance d'`unstable` selon des règles expliquées ci-dessus. Cependant, dans certains cas, il est nécessaire d'envoyer des paquets construits seulement pour `testing`. Pour cela, vous pouvez envoyer vos paquets vers `testing-proposed-updates`.

Souvenez-vous que les paquets envoyés là ne sont pas traités automatiquement, ils doivent passer entre les mains des responsables de distribution. Vous devez donc avoir une bonne raison pour les y envoyer. Pour savoir ce que représente une bonne raison aux yeux des responsables de publication, vous devriez lire les instructions qu'ils envoient régulièrement sur `debian-devel-announce@lists.debian.org`.

Vous ne devriez pas envoyer de paquet à `testing-proposed-updates` quand vous pouvez le mettre à jour par `unstable`. Si vous ne pouvez faire autrement (par exemple, parce que vous avez une nouvelle version de développement dans `unstable`), vous pouvez utiliser cette fonction. Même si un paquet est gelé, des mises à jour par `unstable` sont possibles si l'envoi dans `unstable` ne crée pas de nouvelles dépendances.

Les numéros de version sont habituellement choisis en ajoutant `+debXuY`, où *X* est le numéro de publication majeure de Debian et *Y* est un compteur démarrant à `1`, par exemple, `1:2.4.3-4+deb13u1`.

Veuillez vous assurer de n'avoir oublié aucun des éléments suivants lors de votre envoi :

- vérifiez que le paquet doit vraiment aller dans `testing-proposed-updates`, et ne peut pas passer par `unstable` ;
- vérifiez de n'avoir intégré qu'un minimum de changements ;
- vérifiez d'avoir ajouté une explication appropriée dans le journal de modification (`changelog`) ;

— vérifiez d'avoir ajouté les *Noms de code des distributions* de testing (par exemple, `forky`) comme distribution cible ;
— vérifiez d'avoir construit et testé votre paquet dans `testing`, et non dans `unstable` ;
— vérifiez que le numéro de version est plus élevé que les versions de `testing` et de `testing-proposed-updates`, et moins élevé que celui d'`unstable` ;
— demandez l'autorisation d'envoi aux responsables de publication ;
— après l'envoi et la construction réussie sur toutes les plates-formes, contactez l'équipe de publication à `debian-release@lists.debian.org` et demandez-leur d'approuver votre envoi.

5.16.4 Foire aux questions

5.16.4.1 Quels sont les bogues bloquant l'intégration dans la version stable et comment sont-ils pris en compte ?

Tous les bogues de gravité assez élevée sont par défaut considérés comme bloquant l'intégration du paquet dans la version stable ; actuellement, ce sont les bogues `critical` (critique), `grave` (grave) et `serious` (sérieux).

Certains bogues sont supposés avoir un impact sur la probabilité d'un paquet a être diffusé dans la version `stable` de Debian : en général, si un paquet a des bogues bloquants, il n'ira pas dans `testing`, et par conséquent, ne pourra pas être diffusé dans `stable`.

Le décompte des bogues d'`unstable` inclut tous les bogues critiques pour la publication marqués pour s'appliquer à des combinaisons de *paquet*/*version* disponibles dans `unstable` pour une architecture concernée par la publication. Le décompte des bogues de `testing` est défini de façon analogue.

5.16.4.2 Comment l'installation d'un paquet dans `testing` peut-elle casser d'autres paquets ?

La structure des archives de la distribution est faite de telle façon qu'elles ne peuvent contenir qu'une version d'un paquet ; un paquet est défini par son nom. Donc, quand le paquet source `acmefoo` est installé dans `testing` avec ses paquets binaires `acme-foo-bin`, `acme-bar-bin`, `libacme-foo1` et `libacme-foo-dev`, l'ancienne version est supprimée.

Cependant, l'ancienne version pouvait fournir à un paquet binaire un vieux `soname` d'une bibliothèque, comme `libacme-foo0`. Supprimer l'ancien `acmefoo` va supprimer `libacme-foo0`, ce qui va casser tout paquet qui en dépend.

Évidemment, cela touche principalement des paquets qui fournissent des jeux variables de paquets binaires pour différentes versions (principalement des bibliothèques). Cependant, cela va aussi concerner des paquets sur lesquels une dépendance versionnée du type ==, <=, ou << a été déclarée.

Quand le jeu de paquets binaires fournis par un paquet source change de cette façon, tous les paquets qui dépendent des anciens binaires doivent être mis à jour pour dépendre plutôt de la nouvelle version. Comme l'installation d'un tel paquet source dans `testing` casse tous les paquets qui en dépendent dans `testing`, une attention particulière doit y être portée : tous les paquets en dépendant doivent être mis à jour et prêts à être installés eux-mêmes pour ne pas casser et, une fois que tout est prêt, une intervention manuelle des responsables de publication est normalement requise.

Si vous avez des problèmes avec des groupes compliqués de paquets comme cela, demandez de l'aide sur `debian-devel@lists.debian.org` ou `debian-release@lists.debian.org`.

5.17 The Stable backports archive

5.17.1 Bases

Once a package reaches the `testing` distribution, it is possible for anyone with upload rights in Debian (see below about this) to build and upload a backport of that package to `stable-backports`, to allow easy installation of the version from `testing` onto a system that is tracking the `stable` distribution.

One should not upload a version of a package to `stable-backports` until the matching version has already reached the `testing` archive.

5.17.2 Exception to the testing-first rule

The only exception to the above rule, is when there's an important security fix that deserves a quick upload : in such a case, there is no need to delay an upload of the security fix to the `stable-backports` archive. However, it is strongly advised that the package is first fixed in `unstable` before uploading a fix to the `stable-backports` archive.

5.17.3 Who can maintain packages in the stable-backports archive ?

It is not necessarily up to the original package maintainer to maintain the `stable-backports` version of the package. Anyone can do it, and one doesn't even need approval from the original maintainer to do so. It is however good practice to first get in touch with the original maintainer of the package before attempting to start the maintenance of a package in `stable-backports`. The maintainer can, if they wish, decide to maintain the backport themselves, or help you doing so. It is not uncommon, for example, to apply a patch to the unstable version of a package, to facilitate its backporting.

5.17.4 When can one start uploading to stable-backports ?

The new `stable-backports` is created before the freeze of the next `stable` suite. However, it is not allowed to upload there until the very end of the freeze cycle. The `stable-backports` archive is usually opened a few weeks before the final release of the next `stable` suite, but it doesn't make sense to upload until the release has actually happened.

5.17.5 How long must a package be maintained when uploaded to stable-backports ?

The `stable-backports` archive is maintained for bugs and security issues during the whole life-cycle of the Debian `stable` suite. Therefore, an upload to `stable-backports`, implies a willingness to maintain the backported package for the duration of the `stable` suite, which can be expected to be about 3 years from its initial release.

The person uploading to backports is also supposed to maintain the backported packages for security during the lifetime of `stable`.

It is to be noted that the `stable-backports` isn't part of the LTS or ELTS effort. The `stable-backports` FTP masters will close the `stable-backports` repositories for uploads once `stable` reaches end-of-life (ie : when `stable` becomes maintained by the LTS team only). Therefore there won't be any maintenance of packages from `stable-backports` after the official end of life of the `stable` suite, as uploads will not be accepted.

5.17.6 How often shall one upload to stable-backports ?

The packages in backports are supposed to follow the developments that are happening in Testing. Therefore, it is expected that any significant update in `testing` should trigger an upload into `stable-backports`, until the new `stable` is released. However, please do not backport minor version changes without user visible changes or bugfixes.

5.17.7 How can one learn more about backporting ?

You can learn more about how to contribute directly on the backport web site.

It is also recommended to read the Frequently Asked Questions (FAQ).

CHAPITRE 6

Meilleures pratiques d'empaquetage

La qualité de Debian est largement due à la Charte Debian qui définit explicitement les exigences de base que tous les paquets Debian doivent satisfaire. Cependant, il existe également une expérience générale partagée qui va bien au delà de la Charte Debian et constitue une somme d'années d'expérience dans l'empaquetage. De nombreux contributeurs talentueux ont créé d'excellents outils qui peuvent vous aider, en tant que mainteneur Debian, à créer et maintenir des paquets d'excellente qualité.

Ce chapitre rassemble les meilleures pratiques pour les responsables Debian. La majorité de son contenu est constituée de recommandations plus que d'obligations. Il s'agit essentiellement d'informations subjectives, d'avis et de pointeurs, rassemblés par les développeurs Debian. Il est conseillé d'y choisir ce qui vous convient le mieux.

6.1 Meilleures pratiques pour `debian/rules`

Les recommandations qui suivent s'appliquent au fichier `debian/rules`. Comme ce fichier contrôle le processus de construction des paquets et fait le choix des fichiers qui entreront dans ce paquet (directement ou indirectement), il s'agit du fichier dont les responsables s'occupent généralement le plus.

6.1.1 Scripts d'assistance

La motivation pour utiliser des scripts d'assistance dans `debian/rules` est de permettre aux mainteneurs de définir puis utiliser une logique commune pour de nombreux paquets. Si on prend par exemple l'installation d'entrées de menu, il est nécessaire de placer le fichier dans `/usr/share/menu` (ou `/usr/lib/menu` pour les fichiers de menu exécutables, si besoin), puis d'ajouter des commandes aux scripts des responsables pour ajouter ou enlever les entrées de menu. Comme cette action est commune à de très nombreux paquets, pourquoi faudrait-il que chaque responsable doivent réécrire ses propres méthodes, bogues compris ? De plus, si jamais le répertoire des menus venait à changer, chaque paquet devrait être modifié.

Les scripts d'assistance s'occupent de ce type de tâche. À condition de suivre les conventions utilisées par le script d'assistance, celui-ci s'occupe de tous les détails. Les modifications dans la Charte peuvent alors être implémentées dans le script d'assistance et les paquets n'ont plus qu'à être reconstruits sans autre modification.

Aperçu des outils du responsable Debian contient un certain nombre d'assistants variés. Le système le plus répandu et (de l'avis général) le plus adapté est `debhelper`. Des systèmes antérieurs, tels que `debmake`, étaient monolithiques : ils ne permettaient pas de choisir quelle partie de l'assistant serait utile et obligeaient à se servir de l'ensemble de l'assistant. A contrario, `debhelper` est constitué d'un grand nombre de petits programmes `dh_*` différents. Par exemple, `dh_installman` installe et compresse les pages de manuel, `dh_installmenu` installe les fichiers de menu, et ainsi de suite. En conséquence, il offre la possibilité d'utiliser certains des scripts d'assistance tout en conservant des commandes manuelles dans `debian/rules`.

You can get started with `debhelper` by reading debhelper(7) and looking at the examples that come with the package. `dh_make`, from the `dh-make` package (see *dh-make*), can be used to convert a vanilla source package to a `debhelper`ized package. This shortcut, though, should not convince you that you do not need to bother understanding the individual `dh_*` helpers. If you are going to use a helper, you do need to take the time to learn to use that helper, to learn its expectations and behavior.

6.1.2 Paquets binaires multiples

Un paquet source unique construira souvent plusieurs paquets binaires, soit pour fournir plusieurs variantes du même logiciel (par exemple, le paquet source `vim`) ou pour créer plusieurs petits paquets à la place d'un seul grand paquet (par exemple, l'utilisateur peut installer uniquement le sous-ensemble nécessaire et ainsi économiser de l'espace disque, voir par exemple le paquet source `libxml2`).

Le second cas est simple à gérer dans le fichier `debian/rules`. Il suffit de déplacer les fichiers nécessaires depuis le répertoire de construction vers l'arborescence temporaire du paquet. Cela peut se faire avec les commandes `install` ou `dh_install` du paquet `debhelper`. Veillez alors à contrôler les différentes permutations des paquets, afin de pouvoir indiquer les dépendances inter-paquets appropriées dans `debian/control`.

Le premier cas est plus délicat à gérer, car il implique des recompilations multiples du même logiciel avec différentes options de configuration. Le paquet source `vim` en est un exemple, l'ensemble des actions dans le fichier `debian/rules` étant géré manuellement.

6.2 Meilleures pratiques pour `debian/control`

Les conseils qui suivent sont destinés au fichier `debian/control`. Ils complètent la Charte Debian concernant les descriptions de paquets.

La description d'un paquet telle que définie par le champ correspondant du fichier `control`, comprend à la fois le résumé et la description longue du paquet. *Conseils généraux pour les descriptions de paquets* donne des indications communes à ces deux parties, *Résumé, ou description courte, d'un paquet* donne des indications spécifiques pour le résumé et *Description longue* donne des indications pour la description.

6.2.1 The package name

The package name :

- Must be consistent with widely established conventions, e.g.
  - C and C++ libraries are typically prefixed with lib
  - for many other languages, package names are prefixed or suffixed with the language name (the actual convention depends on the language).
- Should not be a common, unqualified word. In particular, words that represent a whole class of applications (e.g. reader, browser) should be reserved for virtual packages.
- Should ideally be at least 4 characters long, unless the upstream name is shorter than that *and* the project is already widely recognized by that name (e.g. fzf).

6.2.2 Conseils généraux pour les descriptions de paquets

La description d'un paquet doit être écrite pour son utilisateur moyen, c'est-à-dire la personne qui utilisera et tirera profit du paquet. Par exemple, les paquets de développement sont destinés aux développeurs et leur description peut comporter des détails techniques alors que les applications d'usage plus général, telles que les éditeurs, doivent avoir une description accessible à tout utilisateur.

Un examen général des descriptions de paquets tend à montrer que la plupart d'entre elles ont une orientation fortement technique et ne sont donc pas destinées à l'utilisateur moyen. Sauf dans le cas de paquets destinés à des spécialistes, cela doit être considéré comme un problème.

Une recommandation pour rester accessible à tout utilisateur est d'éviter l'utilisation de jargon. Il est déconseillé de faire référence à des applications ou environnements qui pourraient être inconnus de l'utilisateur : parler de GNOME ou KDE est correct, car la plupart des utilisateurs sont familiers avec ces termes, mais parler de GTK ne l'est pas. Il est préférable de supposer que le lecteur n'aura pas de connaissance du sujet et, si des termes techniques doivent être utilisés, ils doivent être expliqués.

Il est conseillé de rester objectif. Les descriptions de paquets ne sont pas une plaquette publicitaire, quelles que soient vos opinions personnelles. Le lecteur peut très bien ne pas avoir les mêmes centres d'intérêt que vous.

Les références aux noms d'autres logiciels, de protocoles, normes ou spécifications doivent utiliser leur forme canonique si elle existe. Par exemple, utilisez « X Window System », « X11 » ou « X », mais pas « X Windows », « X-Windows », ou « X Window ». Utilisez « GTK » et non « GTK+ » ou « gtk », « GNOME » et non « Gnome », « PostScript » et non « Postscript » ou « postscript ».

Si vous rencontrez des difficultés pour écrire la description d'un paquet, vous pouvez demander de l'aide ou une relecture sur `debian-l10n-english@lists.debian.org`.

6.2.3 Résumé, ou description courte, d'un paquet

La Charte indique que la ligne de résumé (la description courte) doit être concise, ne doit pas répéter le nom du paquet, mais doit être informative.

La description courte est une expression qui décrit le paquet, pas une phrase complète, donc les conventions de ponctuation sont inappropriées : pas besoin de commencer par une majuscule ou de finir par un point. Elle devrait éviter également la présence d'article défini ou indéfini — « `a` », « `an` », ou « `the` ». Par exemple :

```
Package: libeg0
Description: exemplification support library
```

Techniquement, c'est une phrase nominale sans article, par opposition à une phrase verbale. Une bonne vérification est de pouvoir remplacer le *nom* du paquet et son *résumé* dans la phrase :

Le paquet *nom* fournit {un,une,le,la,l',du,de la} *résumé* (« `the package` *nom* `provides {a,an,the,some}` *résumé* »).

Les ensembles de paquets peuvent utiliser un schéma alternatif qui divise la description courte en deux parties, la première une description de l'ensemble et la seconde un résumé du rôle du paquet dans l'ensemble :

```
Package: eg-tools
Description: simple exemplification system (utilities)

Package: eg-doc
Description: simple exemplification system - documentation
```

Ces descriptions courtes suivent un modèle modifié. Quand un paquet « *nom* » possède une description courte « *ensemble* (*rôle*) » ou « *ensemble* - *rôle* », les éléments peuvent être placés dans la phrase :

Le paquet *nom* fournit {un,une,le,la,l'} *rôle* pour {le,la,l'} *ensemble* (« `the package` *nom* `provides {a,an,the}` *rôle* `for the` *ensemble* »).

