Référence du développeur Debian

Équipe de la référence du développeur, Andreas Barth,
Adam Di Carlo, Raphaël Hertzog, Lucas Nussbaum,
Christian Schwarz, et Ian Jackson

25 septembre 2018

Référence du développeur Debian

by Équipe de la référence du développeur, Andreas Barth, Adam Di Carlo, Raphaël Hertzog, Lucas Nussbaum, Christian Schwarz, et Ian Jackson

Published 2018-09-25

If you want to print this reference, you should use the pdf version. This page is also available in French, German, Italian, Russian, and Japanese.

Table des matières

Chapitre 1

Portée de ce document

Le but de ce document est de donner une vue d'ensemble des procédures à suivre et des ressources mises à la disposition des développeurs Debian.

Les procédures décrites ci-après expliquent comment devenir responsable Debian (Chapitre 2), comment créer de nouveaux paquets (Section 5.1), comment envoyer des paquets dans l'archive (Section 5.6), comment gérer les rapports de bogues (Section 5.8), comment déplacer, effacer ou abandonner un paquet (Section 5.9), comment faire le portage d'un paquet (Section 5.10), quand et comment faire la mise à jour du paquet d'un autre responsable (Section 5.11).

Ce manuel présente entre autres les listes de diffusion (Section 4.1) et les serveurs (Section 4.4), la structure de l'archive Debian (Section 4.6), des explications sur les serveurs qui acceptent l'envoi de paquets (Section 5.6.1) et une présentation des outils qui peuvent aider un responsable à améliorer la qualité de ses paquets (Annexe A).

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

Candidature de responsable Debian

2.1 Entrée en matière

So, you've read all the documentation, you've gone through the Debian New Maintainers' Guide (or its successor, Guide for Debian Maintainers), understand what everything in the `hello` example package is for, and you're about to Debianize your favorite piece of software. How do you actually become a Debian developer so that your work can be incorporated into the Project?

Firstly, subscribe to debian-devel@lists.debian.org if you haven't already. Send the word `subscribe` in the `Subject` of an email to debian-devel-REQUEST@lists.debian.org. In case of problems, contact the list administrator at listmaster@lists.debian.org. More information on available mailing lists can be found in Section 4.1. debian-devel-announce@lists.debian.org is another list, which is mandatory for anyone who wishes to follow Debian's development.

Vous devriez suivre les discussions de cette liste (sans poster) pendant quelque temps avant de coder quoi que ce soit et 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 Section 2.2 pour les détails. Le canal IRC `#debian` pourra aussi être utile ; voir Section 4.2.

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 