6.2.4 Description longue

La description longue est l'information principale disponible pour les utilisateurs avant d'installer un paquet. Elle devrait fournir toutes les informations nécessaires pour déterminer si le paquet doit être installé. Elle complète le résumé qui est donc supposé avoir été lu précédemment.

La description longue est constituée de phrases complètes.

Le premier paragraphe de cette description devrait tenter de répondre aux questions suivantes : « Que fait ce paquet ? », « Dans quelle tâche aidera-t-il l'utilisateur ? ». Il est important que cette description se fasse de la manière la moins technique possible, sauf si le public auquel est destiné le paquet est par définition technique.

Les descriptions longues de paquets associés, par exemple construits à partir du même paquet source, peuvent partager des paragraphes afin d'améliorer la cohérence et de réduire la charge de travail pour les traducteurs, mais il faut qu'il y ait au moins un paragraphe distinct qui décrive le rôle spécifique du paquet.

Les paragraphes suivants devraient répondre aux questions : « Pourquoi, en tant qu'utilisateur, ai-je besoin de ce paquet ? », « Quelles autres fonctionnalités ce paquet apporte-t-il ? », « Quelles fonctionnalités et défauts comporte-t-il par rapport à d'autres paquets (par exemple, « si vous avez besoin de X, utilisez plutôt Y ») ? », « Ce paquet

est-il lié à d'autres paquets d'une manière non gérée par le système de gestion des paquets (par exemple, « cela est le client destiné au serveur toto ») ? ».

Veillez à éviter les erreurs d'orthographe et de grammaire. Vérifiez l'orthographe avec un outil adapté. Les deux programmes `ispell` et `aspell` comportent un mode spécial permettant de contrôler un fichier `debian/control` files :

```
ispell -d american -g debian/control
```

```
aspell -d en -D -c debian/control
```

Les utilisateurs attendent en général des descriptions de paquets les réponses aux questions suivantes.

- Que fait ce paquet ? S'il s'agit d'un additif à un autre paquet, la description de cet autre paquet doit y être reprise.
- Pourquoi ai-je besoin de ce paquet ? Cela est lié à la remarque précédente, de manière différente (cela est un agent utilisateur pour le courrier électronique, avec une interface rapide et pratique vers PGP, LDAP et IMAP et les fonctionnalités X, Y ou Z).
- Si ce paquet ne doit pas être installé seul, mais est installé par un autre paquet, cela devrait être mentionné.
- Si le paquet est `expérimental` ou ne doit pas être utilisé pour toute autre raison et que d'autres paquets doivent être utilisés à la place, cela doit également être mentionné.
- En quoi ce paquet diffère-t-il de ses concurrents ? Est-il une meilleure implémentation ? A-t-il plus de fonctionnalités ? Des fonctionnalités différentes ? Pourquoi devrais-je choisir ce paquet ?

6.2.5 Page d'accueil amont

Il est recommandé d'ajouter l'URL d'accès à la page d'accueil du paquet dans le champ `Homepage` de la section `Source` du fichier `debian/control`. L'ajout de cette information à la description même du paquet est une pratique considérée obsolète.

6.2.6 Emplacement du système de gestion de versions

Des champs supplémentaires permettent d'indiquer l'emplacement du système de gestion de versions dans `debian/control`.

6.2.6.1 Vcs-Browser

La valeur de ce champ doit être une URL `https://` pointant sur la copie navigable par le web du dépôt de gestion de versions utilisé pour la maintenance du paquet, s'il est disponible.

Cette information est destinée à l'utilisateur final qui voudrait parcourir le travail en cours sur le paquet (par exemple à la recherche d'un correctif qui corrige un bogue marqué `pending` (en attente) dans le système de suivi des bogues.

6.2.6.2 Vcs-*

Value of this field should be a string identifying unequivocally the location of the Version Control System repository used to maintain the given package, if available. * identifies the Version Control System ; currently the following systems are supported by the package tracking system : `arch`, `bzr` (Bazaar), `cvs`, `darcs`, `git`, `hg` (Mercurial), `mtn` (Monotone), `svn` (Subversion).

Cette information est destinée aux utilisateurs qui ont une connaissance suffisante du système de gestion de versions et qui veulent construire une version à jour du paquet depuis les sources du système de suivi. Une autre utilisation possible de cette information pourrait être la construction automatique de la dernière version, dans le système de suivi, d'un paquet donné. À cet effet, l'emplacement pointé devrait éviter d'être lié à une version spécifique et pointer vers la branche principale de développement (pour les systèmes qui ont un tel concept). De plus, l'emplacement indiqué doit être accessible à l'utilisateur final, par exemple en indiquant une adresse d'accès anonyme au dépôt, plutôt qu'une version accessible par SSH.

L'exemple qui suit montre une instance de ce champ pour un dépôt Git du paquet `vim`. Veuillez noter que l'URL a la forme `https://` (au lieu de `ssh://`). Une utilisation des champs `Vcs-Browser` et `Homepage`, décrits précédemment, est aussi indiquée.

```
Source: vim
<snip>
Vcs-Git: https://salsa.debian.org/vim-team/vim.git
Vcs-Browser: https://salsa.debian.org/vim-team/vim
Homepage: https://www.vim.org
```

Maintaining the packaging in a version control system, and setting a Vcs-* header is good practice and makes it easier for others to contribute changes.

Almost all packages in Debian that use a version control system use Git; if you create a new package, using Git is a good idea simply because it's the system that contributors will be familiar with.

DEP-14 defines a common layout for Debian packages.

6.3 Meilleures pratiques pour `debian/changelog`

Les indications de cette partie complètent la Charte Debian pour ce qui concerne les fichiers de journaux des modifications (« `changelog` »).

6.3.1 Entrées de journalisation utiles

Le journal des modifications (« `changelog` ») présente uniquement les changements intervenus dans la version courante. Il est suggéré de mettre l'accent sur les modifications visibles ou affectant potentiellement les utilisateurs, réalisées depuis la version précédente.

Il est conseillé de mettre l'accent sur *ce* qui a été modifié, plutôt que comment, par qui et quand elle a été réalisée. Cela dit, il est conseillé, par courtoisie, d'indiquer les auteurs qui ont apporté une aide significative à la maintenance du paquet (par exemple lorsque ces personnes ont envoyé des correctifs).

Il n'est pas indispensable d'indiquer les détails des modifications triviales. Il est également possible de grouper plusieurs modifications sur une même entrée. Cependant, évitez une documentation trop concise pour les modifications majeures. Il est particulièrement conseillé d'être très clair sur les modifications qui affectent le comportement du programme. Pour des explications plus détaillées, vous pouvez aussi utiliser le fichier `README.Debian`.

Utilisez un anglais simple que la majorité des lecteurs puissent comprendre. Évitez les abréviations et le jargon technique lorsque des modifications permettent la clôture de bogues. Cela est vrai notamment quand vous pensez que les utilisateurs qui les ont envoyés n'ont pas de connaissances techniques importantes. Une formulation polie est à préférer et la vulgarité à prohiber.

Il est parfois souhaitable de faire précéder les entrées du journal des modifications par les noms des fichiers modifiés. Cependant, rien n'oblige à mentionner le moindre fichier modifié, notamment si la modification est simple ou répétitive. L'utilisation de caractères joker est possible.

Ne faites pas de suppositions lorsque vous faites référence à un bogue. Indiquez quel était le problème, comment il a été corrigé et ajoutez la chaîne closes : #nnnnn. Veuillez consulter *Fermeture des rapports de bogue lors des mises à jour* pour plus d'informations.

6.3.2 Sélection de l'urgence du téléversement

The release team have indicated that they expect most uploads to `unstable` to use **urgency=medium**. That is, you should choose **urgency=medium** unless there is some particular reason for the upload to migrate to `testing` more quickly or slowly (see also *Mise à jour depuis unstable*). For example, you might select **urgency=low** if the changes since the last upload are large and might be disruptive in unanticipated ways.

The delays are currently 2, 5 or 10 days, depending on the urgency (high, medium or low). The actual numbers are actually controlled by the britney configuration which also includes accelerated migrations when Autopkgtest passes.

6.3.3 Idées reçues sur les entrées de journalisation

Les entrées de journal des modifications ne devraient **pas** documenter les points spécifiques de la réalisation du paquet (« si vous cherchez le fichier toto.conf, il est situé dans /etc/titi »), car les administrateurs et les utilisateurs sont censés avoir l'habitude de la façon dont ces aspects sont traités sur un système Debian. Pensez, par contre, à documenter la modification de l'emplacement d'un fichier de configuration.

Les seuls bogues fermés par une entrée de journal de modifications devraient être ceux qui sont corrigés par la version correspondante du paquet. Fermer de cette manière des bogues qui n'ont aucun rapport avec la nouvelle version est considéré comme une mauvaise habitude. Veuillez consulter *Fermeture des rapports de bogue lors des mises à jour*.

Les entrées du journal des modifications ne devraient **pas** être utilisées pour des discussions variées avec les émetteurs des rapports de bogue (par exemple : « je n'ai pas d'erreur de segmentation quand je lance toto avec l'option titi, merci d'envoyer plus d'informations »). De même, les considérations générales sur la vie, l'univers et le reste (« désolé, cet envoi m'a pris plus longtemps que prévu, mais j'avais un rhume ») ou encore des demandes d'aide (« la liste de bogues de ce paquet est très longue, merci de me donner un coup de main ») sont à éviter. Ces mentions ne seront généralement pas remarquées par leur public potentiel et peuvent ennuyer les personnes qui cherchent à lire les modifications concrètes du paquet. Voir *Réponses aux bogues* pour plus d'informations sur l'utilisation du système de gestion des bogues.

Une tradition assez ancienne veut que les bogues corrigés dans les NMU soient pris en compte dans la première entrée du journal des modifications d'une nouvelle version construite par le responsable. Depuis l'existence du suivi de version pour le système de gestion de bogues, cette pratique est obsolète à condition de conserver les entrées du journal des modifications des NMU. Il est éventuellement possible de simplement mentionner les NMU dans votre propre entrée de journal des modifications.

6.3.4 Erreurs usuelles dans les entrées de journalisation

Les exemples suivants sont des erreurs usuelles ou des exemples de mauvaises pratiques dans le style des entrées de journaux de modifications (NdT : le texte est volontairement laissé non traduit).

```
* Fixed all outstanding bugs.
```

Cela ne donne évidemment aucune indication au lecteur.

```
* Applied patch from Jane Random.
```

Que faisait ce correctif ?

```
* Late night install target overhaul.
```

Qu'est-ce que cela a amené ? Est-ce que la mention du fait que cela ait été fait tard la nuit doit nous alerter sur la probable mauvaise qualité du code ?

```
* Fix vsync fw glitch w/ ancient CRTs.
```

Trop d'acronymes (que signifie « `fw` », « `firmware` » ?), et la nature réelle du problème n'est pas claire, ou comment il a été corrigé.

```
* This is not a bug, closes: #nnnnnn.
```

Il est inutile de faire un nouvel envoi de paquet pour envoyer cette information. Il suffit simplement d'utiliser le système de suivi des bogues. De plus, aucune explication n'est donnée sur les raisons qui font que le problème n'est pas un bogue.

```
* Has been fixed for ages, but I forgot to close; closes: #54321.
```

Si, pour une raison donnée, vous avez omis de mentionner un numéro de bogue dans une entrée précédente, ce n'est pas grave : il suffit de clore le bogue normalement dans le système de suivi des bogues. Il est inutile de changer

le journal des modifications si on suppose que les explications sur la correction du bogue sont dans le bogue lui-même (cela s'applique également au suivi des bogues des auteurs amont : il est inutile de suivre, dans le journal des modifications, les bogues qu'ils ont corrigés depuis longtemps).

```
* Closes: #12345, #12346, #15432
```

Où est la description ? Si vous ne trouvez pas de message suffisamment explicite, vous pouvez au moins utiliser le titre du rapport de bogue.

6.3.5 Complément des journaux de modifications dans les fichiers `NEWS.Debian`

Les nouvelles importantes sur les modifications survenues dans un paquet peuvent être placées dans des fichiers `NEWS.Debian`. Ces nouvelles seront affichées par des outils tels que `apt-listchanges` avant tout le reste des modifications. Cette méthode est à privilégier pour diffuser aux utilisateurs d'un paquet les modifications importantes qu'il subit. Il est préférable de l'utiliser plutôt que des notes `debconf`, car ce système permet de revenir lire les fichiers `NEWS.Debian` après l'installation. Il est également préférable de faire la liste des modifications majeures dans `README.Debian`, car un utilisateur peut assez facilement ne pas remarquer l'affichage d'une note `debconf` (Ndt : a contrario, les fichiers `NEWS.Debian` ne peuvent être traduits).

Le format de ce fichier est analogue à un journal de modifications Debian, mais n'utilise pas d'astérisque et chaque nouveau message utilise un paragraphe complet plutôt que les mentions succinctes qui seraient utilisées dans le journal des modifications. Il est conseillé de traiter le fichier avec `dpkg-parsechangelog`, ce qui permet d'en vérifier la mise en forme, car il ne sera pas automatiquement modifié pendant la construction du paquet, contrairement au journal des modifications. Voici un exemple de fichier `NEWS.Debian` réel :

```
cron (3.0pl1-74) unstable; urgency=low

    The checksecurity script is no longer included with the cron package:
    it now has its own package, checksecurity. If you liked the
    functionality provided with that script, please install the new
    package.

 -- Steve Greenland <stevegr@debian.org>  Sat,  6 Sep 2003 17:15:03 -0500
```

Le fichier `NEWS.Debian` est installé sous le nom `/usr/share/doc/`*paquet*`/NEWS.Debian.gz`. Il est compressé et porte toujours ce nom même pour les paquets Debian natifs. Si vous utilisez `debhelper`, `dh_installchangelogs` installera les fichiers `debian/NEWS` automatiquement.

À la différence des journaux de modifications, vous n'avez pas besoin de mettre `NEWS.Debian` à jour à chaque nouvelle version. Il est suffisant de le mettre à jour quand une information importante doit être diffusée aux utilisateurs. Si vous n'avez pas d'information importante à diffuser, il n'est pas nécessaire d'utiliser un fichier `NEWS.Debian` avec le paquet. Pas de nouvelles, bonnes nouvelles !

6.4 Best practices around security

When an upstream publishes a cryptographic signature for every new release, you should setup `uscan` to automatically verify the latter. For details, refer to the section on `Upstream source location: "debian/watch"` in the Debian Policy Manual.

https://wiki.debian.org/Hardening has suggestions on how to build security hardened executables.

6.5 Meilleures pratiques pour les scripts du responsable

Les scripts du responsable (« `maintainer scripts` ») sont les fichiers `debian/postinst`, `debian/preinst`, `debian/prerm` et `debian/postrm`. Ces scripts peuvent prendre en charge les phases d'installation ou de désinstallation non automatiquement gérées par la création ou la suppression de fichiers ou de répertoires. Les instructions qui suivent complètent celles de la Charte Debian.

6.5.1 Generic guidelines

Les scripts du responsable doivent être idempotents. Cela signifie que vous devez vous assurer que rien de grave ne se produit si un script est lancé deux fois au lieu d'une.

L'entrée et la sortie standard peuvent être redirigées (par exemple dans des tuyaux, ou « `pipes` ») pour des besoins de journalisation. Il est donc recommandé qu'ils ne soient pas dépendants d'un terminal.

Toute interaction avec l'utilisateur doit être limitée au maximum. Lorsqu'elle est nécessaire, vous devriez utiliser le paquet `debconf` comme interface. Veuillez noter que l'interaction doit impérativement se faire à l'étape `configure` du script `postinst`.

Les scripts du responsable doivent rester aussi simples que possible et utiliser de préférence des scripts shell POSIX stricts. Veuillez noter que si vous avez besoin de spécificités de Bash, vous devez utiliser une ligne « `shebang` » pour Bash. Les scripts POSIX ou Bash sont encouragés par rapport aux scripts Perl, car `debhelper` peut alors y ajouter des fonctions.

Si vous modifiez les scripts du responsable, veillez à vérifier la suppression du paquet, la double installation et la purge. Vérifiez qu'un paquet purgé est entièrement éliminé, c'est-à-dire que les fichiers créés, directement ou indirectement dans les scripts du responsable, sont tous supprimés.

Pour vérifier l'existence d'une commande, vous devriez utiliser quelque chose comme :

```
if command -v install-docs > /dev/null; then ...
```

Vous pouvez utiliser cette fonction pour rechercher dans `$PATH` une commande donnée, passée en paramètre. Elle renvoie « `true` » (zéro) si la commande est trouvée et « `false` » dans le cas contraire. Il s'agit de la méthode la plus portable, car `command -v` est une commande interne pour plusieurs interpréteurs de commande et elle est définie dans POSIX.

L'utilisation de `which` est une alternative acceptable dans la mesure où elle est fournie par le paquet `debianutils` qui est requis.

6.5.2 Expected behavior of complex postinst scripts

`postinst` scripts must normally not fail, except for very rare cases. These are cases where postinst must succeed :

on fresh installs :

- the service configuration is straightforward and can be reasonably expected to work as-is in typical Debian setups
- the service has no external dependencies (e.g. a database which may not yet be configured, or unreachable at install time)

on upgrades :

- `postinst` can verify with high confidence (which may not always be feasible) that the service was running prior to the restart
- the service has no external dependencies or `postinst` can verify that they are functional
- the service configuration has not changed in backwards incompatible ways between the old and new package versions

6.6 Gestion de la configuration avec `debconf`

`debconf` est un système de gestion de configuration utilisable par les divers scripts des paquets (`postinst` notamment) pour interagir avec l'utilisateur sur des choix à opérer pour la configuration du paquet. Les interactions directes avec l'utilisateur doivent maintenant être évitées en faveur de `debconf`, notamment pour permettre des installations non interactives dans le futur.

Debconf is a great tool but it is often poorly used. Many common mistakes are listed in the debconf-devel(7) man page. It is something that you must read if you decide to use debconf. Also, we document some best practices here.

Les conseils qui suivent comportent des indications sur le style d'écriture et la typographie, des considérations générales sur l'utilisation de `debconf` ainsi que des recommandations plus spécifiques relatives à certaines parties de la distribution (le système d'installation notamment).

6.6.1 Proscrire les abus de debconf

Depuis que debconf est apparu dans Debian, il a été tellement utilisé que de nombreuses critiques ont été émises à l'encontre de la distribution Debian pour abus d'utilisation de debconf, avec la nécessité de répondre à un nombre très important de questions avant d'avoir un quelconque outil installé.

Les notes d'utilisation doivent être réservées à leur emplacement naturel : le fichier NEWS.Debian ou README.Debian. N'utilisez les notes que pour des points importants qui peuvent directement concerner l'utilisabilité du paquet. Les notes interrompent l'installation tant qu'elles ne sont pas confirmées et elles peuvent conduire à des envois de courriers électroniques aux utilisateurs.

Choisissez soigneusement les priorités des questions posées dans les scripts du responsable. Veuillez consulter la page de manuel debconf-devel 7 pour plus de détails sur les priorités. La plupart des questions devraient utiliser les priorités intermédiaire (medium) ou basse (low).

6.6.2 Recommandations générales pour les auteurs et les traducteurs

6.6.2.1 Utilisation d'un anglais correct

La plupart des responsables de paquet Debian ne sont pas anglophones. Il n'est donc pas nécessairement facile pour eux d'écrire des écrans correctement.

Pensez à utiliser (voire abuser de) la liste debian-l10n-english@lists.debian.org. Faites relire vos écrans.

Des écrans mal écrits fournissent une image négative de votre paquet, de votre travail ou même de Debian en général.

Évitez autant que possible le jargon technique. Si certains termes vous sont familiers, ils peuvent être incompréhensibles à d'autres. Si vous ne pouvez les éviter, tentez de les expliquer (avec la description étendue). Dans ce cas, tentez de faire la part des choses entre simplicité et verbosité.

6.6.2.2 Courtoisie avec les traducteurs

Les écrans debconf peuvent être traduits. Les paquets debconf et po-debconf fournissent un cadre simple permettant la traduction des écrans par des équipes de traduction ou des traducteurs isolés.

Utilisez des écrans permettant l'utilisation de gettext. Installez le paquet po-debconf sur votre machine de développement et lisez sa documentation (man po-debconf est un bon début).

Évitez de changer les écrans trop souvent. Les modifications de texte ont une incidence sur le travail des traducteurs dont les traductions vont devenir approximatives (« fuzzy »). Une chaîne de caractères devient approximative quand la version originale a été modifiée depuis la traduction, demandant ainsi une mise à jour par un traducteur pour être utilisable. Si les modifications sont mineures, la traduction originale est conservée dans le fichier PO, mais marquée fuzzy.

Si vous prévoyez de modifier les écrans d'origine, veuillez utiliser le système de notification podebconf-report-po, fourni avec le paquet po-debconf, pour contacter les traducteurs. La plupart des traducteurs sont réactifs, et inclure leur mise à jour en même temps que les modifications des écrans d'origine vous évitera des envois ultérieurs pour mettre à jour des traductions. Si vous utilisez des écrans se servant de gettext, le nom et l'adresse électronique des traducteurs sont mentionnés dans les en-têtes des fichiers PO et seront utilisés par podebconf-report-po.

Une façon recommandée de se servir de cet utilitaire est :

```
cd debian/po && podebconf-report-po --call --languageteam --withtranslators --
↪deadline="+10 days"
```

Cette commande synchronisera d'abord les fichiers PO et POT de debian/po avec les fichiers d'écrans listés en debian/po/POTFILES.in. Ensuite, elle déclenchera un appel à de nouvelles traductions sur la liste de diffusion debian-i18n@lists.debian.org. Enfin, elle déclenchera un appel à mise à jour de traduction aux équipes de traductions (indiquées dans le champ Language-Team de chaque fichier PO) ainsi qu'au dernier traducteur (indiqué en Last-translator).

La mention d'une date limite aux traducteurs est toujours appréciée, pour leur permettre d'organiser leur travail. Veuillez ne pas oublier que certaines équipes ont formalisé leur processus de traduction et révision de telle sorte

qu'un délai inférieur à dix jours n'est pas considéré comme raisonnable. Un délai plus court met trop de pression sur les équipes de traduction et ne devrait être réservé qu'aux modifications mineures.

Dans le doute, vous pouvez également contacter l'équipe de traduction d'une langue donnée (debian-l10n-xxxxx@lists.debian.org) ou la liste de diffusion `debian-i18n@lists.debian.org`.

6.6.2.3 Correction (« unfuzzy ») des traductions pour des erreurs typographiques ou de frappe

Lorsque le texte d'un écran `debconf` est corrigé et que vous avez la **certitude** que la modification n'affecte **pas** les traductions, pensez aux traducteurs et rendez leur traductions à nouveau complètes (« *unfuzzy* »).

Si cela n'est pas fait, l'ensemble de l'écran `debconf` ne sera plus traduit tant qu'un traducteur n'aura pas envoyé de mise à jour.

Pour rendre les traductions à nouveau complètes (« *unfuzzy* »), vous pouvez utiliser `msguntypot` (du paquet `po4a`).

1. Recréez les fichiers POT et PO.

```
debconf-updatepo
```

2. Faites une copie du fichier POT.

```
cp templates.pot templates.pot.orig
```

3. Faites une copie de tous les fichiers PO.

```
mkdir po_fridge; cp *.po po_fridge
```

4. Modifier les fichiers d'écrans `debconf` (`templates`) pour corriger les fautes de frappe.
5. Recréez les fichiers POT et PO (de nouveau).

```
debconf-updatepo
```

   À ce moment-là, la correction a marqué certaines chaînes approximatives, et ce changement est malheureusement la seule modification entre les fichiers PO du répertoire et ceux de po_orig. Voici comment corriger cela.

6. Abandonnez les traductions approximatives, récupérer celles du répertoire originel.

```
cp po_fridge/*.po .
```

7. Fusionnez manuellement les fichiers PO avec le nouveau fichier POT, en prenant en compte le fait que les étiquettes « fuzzy » sont inutiles.

```
msguntypot -o templates.pot.orig -n templates.pot *.po
```

8. Nettoyage.

```
rm -rf templates.pot.orig po_fridge
```

6.6.2.4 Proscrire toute supposition sur les interfaces utilisateurs

Les textes des écrans ne devraient pas faire référence aux éléments disponibles sur certaines interfaces de `debconf`. Des phrases telles que « *If you answer Yes* » ne signifient rien avec les interfaces graphiques où des boutons radio sont utilisés pour les questions booléennes.

Les écrans de type `string` ne devraient pas faire référence aux valeurs par défaut dans leur description. Cela est tout d'abord redondant avec les valeurs visibles par les utilisateurs. Mais également, les valeurs présentées par défaut peuvent être différentes du choix du responsable (par exemple, lorsque la base de données de `debconf` a été pré-renseignée).

De manière plus générale, évitez de faire référence à des actions particulières des utilisateurs et donnez simplement des faits.

6.6.2.5 Proscrire l'utilisation de la première personne

Vous devriez éviter l'utilisation de la première personne (« *I will do this...* » ou « *We recommend...* »). L'ordinateur n'est pas une personne et les écrans de `debconf` ne parlent pas au nom des développeurs de Debian. Vous devriez utiliser des constructions neutres. Pour les personnes familières de la publication scientifique, il suffit en général d'adopter le style d'écriture qui y est utilisé. Tentez cependant d'utiliser la forme active si possible. Par exemple : « *Enable this if...* » au lieu de « *This can be enabled if...* ».

6.6.2.6 Neutralité en genre

Pour se conformer à notre engagement de diversité et équité, veuillez utiliser des constructions de genre neutre dans votre écriture. Cela signifie éviter les pronoms tels que il ou elle lors d'une référence à un rôle (tel que « responsable ») dont le genre est inconnu. À la place, vous devriez utiliser une forme plurielle (They singulier).

6.6.3 Définition des champs de modèles (« `templates` »).

This part gives some information which is mostly taken from the debconf-devel(7) manual page.

6.6.3.1 `Type`

`string`

offre un champ de saisie où l'utilisateur peut entrer n'importe quelle chaîne de caractères.

`password`

demande un mot de passe. Ce champ est à utiliser avec précaution, car le mot de passe saisi sera conservé dans la base de données de `debconf`. Il est conseillé d'effacer cette valeur de la base de données dès que possible.

`boolean`

offre un choix du type « vrai » ou « faux ». N'oubliez pas, c'est bien un choix « vrai » ou « faux », **pas « oui » ou « non »**...

`select`

offre le choix entre différentes valeurs. Les choix doivent être indiqués dans un champ appelé « `Choices` ». Les différentes valeurs doivent être séparées par des virgules et des espaces, comme ceci : « `Choices: yes, no, maybe` ».

Si les choix sont traduisibles, le champ « `Choices` » peut être marqué traduisible en utilisant « `__Choices` ». Les deux tirets bas permettent à chaque choix de devenir une chaîne différente proposée à la traduction.

Le système `po-debconf` offre également la possibilité intéressante de ne marquer que **certains** choix traduisibles. Par exemple :

```
Template: foo/bar
Type: Select
#flag:translate:3
__Choices: PAL, SECAM, Other
_Description: TV standard:
 Please choose the TV standard used in your country.
```

Dans cet exemple, seule la chaîne « `Other` » est traduisible, alors que les autres sont des acronymes qui ne devraient pas êtres traduits. Seul « `Other` » sera inclus dans les fichiers PO et POT.

The debconf templates flag system offers many such possibilities. The po-debconf(7) manual page lists all these possibilities.

multiselect

Like the select data type, except the user can choose any number of items from the choices list (or choose none of them).

note

plus qu'une vraie question, ce type indique une note affichée aux utilisateurs. Elle doit être réservée à des informations importantes que l'utilisateur doit absolument voir, car `debconf` fera tout pour s'assurer qu'elle soit visible, en interrompant l'installation jusqu'à ce qu'une touche soit appuyée, voire en envoyant la note par courrier électronique dans certains cas.

text

ce type est maintenant obsolète : il ne faut pas l'utiliser.

error

ce type permet de gérer des messages d'erreur. Il est analogue au type `note`. Les interfaces utilisateur peuvent le présenter différemment (par exemple l'interface `cdebconf` dessine un écran à fond rouge au lieu de l'écran bleu habituel).

Il est recommandé d'utiliser ce type pour tout message qui requiert l'attention de l'utilisateur pour procéder à une correction, quelle qu'elle soit.

6.6.3.2 `Description` : descriptions courte et étendue

Les descriptions de modèle comportent deux parties : la partie courte et la partie étendue. La partie courte est celle qui est placée sur la ligne `Description` du modèle.

La partie courte doit rester courte (une cinquantaine de caractères) afin d'être gérée par la majorité des interfaces de `debconf`. La garder courte facilite également le travail des traducteurs, car les traductions sont souvent plus longues que les textes originaux.

La description courte doit être autonome. Certaines interfaces ne montrent pas la description longue par défaut ou ne la montrent que si l'utilisateur le demande explicitement. Il est ainsi déconseillé d'utiliser des phrases comme « `What do you want to do?` » (« Que voulez vous faire ? »)

La description courte ne doit pas nécessairement être une phrase entière. C'est une façon de la garder courte et efficace.

La partie longue ne doit pas répéter la partie courte. Si vous ne trouvez pas de partie longue appropriée, réfléchissez un peu plus. Demandez dans debian-devel. Demandez de l'aide. Prenez un cours d'écriture ! La description longue est importante. Si, malgré tout cela, vous ne trouvez rien d'intéressant à ajouter, laissez-la vide.

La partie longue doit utiliser des phrases complètes. Les paragraphes doivent rester courts pour améliorer la lisibilité. Ne placez pas deux idées différentes dans le même paragraphe, mais séparez-les en deux paragraphes.

Don't be too verbose. Users tend to ignore too long screens. 20 lines are by experience a border you shouldn't cross, because that means that in the classical dialog interface, people will need to scroll, and lot of people just don't do that.

La partie longue de la description ne devrait **jamais** comporter de question.

Les parties qui suivent donnent des recommandations spécifiques pour certains types de modèles (`string`, `boolean`, etc.).

6.6.3.3 `Choices`

Ce champ doit être utilisé pour les types `select` et `multiselect`. Il contient les choix proposés aux utilisateurs. Ces choix doivent être séparés par des virgules.

6.6.3.4 `Default`

Ce champ optionnel contient la réponse par défaut pour les modèles `string`, `select` et `multiselect`. Dans ce dernier cas, il peut comporter une liste de choix multiples, séparés par des virgules.

6.6.4 Guide de style spécifique à certains champs de modèle

6.6.4.1 Champ `Type`

Pas d'indication particulière si ce n'est choisir le type adapté en se référant à la section précédente.

6.6.4.2 Champ `Description`

Vous trouverez ici des instructions particulières pour l'écriture du champ `Description` (parties courte et longue) selon le type de modèle.

Modèles `string` et `password`

— La description courte est une invite et **pas** un titre. Il faut éviter la forme interrogative (« `IP Address?` ») au profit d'une invite ouverte (« `Adresse IP :` »). L'utilisation d'un deux-points final est recommandée.
— La partie longue complète la partie courte. Il est conseillé d'y expliquer ce qui est demandé, plutôt que de répéter la même demande. Utilisez des phrases complètes. Un style d'écriture abrégé est déconseillé.

Modèles `boolean`

— La partie courte devrait utiliser la forme interrogative, rester courte et se terminer par un point d'interrogation. Un style abrégé est toléré et même encouragé si la question est complexe (les traductions sont souvent plus longues que les versions originales).
— Il est important de ne pas faire référence aux spécificités de certaines interfaces. Une erreur classique est d'utiliser une construction comme « `If you answer Yes...` » (« Si vous répondez Oui... »).

Modèles `select` et `multiselect`

— La description courte est une invite et **pas** un titre. N'utilisez **pas** de constructions comme « `Please choose...` » (« Veuillez choisir... »). Les utilisateurs sont suffisamment intelligents pour comprendre qu'il est nécessaire de choisir quelque chose.
— La description longue complète la partie courte. Elle peut faire référence aux choix disponibles. Elle peut aussi indiquer que l'utilisateur peut sélectionner plus d'un choix parmi ceux disponibles, pour les modèles `multiselect` (bien que l'interface rende en général cela tout à fait clair).

Modèles `note`

— La description courte doit être considérée comme un **titre**.
— La partie longue est ce qui sera affiché comme description plus détaillée de la note. Il est déconseillé d'y utiliser un style abrégé.
— **Do not abuse debconf.** Notes are the most common way to abuse debconf. As written in the debconf-devel(7) manual page : it's best to use them only for warning about very serious problems. The `NEWS.Debian` or `README.Debian` files are the appropriate location for a lot of notes. If, by reading this, you consider converting your Note type templates to entries in `NEWS.Debian` or `README.Debian`, please consider keeping existing translations for the future.

6.6.4.3 Champ `Choices`

Si les choix changent souvent, il est suggéré d'utiliser l'astuce « `__Choices` ». Avec ce format, chaque choix sera une chaîne différente proposée à la traduction, ce qui facilite grandement le travail des traducteurs.

6.6.4.4 Champ `Default`

Si la valeur par défaut pour un modèle choisi est susceptible de dépendre de la langue de l'utilisateur (par exemple si le choix concerne la langue), pensez à utiliser l'astuce _Default, documentée dans la page de manuel po-debconf 7.

Ce champ spécial permet aux traducteurs de mettre le choix le plus adapté à leur langue, qui deviendra le choix par défaut quand cette langue est utilisée, alors que le choix par défaut que vous avez mentionné sera utilisé en anglais.

N'utilisez pas de champ `Default` vide. Si vous ne souhaitez pas avoir de valeur par défaut, n'utilisez pas du tout ce champ.

Quand vous utilisez `po-debconf`, (et vous **devriez**, voir *Courtoisie avec les traducteurs*), veuillez rendre ce champ traduisible si vous pensez qu'il peut l'être.

Exemple, pris dans le paquet geneweb :

```
Template: geneweb/lang
Type: select
__Choices: Afrikaans (af), Bulgarian (bg), Catalan (ca), Chinese (zh), Czech (cs),␣
↪Danish (da), Dutch (nl), English (en), Esperanto (eo), Estonian (et), Finnish (fi),␣
↪French (fr), German (de), Hebrew (he), Icelandic (is), Italian (it), Latvian (lv),␣
↪Norwegian (no), Polish (pl), Portuguese (pt), Romanian (ro), Russian (ru), Spanish␣
↪(es), Swedish (sv)
# This is the default choice. Translators may put their own language here
# instead of the default.
# WARNING : you MUST use the ENGLISH NAME of your language
# For instance, the French translator will need to put French (fr) here.
_Default: English[ translators, please see comment in PO files]
_Description: Geneweb default language:
```

Veuillez noter l'utilisation de crochets pour autoriser des commentaires internes dans les champs de `debconf`. Notez également l'utilisation de commentaires qui apparaîtront dans les fichiers de travail des traducteurs.

Les commentaires sont très utiles, car l'astuce « `_Default` » est parfois déroutante pour les traducteurs qui pourraient y mettre leur propre choix.

6.7 Internationalisation

Cette section fournit des informations générales à destination des développeurs pour simplifier la vie des traducteurs. Vous trouverez plus d'informations à destination des traducteurs et développeurs intéressés par l'internationalisation dans la documentation sur l'internationalisation et la localisation dans Debian.

6.7.1 Gestion des traductions `debconf`

Comme les porteurs, les traducteurs ont une tâche difficile. Ils travaillent sur de nombreux paquets et doivent collaborer avec de nombreux responsables. De plus, ils n'ont généralement pas la langue anglaise comme langue maternelle et vous devez donc faire preuve d'une patience particulière avec eux.

L'objectif de `debconf` est de rendre la configuration des paquets plus facile pour les responsables de paquets et pour les utilisateurs. Initialement, la traduction des écrans de debconf était gérée avec `debconf-mergetemplate`. Cependant, cette technique est désormais obsolète et la meilleure façon d'internationaliser `debconf` est d'utiliser le paquet `po-debconf`. Cette méthode simplifie le travail des traducteurs et des responsables et des scripts de transition sont fournis.

Using `po-debconf`, the translation is stored in `.po` files (drawing from `gettext` translation techniques). Special template files contain the original messages and mark which fields are translatable. When you change the value of a translatable field, by calling `debconf-updatepo`, the translation is marked as needing attention from the translators. Then, at build time, the `dh_installdebconf` program takes care of all the needed magic to add the template along with the up-to-date translations into the binary packages. Refer to the po-debconf(7) manual page for details.

6.7.2 Documentation internationalisée

L'internationalisation de la documentation est primordiale pour les utilisateurs, mais représente un travail très important. Même s'il n'est pas possible de supprimer tout le travail nécessaire, il est possible de faciliter la tâche des traducteurs.

Si vous maintenez une documentation de quelque taille que ce soit, il sera plus pratique pour les traducteurs d'avoir accès au système de suivi des versions source. Cela leur permet de voir les différences entre deux versions de la documentation et, par conséquent, de mieux voir où les traductions doivent être modifiées. Il est recommandé que la documentation traduite contienne l'indication du système de suivi des versions source qui est utilisé. Un système pratique est fourni par doc-check du paquet `debian-installer`, qui permet un survol de l'état de la traduction pour toute langue, par l'utilisation de commentaires structurés dans la version du fichier à traduire et, pour le fichier traduit, la version du fichier sur laquelle est basée la traduction. Il est possible d'adapter ce système dans votre propre dépôt de gestion de versions.

Si vous maintenez de la documentation en format XML ou SGML, il est conseillé d'isoler l'information indépendante de la langue et de la définir sous forme d'entités dans un fichier à part qui sera inclus par toutes les traductions. Cela rend par exemple plus simple la maintenance d'URL dans de nombreux fichiers.

Certains outils (par exemple `po4a`, `poxml`, ou `translate-toolkit`) sont spécialisés dans l'extraction des composants traduisibles depuis différents formats. Ils fabriquent des fichiers PO (un format plutôt habituel pour les traducteurs), qui permettent de voir les traductions à mettre à jour quand le document a été modifié.

6.8 Best practices for `debian/patches`

Debian packages might suffer from bugs in the upstream code that you need to deal with. In the source format "3.0 (quilt)" patches are stored in `debian/patches/` and automatically applied as listed in `debian/patches/series` when the source package is unpacked.

Patches should be documented following DEP-3.

Several tools exist to automate managing the patches. If you manage a source package outside of any Git repository, then your best option is likely `quilt`. Otherwise, you should consider to rely on Git's built-in features or on on the git packaging helper that you use (if any). In particular, for packages using `git-buildpackage`, you should use the `gbp pq` commands to manage the contents of the `debian/patches/` directory.

A single patch can be created with e.g. `git format-patch -1 d33286c` from a single commit. Avoid using `git show` as it lacks the full headers.

If the upstream fix is spread across multiple commits but makes sense to apply (and drop) in Debian as a single patch, one could use a command such as `git format-patch --stdout abc123..def456 > debian/patches/...` and append the `Bug` field only in the commit message of the first commit in the patch.

If one appends *.patch* to the url of a GitHub commit or Pull Request or GitLab commit or Merge Request, the resulting patch file is using this same format (as if it were generated by `git format-patch`).

Remember to always append a `Bug` header to the patch description so that a reader can follow the link to see where the bug was reported or patch submitted. If the purpose of the patch is to specifically divert from upstream permanently, append the header *Forwarded : not-needed* to the end of the description.

6.9 Situations courantes de gestion de paquets

6.9.1 Paquets utilisant `autoconf` ou `automake`

Pouvoir disposer de fichiers `config.sub` et `config.guess` à jour est un point critique pour les porteurs, particulièrement pour les architectures assez volatiles. De très bonnes pratiques applicables à tout paquet qui utilise `autoconf` ou `automake` ont été résumées dans `/usr/share/doc/autotools-dev/README.Debian.gz` du paquet `autotools-dev`. Il est fortement recommandé de lire ce fichier et d'en suivre les recommandations.

6.9.2 Bibliothèques

Les paquets fournissant des bibliothèques sont plus difficiles à maintenir pour plusieurs raisons. La Charte impose de nombreuses contraintes pour en faciliter la maintenance et garantir que les mises à niveau sont aussi simples que possible quand une nouvelle version amont est disponible. Des erreurs dans une bibliothèque sont susceptibles de rendre inutilisables de très nombreux paquets.

Good practices for library packaging have been grouped in the library packaging guide.

6.9.3 Documentation

Veuillez vous assurer que vous suivez la Charte de documentation.

Si votre paquet contient de la documentation construite à partir de fichiers XML ou SGML, il est recommandé de ne pas fournir ces fichiers source dans les paquets binaires. Les utilisateurs qui souhaiteraient disposer des sources de la documentation peuvent alors récupérer le paquet source.

La Charte indique que la documentation devrait être fournie en format HTML. Il est recommandé de la fournir également dans les formats PDF et texte si cela est pratique et si un affichage de qualité raisonnable est possible. Cependant, il est le plus souvent inapproprié de fournir en format texte simple des versions de documentations dont le format source est HTML.

Les manuels les plus importants qui sont fournis devraient être enregistrés avec `doc-base` lors de leur installation. Veuillez consulter la documentation du paquet `doc-base` pour plus d'informations.

La Charte Debian (section 12.1) indique que des pages de manuel devraient être fournies avec chaque programme, utilitaire et fonction, et suggère d'en fournir pour les autres éléments comme les fichiers de configuration. Si le travail que vous empaquetez ne fournit pas de telles pages de manuel, veuillez envisager de les écrire pour les ajouter à votre paquet et les proposer en amont.

Les pages de manuel n'ont pas besoin d'être écrites directement au format troff. Les formats source populaires Docbook, POD et reST peuvent être convertis en utilisant respectivement `xsltproc`, `pod2man` et `rst2man`. De moins grande ampleur, le programme `help2man` peut aussi être utilisé pour écrire une ébauche.

6.9.4 Catégories particulières de paquets

Plusieurs catégories particulières de paquets utilisent des chartes spécifiques avec leurs règles et leurs pratiques d'empaquetage.

- Les paquets liés à Perl utilisent une charte Perl. Des exemples de tels paquets qui appliquent cette charte spécifique sont `libdbd-pg-perl` (module Perl binaire) ou `libmldbm-perl` (module Perl indépendant de l'architecture).
- Python related packages have their Python policy; see `/usr/share/doc/python3/python-policy.txt.gz` in the `python3` package.
- Les paquets liés à Emacs utilisent une charte Emacs.
- Les paquets liés à Java utilisent une charte Java.
- Les paquets liés à OCaml utilisent leur propre charte, que l'on peut trouver dans le fichier `/usr/share/doc/ocaml/ocaml_packaging_policy.gz` du paquet `ocaml`. Un bon exemple est fourni par le paquet source `camlzip`.
- Les paquets fournissant des DTD XML ou SGML devraient suivre les recommandations données dans le paquet `sgml-base-doc`.
- Rust packaging is described in the Debian Rust Team Book;.
- Packages providing services ("daemons") should be functional on a fresh install, to the extent that that is possible without compromising security (e.g. a web server should by default be up and running and serve a dummy page, but must otherwise not allow unauthenticated sensitive operations; consider whether to serve only on the localhost network interface, by default).
- Web application packages should aim to have their dependencies (including javascript) packaged separately, and should carry out whatever setup is necessary for basic and secure functionality out of the box (e.g. create a database, ship configs with reasonable defaults, install files in appropriate location with appropriate permissions, etc). For examples, look at how existing web applications are packaged, e.g. dfsg-new-queue for Go, gitlab for ruby on rails, node-shiny-server for NPM. diaspora-installer is a dummy package which downloads diaspora (also pulling in runtime dependencies as rubygems) and configures it to use PostgreSQL and Nginx.

6.9.5 Données indépendantes de l'architecture

Il est fréquent qu'un grand nombre de données indépendantes de l'architecture soient fournies avec un programme. Cela peut être par exemple des fichiers audio, un ensemble d'icônes, des motifs de papier-peint ou d'autres fichiers graphiques. Si la taille de ces données est négligeable par rapport à la taille du reste du paquet, il est probablement préférable de laisser l'ensemble dans un seul paquet.

Cependant, si cette taille est importante, vous devriez réfléchir à les fournir dans un paquet séparé, indépendant de l'architecture (`_all.deb`). Cela permet ainsi d'éviter la duplication des mêmes données dans de nombreux paquets binaires, un par architecture. Bien que cela ajoute des entrées dans les fichiers `Packages`, cela permet d'économiser une place importante sur les miroirs de Debian. La séparation des données indépendantes de l'architecture réduit également le temps de traitement de `lintian` (voir *Contrôle de paquets (« lint »)*) lorsqu'il est utilisé sur l'archive Debian en entier.

6.9.6 Besoin de paramètres régionaux spécifiques lors de la construction

Si des paramètres régionaux (« `locale` ») sont nécessaires pour la construction d'un paquet, vous pouvez créer un fichier temporaire avec l'astuce suivante.

Si la variable `LOCPATH` est placée sur l'équivalent de `/usr/lib/locale` et `LC_ALL` sur le nom des paramètres régionaux à créer, vous devriez pouvoir obtenir le résultat escompté sans avoir les privilèges du superutilisateur. La séquence ressemblera alors à :

```
LOCALE_PATH=debian/tmpdir/usr/lib/locale
LOCALE_NAME=en_IN
LOCALE_CHARSET=UTF-8

mkdir -p $LOCALE_PATH
localedef -i $LOCALE_NAME.$LOCALE_CHARSET -f $LOCALE_CHARSET $LOCALE_PATH/$LOCALE_
↪NAME.$LOCALE_CHARSET

# Using the locale
LOCPATH=$LOCALE_PATH LC_ALL=$LOCALE_NAME.$LOCALE_CHARSET date
```

6.9.7 Paquets de transition conformes à `deborphan`

Le programme `deborphan` permet aux utilisateurs d'identifier les paquets pouvant être supprimés sans crainte du système, c'est-à-dire ceux dont aucun paquet ne dépend. Par défaut, l'utilitaire n'effectue sa recherche que parmi les paquets de bibliothèque et les sections `libs` et `oldlibs`, afin de traquer les bibliothèques inutilisées. Cependant, avec le paramètre approprié, il peut rechercher d'autres paquets inutiles.

Par exemple, avec l'option `--guess-dummy`, `deborphan` essaye de trouver tous les paquets de transition qu'il était nécessaire de mettre à niveau, mais qui peuvent maintenant être supprimés. Pour ce faire, `deborphan` recherche les chaînes `dummy` ou `transitional` dans leur description courte, bien que ce serait mieux de rechercher les deux, dans la mesure où il y a des paquets `dummy` ou `transitional` qui ont d'autres finalités.

Ainsi, lorsque vous avez besoin de créer un tel paquet, veuillez prendre soin d'ajouter `transitional dummy package` à sa description courte pour rendre ce statut explicite. Il est facile de trouver des exemples avec les commandes `apt-cache search .|grep dummy` ou `apt-cache search .|grep transitional`.

De même, vous devriez configurer sa section en `oldlibs` et sa priorité en `optional` afin de faciliter le travail de `deborphan`.

6.9.8 Meilleures pratiques pour les fichiers `.orig.tar.{gz,bz2,xz}`

Il existe deux sortes différentes d'archives source d'origine. Les sources originelles (« `pristine` ») et les sources reconstruites (« `repackaged` »).

6.9.8.1 Source originelle (« pristine »)

La caractéristique définissant une archive source originelle est que le fichier `.orig.tar.{gz,bz2,xz}` est strictement identique à l'archive fournie par l'auteur amont.[1]. Cela permet d'utiliser des sommes de contrôle pour vérifier que toutes les modifications effectuées entre la version Debian et la version amont sont contenues dans le fichier de différences Debian. De même, si la taille des sources d'origine est importante, les auteurs amont et tous ceux qui disposent de l'archive amont d'origine peuvent économiser du temps de téléchargement s'ils souhaitent contrôler le paquet en détail.

Il n'existe pas de convention universellement acceptée pour la structure de répertoires que devraient adopter les auteurs amont dans les archives qu'ils publient, mais `dpkg-source` peut de toute manière traiter la plupart des archives amont comme des sources originelles. La stratégie de cette commande est la suivante :

1. elle extrait l'archive dans un répertoire temporaire :

```
zcat path/to/packagename_upstream-version.orig.tar.gz | tar xf -
```

2. si, après cela, le répertoire temporaire ne contient qu'un seul répertoire sans fichiers, `dpkg-source` renomme ce répertoire en *nomdupaquet-version-amont*`(.orig)`. Le nom du répertoire parent de l'archive tar n'a pas d'importance et est oublié ;
3. si ce n'est pas le cas, l'archive amont a été créée sans répertoire parent (honte à l'auteur amont !). Dans ce cas, `dpkg-source` renomme le répertoire temporaire *lui-même* en *nomdupaquet-version-amont*`(.orig)`.

6.9.8.2 Source amont reconstruite

Vous **devriez** envoyer les paquets avec une archive source inchangée, dans la mesure du possible. Il existe cependant plusieurs raisons qui peuvent rendre cela impossible. C'est notamment le cas si les auteurs amont ne distribuent pas d'archive tar compressée du tout ou si l'archive amont contient des parties non conformes aux principes du logiciel libre selon Debian, qui doivent être supprimées avant l'envoi.

Dans ces cas, les responsables doivent construire eux-mêmes une archive `.orig.tar.{gz,bz2,xz}`. Cette archive sera appelée une archive amont reconstruite. Il est important de noter qu'elle reste différente d'un paquet natif. Une archive reconstruite est toujours fournie avec les changements propres à Debian dans un fichier `.diff.gz` ou `.debian.tar.{gz,bz2,xz}` séparé et son numéro de version est toujours composé de *upstream-version* et *debian-version*.

Il peut exister des cas où il est souhaitable de reconstruire une archive source alors que les auteurs amont fournissent bien une archive `.tar.{gz,bz2,xz}` qui pourrait être utilisée directement. Le plus évident est la recherche d'un gain de place *significatif* par recompression ou par suppression de scories inutiles de l'archive source d'origine. Il est important que le responsable exerce avec discernement son propre jugement et soit prêt à le justifier si l'archive source est reconstruite alors qu'elle aurait pu être fournie telle quelle.

Un fichier `.orig.tar.{gz,bz2,xz}` reconstruit :

1. **should** be documented in the resulting source package. Detailed information on how the repackaged source was obtained, and on how this can be reproduced should be provided in `debian/copyright`, ideally in a way that can be done automatically with uscan. If that really doesn't work, at least provide a `get-orig-source` target in your `debian/rules` file that repeats the process, even though that was actually deprecated in the 4.1.4 version of the Debian policy.
2. **ne devrait pas** contenir de fichier non distribué par les auteurs amont, ou dont vous avez modifié le contenu ;[2]
3. **should**, except where impossible for legal reasons, preserve the entire building and portability infrastructure provided by the upstream author. For example, it is not a sufficient reason for omitting a file that it is used

1. Il est impossible d'empêcher les auteurs amont de modifier l'archive qu'ils distribuent sans également incrémenter le numéro de version. Il est donc impossible de garantir qu'une archive originelle est identique à ce que l'auteur amont *distribue* à un instant donné. Tout ce qu'il est possible de garantir est qu'elle a été identique à ce que les auteurs amont *ont distribué* à un moment donné. Si une différence apparaît plus tard (par exemple si les auteurs amont découvrent ne pas avoir utilisé la compression maximale dans leur distribution d'origine et la recompresse, c'est tout simplement dommage. Comme il n'existe pas de méthode adaptée pour envoyer un nouveau fichier `.orig.tar.{gz,bz2,xz}` pour la même version, il est même totalement inutile de traiter cette situation comme un bogue.

2. Avec pour exception particulière, si l'omission de fichiers non libres provoque une erreur de compilation du source sans l'aide du diff Debian, il peut être pertinent de modifier les fichiers pour enlever les portions non libres ou d'expliquer la situation dans un fichier `README.source` à la racine de l'arbre des sources. Veuillez dans ce cas encourager l'auteur amont à rendre les portions non libres faciles à séparer du reste des sources.

only when building on MS-DOS. Similarly, a `Makefile` provided by upstream should not be omitted even if the first thing your `debian/rules` does is to overwrite it by running a configure script.

(*Raison :* les utilisateurs Debian ont l'habitude, pour compiler des logiciels sur des systèmes non Debian, de prendre les sources depuis les miroirs Debian plutôt que d'essayer de trouver le dépôt officiel amont) ;

4. **may** use *packagename*`-`*upstream-version*`+dfsg` (or any other suffix which is added to the tarball name) as the name of the top-level directory in its tarball. This makes it possible to distinguish pristine tarballs from repackaged ones.
5. **should** be compressed with `xz` (or `gzip` or `bzip2`) with maximal compression.

6.9.8.3 Modification de fichier binaire

Sometimes it is necessary to change binary files contained in the original tarball, or to add binary files that are not in it. This is fully supported when using source packages in "3.0 (quilt)" format ; see the dpkg-source(1) manual page for details. When using the older format "1.0", binary files can't be stored in the `.diff.gz` so you must store a `uuencode`d (or similar) version of the file(s) and decode it at build time in `debian/rules` (and move it in its official location).

6.9.9 Meilleures pratiques pour les paquets de débogage

Un paquet de débogage est un paquet qui contient des informations supplémentaires que `gdb` peut utiliser. Puisque les binaires de Debian sont élagués par défaut, les informations de débogage, comme les noms de fonction et les numéros de ligne, ne sont pas disponibles lors de l'utilisation de `gdb` sur les paquets binaires. Les paquets de débogage permettent aux utilisateurs, qui ont besoin de ces informations de débogage complémentaires, d'ajouter ces informations supplémentaires et d'éviter d'augmenter la taille d'un système normal.

Les paquets de débogage contiennent des symboles de débogage distincts que `gdb` peut trouver et charger à la volée lors du débogage d'un programme ou d'une bibliothèque. Par convention dans Debian, ces symboles sont conservés dans `/usr/lib/debug/`*chemin*, où *chemin* est l'arborescence vers l'exécutable ou la bibliothèque. Par exemple, les symboles de débogage pour `/usr/bin/truc` sont dans `/usr/lib/debug/usr/bin/truc`, et les symboles de débogage pour `/usr/lib/libtruc.so.1` sont dans `/usr/lib/debug/usr/lib/libtruc.so.1`.

6.9.9.1 Paquets de débogage créés automatiquement

Les paquets de symboles de débogage peuvent être créés automatiquement pour n'importe quel paquet binaire contenant des exécutables, et sauf cas pathologiques, il ne devrait plus être nécessaire d'utiliser l'ancienne méthode de création manuelle. Le nom de paquet pour un nom de paquet de symboles de débogage généré automatiquement se termine par `-dbgsym`.

Les paquets `dbgsym` ne sont pas installés dans des archives standard, mais dans des archives spéciales. Cela signifie que si vous avez besoin de symboles de débogage pour le débogage, vous devez ajouter ces archives dans votre configuration d'apt et puis installer le paquet `dbgsym` qui vous intéresse. Veuillez lire https://wiki.debian.org/HowToGetABacktrace pour connaître la manière de faire.

6.9.9.2 Paquets -dbg manuels

Avant l'introduction des paquets automatiques `dbgsym`, les paquets de débogage devaient être créés manuellement. Le nom de ces paquets se terminent par `-dbg`. Il est recommandé de migrer de tels anciens paquets vers les nouveaux paquets `dbgsym` autant que possible. La procédure de conversion est décrite dans https://wiki.debian.org/AutomaticDebugPackages, mais l'idée générale est d'utiliser le commutateur `--dbgsym-migration='pkgname-dbg (<< currentversion~)'` de la commande `dh_strip`.

Cependant, parfois il n'est pas possible de convertir vers les nouveaux paquets `dbgsym`, ou vous rencontrerez les anciens paquets manuels -dbg dans les archives, donc vous devrez les gérer. Il n'est pas recommandé de créer des paquets -dbg pour les nouveaux paquets, sauf si les paquets automatiques ne fonctionnent pas pour une raison quelconque.

Une raison peut être que les paquets de débogage contiennent une construction spéciale de débogage d'une bibliothèque ou autre binaire. Cependant, généralement séparer des informations de débogage des binaires déjà construits est suffisant et économisera de l'espace au moment de la compilation.

C'est le cas, par exemple, pour les symboles de débogage des extensions de Python. Pour l'instant, la façon correcte d'empaqueter les symboles de débogage des extensions de Python est d'utiliser les paquets `-dbg` comme décrit dans https://wiki.debian.org/Python/DbgBuilds.

Pour créer des paquets `-dbg`, le responsable doit les stipuler explicitement dans `debian/control`.

Les symboles de débogage peuvent être extraits d'un fichier objet à l'aide de `objcopy --only-keep-debug`. Ensuite les informations de débogage peuvent être supprimées du fichier objet et `objcopy --add-gnu-debuglink` peut être utilisé pour préciser le chemin vers le fichier contenant les symboles de débogage. objcopy 1 explique en détail le fonctionnement.

Remarquez que le paquet de débogage devrait dépendre du paquet dont il fournit les symboles de débogage, et que cette dépendance devrait être spécifique à la version. Par exemple

```
Depends: libfoo (= ${binary:Version})
```

La commande `dh_strip` de `debhelper` permet de créer les paquets de débogage et prend soin d'utiliser `objcopy` pour séparer les symboles de débogage à votre place. Si le paquet utilise `debhelper/9.20151219`, ou une version plus récente, inutile de faire quelque chose. `debhelper` créera les paquets de débogage (sous forme `paquet-dbgsym`) pour vous sans modifications supplémentaires dans votre paquet source.

6.9.10 Meilleures pratiques pour les métapaquets

Un métapaquet est un paquet quasiment vide qui facilite l'installation d'un ensemble de paquets cohérents qui peut évoluer avec le temps. Il atteint cet objectif en dépendant de tous les paquets de l'ensemble. Grâce à la puissance d'apt, le responsable du métapaquet peut configurer les dépendances et le système de l'utilisateur obtiendra automatiquement les paquets supplémentaires. Les paquets devenus inutiles qui avaient été installés automatiquement seront aussi marqués comme candidats à la suppression (et même automatiquement supprimés par `aptitude`). Par exemple, `gnome` et `linux-image-amd64` sont deux métapaquets (construits par les paquets source `meta-gnome2` et `linux-latest`).

La description longue du métapaquet doit clairement expliquer son objectif, afin d'informer les utilisateurs sur ce qu'ils perdront s'ils suppriment le paquet. Il est recommandé d'être explicite sur les conséquences. C'est tout particulièrement important pour les métapaquets installés lors de l'installation initiale qui n'ont pas été installés explicitement par l'utilisateur. Ils ont tendance à être importants pour garantir les mises à niveau du système et la description devrait essayer de dissuader les utilisateurs de les désinstaller pour éviter d'éventuels dommages.

CHAPITRE 7

Au-delà de l'empaquetage

Debian, c'est beaucoup plus que de l'empaquetage de logiciels et de la maintenance de paquets. Ce chapitre contient des informations sur les façons, souvent vraiment importantes, de contribuer à Debian au-delà des simples création et entretien de paquets.

En tant qu'organisation de volontaires, Debian repose sur la liberté de choisir ce sur quoi l'on désire travailler et de choisir la partie la plus importante à laquelle on veut consacrer son temps.

7.1 Signalement de bogues

Nous vous encourageons à signaler des bogues quand vous en trouvez dans les paquets Debian. En fait, les développeurs Debian sont souvent les testeurs de première ligne. Trouver et signaler les bogues dans les paquets d'autres développeurs améliore la qualité de Debian.

Lisez les instructions pour signaler un bogue dans le système de suivi des bogues Debian.

Essayez de signaler un bogue à partir d'un compte utilisateur normal avec lequel vous pouvez recevoir des courriers, pour que les personnes puissent vous joindre si elles ont besoin de plus d'informations à propos du bogue. Ne signalez pas de bogues en tant que root.

Vous pouvez utiliser un outil comme reportbug(1) pour signaler des bogues. Il peut automatiser et, dans l'ensemble, faciliter le processus.

Assurez-vous que le bogue n'a pas déjà été signalé. Chaque paquet dispose d'une liste de bogues facilement accessible à `https://bugs.debian.org/`*nomdupaquet*. Des outils comme querybts(1) peuvent également vous fournir ces informations (et `reportbug` invoquera également normalement `querybts` avant l'envoi).

Essayez d'envoyer vos bogues au bon endroit. Quand, par exemple, votre bogue concerne un paquet qui écrase des fichiers d'un autre paquet, vérifiez les listes des bogues pour les *deux* paquets afin d'éviter de créer des rapports de bogue dupliqués.

Vous pouvez également parcourir les bogues d'autres paquets, en les regroupant s'ils sont indiqués plus d'une fois, ou en les marquant avec `fixed` quand ils ont déjà été corrigés. Notez cependant que si vous n'êtes ni le rapporteur du bogue, ni le responsable du paquet, vous ne devriez pas fermer réellement le bogue (à moins d'avoir obtenu la permission du responsable).

De temps en temps, vous pourriez vouloir vérifier ce qui s'est passé à propos des bogues que vous avez signalés. Saisissez cette occasion pour fermer les bogues que vous ne pouvez plus reproduire. Pour trouver tous les bogues que vous avez signalés, vous avez simplement besoin de vous rendre à la page `https://bugs.debian.org/from:`*votre-adresse-de-courrier*.

7.1.1 Signalement d'un grand nombre de bogues en une fois (`mass bug filing`)

Signaler de nombreux bogues pour le même problème sur un grand nombre de paquets — c'est-à-dire plus de dix — est une pratique déconseillée. Prenez toutes les mesures possibles pour éviter cette situation. Si le problème peut être détecté automatiquement par exemple, ajoutez un nouveau test dans le paquet `lintian` pour générer une erreur ou un avertissement.

Si vous voulez signaler plus de dix rapports sur le même sujet, il est préférable d'indiquer votre intention sur la liste `debian-devel@lists.debian.org` et de le mentionner dans le sujet de votre message. Cela donnera à d'autres développeurs la possibilité de vérifier que le problème existe vraiment. De plus, cela permet d'éviter que plusieurs responsables ne rédigent les mêmes rapports de bogue simultanément.

Veuillez utiliser les programmes `dd-list` et si nécessaire, `whodepends` (du paquet `devscripts`) pour générer une liste de tous les paquets concernés et incluez la sortie dans votre courrier à `debian-devel@lists.debian.org`.

Quand vous envoyez un grand nombre de rapports sur le même sujet, vous devriez les envoyer à `maintonly@bugs.debian.org` pour éviter qu'ils soient renvoyés vers les listes de diffusion.

Le programme `mass-bug` (du paquet `devscripts`) peut être utilisé pour envoyer des rapports de bogue pour une liste de paquets.

7.1.1.1 Étiquettes d'utilisateur (`Usertags`)

Vous pouvez utiliser les étiquettes d'utilisateur du BTS lors du signalement de bogues sur un grand nombre de paquets. Les étiquettes d'utilisateur se comportent de la même façon que les étiquettes `patch` et `wishlist` à la différence qu'elles sont définies par l'utilisateur et occupent un espace de définition spécifique propre à l'utilisateur. Cela permet à plusieurs groupes de développeurs de marquer `Usertags` le même bogue de différentes façons sans conflit.

Pour ajouter des étiquettes d'utilisateur lors du signalement de bogues, précisez les pseudo-en-têtes `User` et `Usertags` :

```
To: submit@bugs.debian.org
Subject: title-of-bug

Package: pkgname
[ ... ]
User: email-addr
Usertags: tag-name [ tag-name ... ]

description-of-bug ...
```

Remarquez que les étiquettes sont séparées par des espaces et ne peuvent contenir de tiret bas. Si vous signalez des bogues au nom d'un groupe ou d'une équipe spécifique, il vaut mieux définir `User` comme une liste de diffusion appropriée après y avoir décrit votre intention.

Pour voir les bogues marqués par une étiquette d'utilisateur en particulier, rendez-vous sur la page `https://bugs.debian.org/cgi-bin/pkgreport.cgi?users=`*adresse-mail*`&tag=`*nom-d-etiquette*.

7.2 Effort d'assurance qualité

7.2.1 Travail quotidien

Bien qu'il y ait un groupe de personnes dédié à l'assurance qualité (`QA`), les devoirs de QA ne leur sont pas exclusivement réservés. Vous pouvez participer à cet effort en conservant vos paquets aussi exempts de bogues que possible et aussi corrects que possible selon lintian (voir *lintian*). Si cela vous paraît impossible, vous devriez alors envisager d'abandonner certains de vos paquets (voir *Abandon de paquet*). Sinon, vous pouvez demander de l'aide à d'autres personnes pour qu'elles puissent rattraper votre retard dans la correction des bogues (vous pouvez demander de l'aide sur `debian-qa@lists.debian.org` ou `debian-devel@lists.debian.org`). En même temps, vous pouvez rechercher des co-responsables (voir *Maintenance collective*).

7.2.2 Chasses aux bogues

De temps en temps, le groupe d'assurance qualité organise des chasses aux bogues (`Bug Squashing Party`) pour essayer de résoudre autant de problèmes que possible. Elles sont annoncées sur `debian-devel-announce@lists.debian.org` en précisant quel domaine sera visé pendant la chasse : habituellement, il s'agit des bogues empêchant l'intégration du paquet dans la distribution (bogues de gravité `Release Critical`), mais il peut être décidé d'aider à finir une transition majeure (comme une nouvelle version de Perl qui demande la recompilation de tous les modules binaires).

The rules for non-maintainer uploads differ during the parties because the announcement of the party is considered prior notice for NMU. If you have packages that may be affected by the party (because they have release critical bugs for example), you should send an update to each of the corresponding bugs to explain their current status and what you expect from the party. If you don't want an NMU, or if you're only interested in a patch, or if you will deal with the bug yourself, please explain that in the BTS.

Les personnes qui participent à la chasse ont des règles spécifiques pour les NMU, elles peuvent en faire une sans avertissement préalable si elles envoient leur paquet avec un délai d'au moins trois jours dans `DELAYED/3-day`. Toutes les autres règles de NMU s'appliquent comme d'habitude ; le correctif de la NMU devrait être envoyé dans le BTS (pour l'un des bogues ouverts corrigé par la NMU ou pour un nouveau bogue marqué corrigé). Les participants devraient également respecter tout souhait du responsable s'il en a exprimé.

Si vous ne vous sentez pas à l'aise avec une NMU, envoyez simplement un correctif au BTS. C'est de loin préférable à une NMU défectueuse.

7.3 Contact avec d'autres responsables

Pendant vos activités dans Debian, vous contacterez d'autres responsables pour différentes raisons. Vous pourrez vouloir discuter d'une nouvelle façon de coopérer au sein d'un ensemble de paquets liés, ou vous pouvez simplement rappeler à quelqu'un qu'une nouvelle version est disponible et que vous en avez besoin.

Chercher l'adresse du responsable d'un paquet peut être fastidieux. Heureusement, il existe un alias de courrier simple, *paquet*`@packages.debian.org`, qui fournit un moyen d'envoyer un courrier à un responsable, quelle que soit son adresse (ou ses adresses). Remplacez *paquet* par le nom du paquet source ou binaire.

Vous pouvez également vouloir contacter les personnes inscrites à un paquet source donné à l'aide de *Suivi de paquets Debian (Debian Package Tracker)*. Vous pouvez le faire en utilisant l'adresse *paquet*`@packages.qa.debian.org`.

7.4 Gestion des responsables non joignables

Si vous remarquez qu'un paquet manque de maintenance, vous devriez vous assurer que le responsable est toujours actif et qu'il continue à travailler sur ses paquets. Il est possible qu'il ne soit plus actif, mais qu'il n'ait pas démissionné du système. D'un autre côté, il est possible qu'il ait simplement besoin d'un rappel.

There is a simple system (the MIA database) in which information about maintainers who are deemed Missing In Action is recorded. When a member of the QA group contacts an inactive maintainer or finds more information about one, this is recorded in the MIA database. This system is available in the directory `/org/qa.debian.org/mia` on the host `qa.debian.org`. The directory contains the tool `mia-query` with which the database can be queried. Use `/org/qa.debian.org/mia/mia-query --help` to see how to query the database. For example, to find out information about Debian Developer named Maria you could run :

```
ssh qa.debian.org /org/qa.debian.org/mia/mia-query --name Maria
```

If you find that no information has been recorded about an inactive maintainer yet, or that you can add more information, you should generally proceed as follows.

La première étape est de contacter poliment le responsable et d'attendre une réponse pendant un temps raisonnable. Il est assez difficile de définir ce qu'est un temps raisonnable, mais il est important de prendre en compte que la vraie vie est parfois assez mouvementée. Une façon de gérer cela pourrait être d'envoyer un rappel après deux semaines.

Une adresse de courriel non valable est une violation de la politique de Debian. Si un courriel est rejeté (`bounce`), veuillez soumettre un bogue pour le paquet et informer la base de données MIA.

Si le responsable ne répond pas après quatre semaines (un mois), on peut supposer qu'il n'y aura probablement pas de réponse. Si cela se produit, vous devriez poursuivre vos investigations et essayer de réunir toutes les informations utiles sur ce responsable. Cela inclut :

— les informations `echelon` disponibles dans la base de données LDAP des développeurs, qui indiquent quand le développeur a envoyé un message pour la dernière fois sur une liste de diffusion Debian (cela inclut les envois vers les listes `debian-devel-changes@lists.debian.org`). Pensez aussi à vérifier si le responsable est indiqué comme en vacances dans la base de données ;

— le nombre de paquets de ce responsable et l'état de ces paquets. En particulier, reste-t-il des bogues empêchant l'intégration des paquets dans la distribution qui sont ouverts depuis des lustres ? De plus, combien de bogues y a-t-il en général ? Un autre renseignement important est si les paquets ont subi des NMU, et si oui, par qui ;

— est-ce que le responsable est actif en dehors de Debian ? Par exemple, il peut avoir envoyé des messages récemment à des listes de diffusion non-Debian ou des groupes de discussion ;

Un problème particulier est représenté par les paquets parrainés — le responsable n'est pas un développeur Debian officiel. Les informations `echelon` ne sont pas disponibles pour les personnes parrainées, par exemple ; vous devez donc trouver et contacter le responsable Debian qui a réellement envoyé le paquet. Étant donné qu'il a signé le paquet, il est responsable de l'envoi de toute façon et il sait probablement ce qui s'est passé avec la personne qu'il parraine.

Il est également permis d'envoyer une demande à `debian-devel@lists.debian.org` demandant si quelqu'un a des informations sur le responsable manquant. Veuillez mettre en copie (CC) la personne en question.

Une fois réunies toutes ces informations, vous pouvez contacter `mia@qa.debian.org`. Les personnes de cet alias utiliseront les informations que vous aurez fournies pour décider comment procéder. Par exemple, elles peuvent abandonner tout ou partie des paquets du responsable. Si un paquet a subi une NMU, elles peuvent préférer contacter le responsable ayant fait cette NMU avant de déclarer le paquet orphelin — il pourrait être intéressé par le paquet.

Un dernier mot : veuillez rester poli. Tout le monde est volontaire et ne peut dédier l'intégralité de son temps à Debian. Vous n'êtes pas non plus au courant de la situation de la personne impliquée. Elle est peut-être sérieusement malade ou pourrait même nous avoir définitivement quitté — vous ne savez pas qui recevra vos courriers. Imaginez le sentiment d'un proche qui lit un courrier pour la personne décédée, et trouve un message très impoli, de colère et accusateur !

D'un autre côté, bien que nous soyons des bénévoles, le responsable d'un paquet s'est engagé et a donc la responsabilité d'entretenir le paquet. Aussi, vous pouvez faire valoir l'importance du bien collectif — si un responsable n'a plus le temps ou l'envie, il devrait « laisser filer » et donner le paquet à quelqu'un ayant plus de temps ou plus intéressé.

Si vous êtes intéressé pour travailler dans l'équipe MIA, veuillez étudier le fichier `README` dans `/org/qa.debian.org/mia` sur `qa.debian.org` où les détails techniques et les procédures MIA sont documentés et contactez `mia@qa.debian.org`.

7.5 Dealing with repeatedly negligent Debian developers

Debian developers are expected to comply with a series of technical and non-technical policies (for example, respectively, the Debian Machine Usage Policy and the Debian Code of Conduct).

To err is human and one should generally assume good intent, but repeated technical mistakes and/or disrespectful interactions, combined with a dismissive attitude towards Debian policies should be referred to the Debian Community Team. Most escalations to the Community Team are resolved amicably, but extreme cases may escalate further to the Debian Project Leader (leader@debian.org) and the Debian Account Managers (da-manager@debian.org) and could ultimately lead to expulsion from the Debian project.

7.6 Interaction avec de futurs développeurs Debian

Le succès de Debian dépend de sa faculté à attirer et conserver de nouveaux et talentueux volontaires. Si vous êtes un développeur expérimenté, nous vous recommandons de vous impliquer dans le processus pour devenir un nouveau responsable. Cette section décrit comment aider les futurs développeurs.

7.6.1 Parrainage de paquets

Parrainer un paquet signifie envoyer un paquet pour un responsable qui n'est pas encore autorisé à le faire lui-même. Ce n'est pas une mince affaire, le parrain doit vérifier l'empaquetage et s'assurer qu'il respecte le haut niveau d'exigence qualité que Debian s'efforce de conserver.

Les développeurs Debian peuvent parrainer des paquets, mais pas les responsables Debian.

Le processus de parrainage d'un paquet est :

1. The maintainer prepares a source package (`.dsc`) and puts it online somewhere (like on mentors.debian.net) or even better, provides a link to a public VCS repository (see *salsa.debian.org : dépôts Git et plateforme de développement collaborative*) where the package is maintained.
2. le parrain télécharge (ou extrait du dépôt) le paquet source ;
3. le parrain vérifie le paquet source. En cas de problème, il informe le responsable et lui demande de fournir une version corrigée (le processus reprend à la première étape) ;
4. le parrain ne trouve aucun problème résiduel. Il construit le paquet, le signe, et l'envoie dans Debian.

Avant de se plonger dans les détails du parrainage de paquet, vous devriez vous demander si l'ajout du paquet proposé est profitable à Debian.

Il n'y a pas de règle simple pour répondre à cette question, cela peut dépendre de plusieurs facteurs : le code source amont est-il suffisamment achevé et pas miné de trous de sécurité ? Existe-t-il déjà des paquets qui font la même chose et que donnent les comparaisons avec ce nouveau paquet ? Le paquet a-t-il été demandé par des utilisateurs et le nombre d'utilisateurs est-il significatif ? Les développeurs amont sont-ils actifs ?

Vous devriez également vérifier que le futur responsable sera un bon responsable. A-t-il déjà de l'expérience avec d'autres paquets ? Si oui, réalise-t-il du bon travail avec ceux-ci (contrôlez quelques bogues) ? Est-il familier avec le paquet et son langage de programmation ? A-t-il les compétences nécessaires pour ce paquet ? Si non, est-il capable de les acquérir ?

C'est aussi une bonne idée de savoir comment il se situe par rapport à Debian : est-il en accord avec la philosophie de Debian et a-t-il l'intention de rejoindre Debian ? Étant donné la facilité de devenir un membre de Debian, vous voudrez peut-être n'être le parrain que de personnes qui prévoient de rejoindre Debian. De cette manière, vous savez dès le début que vous n'aurez pas à jouer le rôle de parrain indéfiniment.

7.6.1.1 Parrainage d'un nouveau paquet

Les nouveaux responsables ont souvent quelques difficultés à créer des paquets Debian — cela est bien compréhensible. Ils feront des erreurs. C'est pourquoi le parrainage d'un tout nouveau paquet dans Debian demande une inspection minutieuse de l'empaquetage. Parfois plusieurs itérations seront nécessaires avant que le paquet ne soit assez bon pour être envoyé dans Debian. Ainsi, être un parrain signifie être un mentor.

Ne parrainez jamais de paquet sans l'avoir vérifié. La vérification de nouveau paquet réalisée par les responsables de l'archive veille principalement à ce que le logiciel soit vraiment libre. Bien sûr, ils tombent parfois sur des problèmes d'empaquetage, mais ça ne devrait vraiment pas arriver. Il est de votre responsabilité de vérifier que le paquet envoyé est compatible avec les principes du logiciel libre selon Debian et qu'il est de bonne qualité.

La construction du paquet et l'essai du logiciel fait partie de la vérification, mais ça ne suffit pas. La suite de cette section est une liste non exhaustive de points à vérifier lors de votre contrôle. [1]

— Vérifiez que l'archive source fournie est la même que celle distribuée par l'auteur amont (quand les sources ont été réempaquetées pour Debian, créez vous-même l'archive modifiée).
— Exécutez `lintian` (consulter *lintian*). De nombreux problèmes usuels seront découverts. Prenez soin de vérifier que tous les contrôles de `lintian` ignorés par le responsable sont pleinement justifiés.
— Run `licensecheck` (see *licensecheck*) and verify that `debian/copyright` seems correct and complete. Look for license problems (like files with "All rights reserved" headers, or with a non-DFSG compliant license). `grep -ri` is your friend for this task.
— Construisez le paquet avec `pbuilder` (ou n'importe quel outil du même genre, consultez *pbuilder*) pour vérifier que les dépendances de constructions sont exhaustives.
— Relisez `debian/control` : est-il conforme aux meilleures pratiques (consultez *Meilleures pratiques pour debian/control*) ? Les dépendances sont elles exhaustives ?

1. D'autres vérifications sont disponibles dans le wiki où plusieurs développeurs partagent leurs propres listes de contrôle de parrainage.

— Relisez `debian/rules` : est-il conforme aux meilleures pratiques (consultez *Meilleures pratiques pour debian/rules*) ? Pouvez-vous apporter quelques améliorations ?
— Relisez les scripts du responsable (`preinst`, `postinst`, `prerm`, `postrm`, `config`) : est-ce que `preinst` et `postrm` fonctionneront si les dépendances ne sont pas installées ? Est-ce que tous les scripts sont idempotents (c'est-à-dire peuvent-ils être exécutés plusieurs fois sans conséquences) ?
— Vérifiez toutes les modifications des fichiers amont (dans `.diff.gz`, `debian/patches/` ou directement dans l'archive `debian` pour les fichiers binaires). Sont-elles justifiées ? Sont-elles correctement documentées (conformément à DEP-3 pour les correctifs) ?
— Pour chaque fichier, demandez-vous pourquoi le fichier est là et si c'est la bonne façon d'atteindre le but voulu. Est-ce que le responsable suit les meilleurs pratiques d'empaquetage (consultez *Meilleures pratiques d'empaquetage*) ?
— Construisez les paquets, installez-les et essayez le logiciel. Vérifiez de pouvoir supprimer et purger les paquets. Essayez-les si possible avec `piuparts`.

Si la vérification n'a révélé aucun problème, vous pouvez construire le paquet et l'envoyer dans Debian. Rappelez-vous que même sans être le responsable du paquet, le parrain est toujours responsable de ce qu'il envoie dans Debian. C'est pourquoi vous devriez suivre le paquet (cf. *Suivi de paquets Debian (Debian Package Tracker)*).

Remarquez que vous ne devriez pas avoir à modifier le paquet source pour mettre votre nom dans les fichiers `changelog` ou `control`. Le champ `Maintainer` du fichier `control` et le fichier `changelog` devraient afficher la personne qui a fait l'empaquetage, c'est-à-dire, le filleul. Ainsi, ils recevront tous les courriers du système de suivi des bogues (BTS).

À la place, vous devriez indiquer à `dpkg-buildpackage` d'utiliser votre clef en signature. L'option `-k` le permet :

```
dpkg-buildpackage -kKEY-ID
```

Si vous utilisez `debuild` et `debsign`, vous pouvez même le configurer de façon permanente dans `~/.devscripts` :

```
DEBSIGN_KEYID=KEY-ID
```

7.6.1.2 Parrainage de la mise à jour d'un paquet existant

Vous pourrez en général supposer que le paquet a déjà été vérifié complètement. Au lieu de recommencer depuis le début, vous devriez analyser précautionneusement les différences entre la version actuelle et la nouvelle version préparée par le responsable. Si vous n'avez pas fait la vérification initiale vous-même, vous pourriez tout de même regarder de plus près au cas où elle aurait été négligée.

Afin de comparer les différences, il vous faudra les deux paquets. Téléchargez la version actuelle du paquet source (avec `apt-get source`) et reconstruisez-le (ou téléchargez les paquets binaires actuels avec `aptitude download`). Téléchargez le paquet source à parrainer (normalement avec `dget`).

Lisez la nouvelle entrée du journal de modification, elle devrait vous apprendre ce que vous devriez trouver lors de la vérification. L'outil principal à utiliser est `debdiff` (du paquet `devscripts`), vous pouvez l'exécuter avec deux paquets source (fichiers `.dsc`), deux paquets binaires, ou deux fichiers `.changes` (seront alors comparés tous les paquets binaires présents dans le fichier `.changes`).

Si vous comparez les paquets source (à l'exception des fichiers amont dans le cas d'une nouvelle version amont, en filtrant par exemple la sortie de `debdiff` avec `filterdiff -i '*/debian/*'`), vous devez comprendre toutes les modifications et elles devraient être convenablement documentées dans le journal de modifications Debian.

Si tout est correct, construisez le paquet et comparez les paquets binaires pour vérifier que les modifications du paquet source n'ont pas des conséquences inattendues (comme certains fichiers supprimés par erreur, des dépendances manquantes, etc.).

Vous pourriez consulter le système de suivi des paquets (consultez *Suivi de paquets Debian (Debian Package Tracker)*) pour vérifier que le responsable n'a rien oublié d'important. Des mises à jour de traductions pourraient attendre d'être intégrées dans le BTS. Le paquet pourrait avoir été la cible d'une NMU précédente et le responsable pourrait avoir oublié d'intégrer les modifications apportées. Un bogue critique pour la publication oublié pourrait empêcher la migration vers `testing`. Si vous trouvez quelque chose à améliorer, il est temps de le signaler pour que le responsable puisse s'améliorer la prochaine fois, et ainsi lui donner l'opportunité de mieux comprendre ses responsabilités.

Si vous n'avez pas trouvé de problème majeur, envoyez la nouvelle version. Sinon, demandez au responsable de fournir une version corrigée.

7.6.2 Granting upload permissions to DMs

After a Debian Maintainer's key has been added to the debian-maintainers keyring, a Debian Developer may grant upload permissions to the DM for specific packages by uploading a signed dak command to ftp.upload.debian.org as described in the FTP-Master's announcement to debian-devel.

This process can be simplified with the help of the `dcut` command from the `dput-ng` package. Note that this does not work with the `dcut` command from the `dput` package !

For example :

```
dcut dm --uid 0xfedcba9876543210 --allow nano --deny bash
```

If the DM's key is not in the keyring package yet but in the DD's local keyring, use the `--force` option and the fingerprint, without spaces and, in this special case, without the 0x prefix and in all uppercase :

```
dcut --force dm --uid FEDCBA9876543210FEDCBA9876543210 --allow nano
```

7.6.3 Recommandation d'un nouveau développeur

Les recommandations pour un futur développeur sont disponibles sur le site web de Debian.

7.6.4 Gestion des nouvelles candidatures

La liste de contrôle pour les responsables de candidature est disponible sur le site web de Debian.

CHAPITRE 8

Internationalisation et traduction

Debian prend en charge un nombre toujours croissant de langues naturelles. Même si l'anglais est votre langue maternelle et que vous ne parlez pas d'autre langue, il est de votre devoir de responsable d'être conscient des problèmes d'internationalisation (abrégé en i18n à cause des 18 lettres entre le « i » et le « n » de « internationalisation »). C'est pourquoi, même si des programmes seulement en anglais vous suffisent, vous devriez lire la majeure partie de ce chapitre.

i18n means modification of software or related technologies so that software can potentially handle multiple languages, customs, and other differences, while L10N (localization) means implementation of a specific language for already-internationalized software.

La l10n et l'i18n sont interconnectées, mais les difficultés liées à chacune sont très différentes. Il n'est pas vraiment difficile de permettre à un programme de changer la langue dans laquelle sont affichés les textes selon les paramètres de l'utilisateur, mais il est très coûteux en temps de traduire réellement ces messages. D'un autre côté, définir le codage des caractères est trivial, mais adapter le code pour utiliser des codages de caractères différents est un problème vraiment difficile.

En laissant de côté les problèmes d'i18n pour lesquels il n'existe pas de règle générale, il n'y a pas réellement d'infrastructure centralisée pour la l10n dans Debian qui puisse être comparée au mécanisme `buildd` pour le portage. Le plus gros du travail doit donc être réalisé manuellement.

8.1 Gestion des traductions au sein de Debian

La gestion des traductions des textes contenus dans un paquet est encore une tâche manuelle et le processus dépend du type de texte que vous désirez voir traduit.

For program messages, the gettext infrastructure is used most of the time. Often the translation is handled upstream within projects like the Free Translation Project, the GNOME Translation Project or the KDE Localization project. The only centralized resources within Debian are the Central Debian translation statistics, where you can find some statistics about the translation files found in the actual packages and download those files.

Les descriptions de paquets sont traduites depuis plusieurs années et les responsables n'ont rien à faire de particulier pour gérer les traductions de descriptions de paquets ; les traducteurs peuvent utiliser le projet de traduction des descriptions de Debian (DDTP).

For `debconf` templates, maintainers should use the `po-debconf` package to ease the work of translators. Some statistics can be found on the Central Debian translation statistics site.

Pour les pages web, chaque équipe l10n a accès au système de gestion de versions correspondant et les statistiques sont disponibles sur le site central des statistiques de traduction Debian.

Pour la documentation globale à propos de Debian, le processus est plus ou moins le même que pour les pages web (les traducteurs ont accès au système de gestion de versions), mais il n'y a pas de page de statistiques.

Another part of i18n work is package-specific documentation (man pages, info documents, other formats). At least the man page translations are po-based as most other things mentioned above.

8.2 FAQ I18N et L10N pour les responsables

Voici une liste des problèmes que les responsables peuvent rencontrer concernant l'i18n et la l10n. Lorsque vous lirez cela, gardez à l'esprit qu'il n'y a pas de consensus sur ces points au sein de Debian et que ce ne sont que des conseils. Si vous avez une meilleure idée pour un problème donné ou si vous êtes en désaccord avec certains points, vous êtes libre de fournir vos impressions pour que ce document puisse être amélioré.

8.2.1 Comment faire en sorte qu'un texte soit traduit

To translate package descriptions, you have nothing to do; the DDTP infrastructure will dispatch the material to translate to volunteers with no need for interaction on your part.

For all other material (`debconf` templates, gettext files, man pages, or other documentation), the best solution is to ask on debian-i18n for a translation in different languages. Some translation team members are subscribed to this list, and they will take care of the needed coordination, to get the material translated and reviewed. Once they are done, you will get your translated document from them in your mailbox or via a wishlist bugreport. It is also recommended, to use the `po-debconf` tools for i18n integration.

8.2.2 Comment faire en sorte qu'une traduction donnée soit relue

De temps en temps, des personnes indépendantes traduiront certains textes inclus dans votre paquet et vous demanderont d'inclure la traduction dans le paquet. Cela peut devenir problématique si vous n'êtes pas familier avec la langue donnée. C'est une bonne idée d'envoyer le document à la liste de diffusion l10n correspondante en demandant une relecture. Une fois celle-ci faite, vous pourrez avoir une meilleure confiance en la qualité de la traduction et l'inclure sans crainte dans votre paquet.

8.2.3 Comment faire en sorte qu'une traduction donnée soit mise à jour

Si vous avez certaines traductions d'un texte donné qui traînent, chaque fois que vous mettez à jour l'original, vous devriez demander au précédent traducteur de mettre à jour sa traduction avec vos nouveaux changements. Gardez à l'esprit que cette tâche demande du temps; au moins une semaine pour obtenir une mise à jour relue.

Si le traducteur ne répond pas, vous pouvez demander de l'aide sur la liste de diffusion correspondante. Si tout échoue, n'oubliez pas de mettre un avertissement dans le document traduit, indiquant que la traduction est un peu obsolète et que le lecteur devrait se référer au document d'origine si possible.

Évitez de supprimer complètement une traduction à cause de son obsolescence. Un vieux document est souvent mieux que pas de documentation du tout pour les personnes non anglophones.

8.2.4 Comment gérer un rapport de bogue concernant une traduction

La meilleure solution peut être de marquer le bogue comme transmis au développeur amont (`forwarded`) et de faire suivre le bogue à la fois au précédent traducteur et à son équipe (en utilisant la liste de diffusion debian-l10n-XXX correspondante).

8.3 FAQ I18N et L10N pour les traducteurs

Lorsque vous lirez cela, gardez à l'esprit qu'il n'y a pas de procédure générale dans Debian concernant ces points et que, dans tous les cas, vous devriez collaborer avec votre équipe et les responsables des paquets.

8.3.1 Comment aider l'effort de traduction

Choisissez ce que vous désirez traduire, assurez-vous que personne ne travaille déjà dessus (en utilisant votre liste de diffusion debian-l10n-XXX), traduisez-le, faites-le relire par d'autres personnes dont c'est également la langue maternelle sur votre liste de diffusion l10n et fournissez-le au responsable du paquet (voir le point suivant).

8.3.2 Comment fournir une traduction pour inclusion dans un paquet

Assurez-vous que votre traduction est correcte (en demandant une relecture sur votre liste de discussion l10n) avant de la fournir pour inclusion. Cela fera gagner du temps à tout le monde et évitera le chaos qui résulterait d'avoir plusieurs versions du même document dans les rapports de bogue.

La meilleure solution est de créer un rapport de bogue standard contenant la traduction sur le paquet. Assurez-vous d'utiliser les deux étiquettes `patch` et `l10n`, et n'utilisez pas une gravité supérieure à `wishlist`, car l'absence de traduction n'a jamais empêché un programme de fonctionner.

8.4 Meilleures pratiques actuelles concernant la l10n

— En tant que responsable, ne modifiez jamais les traductions en aucune façon (même pour reformater l'affichage) sans demander à la liste de diffusion l10n correspondante. Vous risquez, par exemple, de casser l'encodage du fichier en agissant ainsi. De plus, ce que vous considérez comme une erreur peut être correct (ou même nécessaire) pour une langue donnée.
— En tant que traducteur, si vous trouvez une erreur dans le texte d'origine, assurez-vous de l'indiquer. Les traducteurs sont souvent les lecteurs les plus attentifs d'un texte donné et s'ils ne signalent pas les erreurs découvertes, personne ne le fera.
— Dans tous les cas, rappelez-vous que le problème principal avec la l10n est qu'elle demande la coopération de plusieurs personnes et qu'il est très facile de démarrer une guerre incendiaire à propos de petits problèmes dus à des incompréhensions. Donc, si vous avez des problèmes avec votre interlocuteur, demandez de l'aide sur la liste de diffusion l10n correspondante, sur debian-i18n ou même sur debian-devel (attention, cependant, les discussions sur la l10n tournent très souvent à l'incendie sur cette liste :)
— En tous cas, la coopération ne peut être atteinte qu'avec un **respect mutuel**.

CHAPITRE 9

Aperçu des outils du responsable Debian

Cette section contient un aperçu rapide des outils dont dispose le responsable. Cette liste n'est ni complète, ni définitive, il s'agit juste d'un guide des outils les plus utilisés.

Les outils du responsable Debian sont destinés à aider les responsables et libérer leur temps pour des tâches plus cruciales. Comme le dit Larry Wall, « il y a plus d'une façon de le faire ».

Certaines personnes préfèrent utiliser des outils de haut niveau, d'autres pas. Debian n'a pas de position officielle sur la question ; tout outil conviendra du moment qu'il fait le boulot. C'est pourquoi cette section n'a pas été conçue pour indiquer à chacun quel outil il doit utiliser ou comment il devrait faire pour gérer sa charge de responsable. Elle n'est pas non plus destinée à favoriser l'utilisation d'un outil aux dépens d'un autre.

Most of the descriptions of these packages come from the actual package descriptions themselves. Further information can be found in the package documentation itself. You can also see more info with the command `apt-cache show` *package-name*.

9.1 Outils de base

Les outils suivants sont pratiquement nécessaires à tout responsable.

9.1.1 dpkg-dev

`dpkg-dev` contient les outils (y compris `dpkg-source`) nécessaires pour dépaqueter, construire, et envoyer les paquets source Debian. Ces utilitaires fournissent les fonctionnalités de bas niveau indispensables pour créer et manipuler les paquets ; en tant que tels, ils sont essentiels à tout responsable Debian.

9.1.2 debconf

`debconf` fournit une interface unifiée pour configurer les paquets de façon interactive. Il est indépendant de l'interface et permet une configuration en mode texte, par une interface HTML ou par boîtes de dialogue. D'autres types d'interface peuvent être ajoutés sous forme de modules.

Vous en trouverez la documentation dans le paquet `debconf-doc`.

Beaucoup pensent que ce système devrait être utilisé pour tout paquet nécessitant une configuration interactive, cf. *Gestion de la configuration avec debconf*. `debconf` n'est pas requis par la Charte Debian pour le moment, mais cela pourrait changer.

9.1.3 fakeroot

fakeroot simule les privilèges de root. Cela permet de fabriquer un paquet sans être root (en général, les paquets installent des fichiers appartenant à root). Si vous avez installé fakeroot, dpkg-buildpackage l'utilisera automatiquement.

9.2 Contrôle de paquets (« lint »)

Selon le « Free On-line Dictionary of Computing » (FOLDOC), « lint » est « un outil de traitement de langage C qui contient beaucoup plus de tests complets sur le code qu'ont habituellement les compilateurs C ». Les outils de contrôle de paquets aident les responsables à découvrir automatiquement les problèmes habituels et les violations de Charte dans leurs paquets.

9.2.1 lintian

lintian dissèque les paquets pour y repérer des bogues et des manquements aux règles de développement. Il contient des tests automatisés pour vérifier de nombreuses règles et quelques erreurs courantes.

Vous devriez récupérer la dernière version de lintian depuis unstable régulièrement et vérifier tous vos paquets. Notez que l'option -i donne des explications détaillées sur la signification de chaque erreur, la partie concernée dans la Charte et le moyen habituel de régler le problème.

Voir *Tests du paquet* pour plus d'informations sur comment et quand utiliser lintian.

You can also see a summary of all problems reported by Lintian on your packages at https://udd.debian.org/lintian/. The reports contain the latest lintian output for the whole development distribution (unstable).

9.2.2 lintian-brush

lintian-brush fournit un ensemble de scripts qui peuvent corriger automatiquement plus de 80 problèmes trouvés par lintian dans les paquets Debian.

Il est fournit avec un script d'enveloppe qui appelle les scripts, met à jour le journal des modifications (si on le souhaite) et envoie chaque révision à la gestion de version.

9.2.3 piuparts

piuparts est l'outil de test d'installation, de mise à niveau et de retrait de paquets .deb.

piuparts s'assure que les paquets .deb gèrent correctement leur installation, leur mise à niveau et leur retrait. Il le fait en créant une installation minimale de Debian dans un chroot et en installant, mettant à niveau et retirant les paquets dans cet environnement, puis en comparant l'état de l'arborescence des répertoires avant et après. piuparts rapporte tous les fichiers qui ont été ajoutés, retirés ou modifiés durant ce processus.

piuparts est censé être un outil d'assurance qualité pour les gens qui créent des paquets .deb afin de les tester avant leur envoi dans l'archive Debian.

9.2.4 debdiff

debdiff (du paquet devscripts, *devscripts*) compare les listes de fichiers ainsi que les fichiers de contrôle de deux paquets. C'est un simple test de régression qui peut aider à remarquer si le nombre de paquets binaires a changé depuis le dernier envoi ou si autre chose a changé dans le fichier de contrôle. Bien sûr, certains des changements indiqués sont normaux, mais cela peut aider à empêcher différents accidents.

Vous pouvez l'exécuter sur un couple de paquets binaires :

```
debdiff package_1-1_arch.deb package_2-1_arch.deb
```

Ou même sur un couple de fichiers de changements :

```
debdiff package_1-1_arch.changes package_2-1_arch.changes
```

Pour plus d'informations, veuillez consulter debdiff 1.

9.2.5 diffoscope

diffoscope fournit une comparaison en profondeur de fichiers, d'archives et de répertoires.

diffoscope essaie d'explorer à fond ce qui rend différents des fichiers ou des répertoires. Il décompresse les archives de façon récursive de toute nature et transforme divers formats binaires en une forme plus lisible pour les comparer.

Développé à l'origine pour comparer deux fichiers .deb ou deux fichiers changes, il peut maintenant comparer deux fichiers d'archives, des images ISO ou des PDF aussi facilement et prend en charge un grand éventail de types de fichier.

Les différences peuvent être affichées dans des rapports au format texte ou HTML, ou comme une sortie JSON.

9.2.6 duck

duck, le « Debian Url CheckKer » (vérificateur d'URL de Debian), traite plusieurs champs des fichiers debian/control, debian/upstream, debian/copyright, debian/patches/* et systemd.unit, et vérifie la validité des URL, des liens VCS et des domaines d'adresse de courrier électronique qui s'y trouvent.

9.2.7 adequate

adequate checks packages installed on the system and reports policy violations.

Les vérifications suivantes sont actuellement implémentées :
- liens symboliques cassés
- fichier de copyright absent
- fichiers de configuration obsolètes
- modules Python pas compilés en pseudocode
- bibliothèques absentes, symboles non définis, non correspondances de symboles de taille
- collisions de noms de programme
- alternatives absentes
- interpréteurs et détecteurs de binfmt absents
- dépendances à pkg-config absentes
- invalid user/group values in systemd/D-Bus/sysvinit files.

See /usr/share/doc/adequate/README on how to have adequate be ran automatically using autopkgtest.

9.2.8 i18nspector

i18nspector est un outil pour rechercher les problèmes courants dans les fichiers de modèles de traduction (POT), de catalogues de messages (PO) et de catalogues de messages compilés (MO).

9.2.9 cme

cme is a configuration editor. With libconfig-model-dpkg-perl installed, it can edit and validate Debian source package files.

9.2.10 licensecheck

licensecheck tente de déterminer la licence qui s'applique aux fichiers qui lui sont soumis en recherchant au début du fichier des textes appartenant à diverses licences.

9.2.11 blhc

blhc est un outil qui vérifie les journaux de constructions à la recherche des attributs de renforcement (« hardening ») manquants.

9.3 Assistance pour debian/rules

Des outils de construction de paquets facilitent le processus d'écriture du fichier debian/rules. *Scripts d'assistance* contient plus d'informations sur l'intérêt de les utiliser ou non.

9.3.1 debhelper

debhelper regroupe un ensemble de programmes pouvant être utilisés dans debian/rules pour automatiser les tâches courantes relatives à la construction de paquets Debian binaires. debhelper inclut des programmes pour installer différents fichiers, les compresser, ajuster leurs droits et intégrer votre paquet dans le système de menu Debian.

À la différence d'autres approches, debhelper est divisé en plusieurs petits utilitaires simples qui agissent de manière cohérente. Ce découpage permet un contrôle des opérations plus fin que certains des autres outils pour debian/rules.

Il existe aussi un certain nombre de petites extensions debhelper trop éphémères pour être documentées ici. La plupart seront listées avec apt-cache search ^dh-.

Lors du choix du niveau de compatibilité de debhelper de votre paquet, il conviendrait de choisir le niveau de compatibilité le plus élevé pris en charge par la version stable la plus récente. L'utilisation d'un niveau de compatibilité plus élevé est réservée aux cas où des fonctionnalités particulières sont nécessaires et ne sont fournies que par ce niveau de compatibilité.

In the past the compatibility level was defined in debian/compat, however nowadays it is much better to not use that but rather to use a versioned build-dependency like debhelper-compat (=12).

9.3.2 dh-make

dh-make contient dh_make, un programme qui crée un squelette de fichiers nécessaires à la construction d'un paquet Debian à partir d'une arborescence source. Comme le nom le suggère, dh_make est une réécriture de debmake et ses fichiers modèles utilisent les programmes dh_* de debhelper.

Quoique les fichiers de règles fabriqués par dh_make constituent en général une base suffisante pour un paquet fonctionnel, ce ne sont que les fondations : la charge incombe toujours au responsable d'affiner les fichiers générés et de rendre le paquet complètement fonctionnel et en conformité avec la Charte.

9.3.3 equivs

equivs est encore un assistant. Il est souvent conseillé pour un usage local, pour faire un paquet qui satisfasse des dépendances. Il est aussi parfois utilisé pour faire des « métapaquets », dont l'unique objet est de dépendre d'autres paquets.

9.4 Construction de paquets

Les paquets suivants facilitent le processus de construction des paquets, en contrôlant globalement dpkg-buildpackage ainsi que la gestion des tâches.

9.4.1 git-buildpackage

git-buildpackage permet de mettre à jour ou de récupérer des paquets source dans un référentiel Git, il permet de fabriquer un paquet Debian depuis le dépôt Git et assiste le responsable lors de l'intégration de modifications amont dans le dépôt.

Ce paquet fournit l'infrastructure facilitant l'utilisation de Git pour le responsable Debian. Il permet de conserver des branches Git distinctes pour les distributions stable, unstable et éventuellement experimental et de bénéficier des avantages d'un système de gestion de versions.

9.4.2 debootstrap

debootstrap permet d'amorcer un système Debian de base à n'importe quel endroit de votre système de fichiers. « Système de base » signifie ici le strict minimum de paquets nécessaires pour fonctionner et installer le reste du système.

Un système comme celui-ci peut être utilisé de nombreuses façons différentes. Par exemple, avec chroot, vous pouvez y tester les dépendances de construction. Vous pouvez aussi vérifier le comportement d'un paquet installé dans un environnement minimal. Les automates de constructions « chrootés » utilisent ce paquet ; voir ci-après.

9.4.3 pbuilder

pbuilder construit un système « chrooté » et compile des paquets dans ce système. C'est très pratique pour vérifier que les dépendances de compilation d'un paquet sont correctes et pour s'assurer qu'aucune dépendance de construction inutile ou incorrecte n'existe dans le paquet résultant.

cowbuilder est un outil similaire, qui accélère le processus de construction en utilisant un système de fichiers COW sur n'importe quel système de fichiers standard Linux.

9.4.4 sbuild

sbuild est un autre compilateur automatique. Il peut également être utilisé dans un environnement « chrooté ». Il peut être utilisé seul ou comme partie d'un environnement de compilation distribué en réseau. Comme le précédent, il fait partie du système utilisé par les porteurs pour construire des paquets binaires pour toutes les architectures disponibles. Voir *wanna-build* pour plus d'informations et https://buildd.debian.org/pour voir le système en fonctionnement.

9.5 Envoi de paquets

Les paquets suivants aident à automatiser ou simplifier le processus d'envoi de paquets dans l'archive officielle.

9.5.1 dupload

dupload is a package and a script to automatically upload Debian packages to the Debian archive, to log the upload, and to optionally send mail about the upload of a package. It supports various kinds of hooks to extend its functionality, and can be configured for new upload locations or methods, although by default it provides various hooks performing checks and comes configured with all Debian upload locations.

9.5.2 dput

The dput package and script do much the same thing as dupload, but in a different way. Out of the box it supports to run dinstall in dry-run mode after the upload.

9.5.3 dcut

dcut (du paquet dput, *dput*) permet de supprimer des fichiers du répertoire d'envoi FTP.

9.6 Automatisation de la maintenance

Les outils suivants permettent d'automatiser les différentes tâches de maintenance en ajoutant des entrées au journal des modifications ou des lignes de signatures, en cherchant des bogues depuis Emacs et en utilisant le fichier officiel config.sub le plus récent.

9.6.1 devscripts

devscripts is a package containing wrappers and tools that are very helpful for maintaining your Debian packages. Example scripts include debchange (or its alias, dch), which manipulates your debian/changelog file from the command-line, and debuild, which is a wrapper around dpkg-buildpackage. The bts utility is also very helpful to update the state of bug reports on the command line. uscan can be used to watch for new upstream

versions of your packages (see https://wiki.debian.org/debian/watch for more info on that). `suspicious-source` outputs a list of files which are not common source files.

See the devscripts(7) manual page for a complete list of available scripts.

9.6.2 reportbug

`reportbug` est un outil conçu pour rendre relativement faciles les rapports de bogues dans Debian et dans les distributions dérivées. Parmi ces fonctionnalités :

- intégration à mutt et aux lecteurs de mail mh/nmh ;
- accès aux rapports de bogues non résolus pour faciliter l'identification des problèmes qui ont été déjà rapportés ;
- vérification automatique des dernières versions des paquets.

`reportbug` est conçu pour une utilisation sur des systèmes où un agent de transport de courrier électronique est installé ; néanmoins, on peut éditer le fichier de configuration et envoyer les rapports avec n'importe quel serveur de courrier électronique.

Ce paquet fournit aussi le script `querybts` pour naviguer dans le système de suivi des bogues de Debian.

9.6.3 autotools-dev

`autotools-dev` contient les meilleurs pratiques pour les responsables des paquets qui utilisent `autoconf` ou `automake`. Il contient également les fichiers canoniques `config.sub` et `config.guess`, connus pour fonctionner avec tous les portages Debian.

9.6.4 dpkg-repack

`dpkg-repack` crée un paquet Debian à partir d'un paquet déjà installé. Si des changements ont été effectués sur le paquet depuis qu'il a été installé (des fichiers de `/etc` modifiés par exemple), le nouveau paquet héritera de ces changements.

Cet utilitaire peut faciliter la copie de paquet d'un ordinateur à un autre, ou la recréation de paquets installés sur un système qui ne sont plus disponibles ailleurs, ou pour sauvegarder l'état actuel d'un paquet avant de le mettre à niveau.

9.6.5 alien

`alien` convertit des paquets binaires entre différents formats de paquets, y compris des paquets Debian, RPM (RedHat), LSB (Linux Standard Base), Solaris et Slackware.

9.6.6 elpa-dpkg-dev-el

`elpa-dpkg-dev-el` is an Emacs lisp package that provides assistance when editing some of the files in the `debian` directory of your package. For instance, there are handy functions for listing a package's current bugs, and for finalizing the latest entry in a `debian/changelog` file.

9.6.7 dpkg-depcheck

`dpkg-depcheck` (du paquet `devscripts`, *devscripts*) exécute une commande sous `strace` pour déterminer tous les paquets utilisés par la commande.

Pour les paquets Debian, c'est utile pour créer une ligne `Build-Depends` d'un nouveau paquet : exécuter le processus de compilation avec `dpkg-depcheck` fournira une bonne première approximation des dépendances de compilation. Par exemple :

```
dpkg-depcheck -b debian/rules build
```

`dpkg-depcheck` peut aussi être utilisé pour vérifier les dépendances d'exécution, d'autant plus si le paquet utilise exec 2 pour exécuter d'autres programmes.

Pour plus d'informations, veuillez voir dpkg-depcheck 1.

9.6.8 debputy

The debputy tools is new since 2024. While its main purpose is to offer a new Debian package build paradigm, it includes subcommands that can be used on any existing Debian package to validate the correctness of most of the files in debian/*, and in many cases also automatically fix them.

To check correctness of files in debian/* run :

```
debputy lint --spellcheck
```

To format debian/control and debian/tests/control files

```
debputy reformat --style black
```

Using the reformat command obsoletes using wrap-and-sort -ast.

The debputy tool also includes a language server which, when integrated with a code editor, can give real-time feedback on the correctness of files in debian/* while editing them.

For more information please see debputy 1.

9.7 Outils de portage

Les outils suivants sont pratiques pour les porteurs et la compilation croisée (cross-compilation).

9.7.1 dpkg-cross

dpkg-cross est un outil qui installe les bibliothèques et les en-têtes nécessaires à une compilation croisée d'une manière similaire à dpkg. De plus, les fonctionnalités de dpkg-buildpackage et dpkg-shlibdeps ont été améliorées pour accepter les compilations croisées.

9.8 Documentation et information

Les paquets suivants fournissent des informations pour les responsables ou de l'aide pour construire de la documentation.

9.8.1 debian-policy

Le paquet debian-policy fournit la charte Debian (Debian Policy Manual) ainsi que les documents qui lui sont associés et qui sont :

- la charte Debian
- la norme d'organisation du système de fichiers (FHS) ;
- les règles pour Menu dans Debian ;
- les règles pour Perl dans Debian ;
- la spécification de gestion de configuration pour Debian ;
- La spécification lisible par une machine des fichiers debian/copyright ;
- autopkgtest : tests automatiques de paquets Debian ;
- la liste officielle des noms de paquets virtuels ;
- la liste de contrôle de la Charte pour mettre à niveau les paquets.

Le manuel de la charte Debian contientt l'ensemble des règles relatives aux paquets et des détails du mécanisme de l'empaquetage. Il couvre tout, des options requises de gcc à la manière dont les scripts du responsable (postinst, etc.) fonctionnent, aux sections et priorités des paquets, etc.

Le fichier /usr/share/doc/debian-policy/upgrading-checklist.txt.gz qui liste les modifications entre les versions de la charte est aussi pertinent.

9.8.2 doc-debian

doc-debian fournit une nombreuse documentation utile spécifique à Debian :

— le Manifeste Linux de Debian ;
— la constitution du projet Debian ;
— le contrat social de Debian ;
— les principes du logiciel libre selon Debian ;
— la documentation du système de suivi des bogues Debian ;
— l'introduction aux listes de diffusion de Debian.

9.8.3 developers-reference

Le paquet developers-reference fournit le document que vous êtes en train de lire, le guide de référence du développeur pour Debian, un ensemble de règles et de bonnes pratiques qui ont été établies par et pour la communauté des développeurs Debian.

9.8.4 maint-guide

Le paquet maint-guide fournit le guide du nouveau responsable Debian.

Ce document tente de décrire aux utilisateurs ordinaires de Debian et aux développeurs en devenir, la construction d'un paquet Debian. Il utilise un langage peu technique et est complété par des exemples pratiques.

9.8.5 debmake-doc

The debmake-doc package contains the Guide for Debian Maintainers.

This document is newer than Debian New Maintainers' Guide and intends to replace it. The Guide for Debian Maintainers caters to those learning Debian packaging and covers a wide range of topics and tools, along with plenty of examples about various types of packaging issues.

9.8.6 packaging-tutorial

Ce tutoriel est une introduction à la construction de paquet Debian. Il apprend aux développeurs en devenir comment modifier un paquet existant, comment créer ses propres paquets et comment interagir avec la communauté Debian.

En plus du tutoriel principal, il fournit trois exercices pratiques sur la modification du paquet grep, l'empaquetage du jeu gnujump et enfin sur l'empaquetage d'une bibliothèque Java.

9.8.7 how-can-i-help

how-can-i-help montre les possibilités de contribuer à Debian. Il est ancré à APT pour lister ces opportunités de contributions à Debian sur des paquets installés localement (paquets orphelins, bogues marqués « newcomer » — pour débutant) après chaque invocation d'APT. Le programme peut aussi bien être exécuté directement et, dans ce cas, il liste la totalité des contributions possibles (et pas seulement les plus récentes).

9.8.8 docbook-xml

docbook-xml provides the DocBook XML DTDs, which are commonly used for Debian documentation (as is the older debiandoc SGML DTD).

docbook-xsl fournit les fichiers XSL pour construire et décliner les sources en de multiples formats de sortie. Vous devriez utiliser un processeur de ligne de commande XSLT, tel que xsltproc, pour utiliser les feuilles de style XSL. La documentation des feuilles de style est disponible dans les nombreux paquets docbook-xsl-doc-*.

To produce PDF from FO, you need an FO processor, such fop or dblatex.

9.8.9 debiandoc-sgml

debiandoc-sgml fournit la définition de type de document (Document Type Definition ou DTD) SGML pour DebianDoc, souvent utilisé pour la documentation Debian, mais est maintenant déconseillé (docbook-xml ou python3-sphinx devraient être utilisés à la place).

9.8.10 debian-keyring

Contains the public OpenPGP keys of Debian Developers and Maintainers. See *Gestion de clé publique* and the package documentation for more information.

9.8.11 elpa-debian-el

elpa-debian-el provides an Emacs mode for viewing Debian binary packages. This lets you examine a package without unpacking it.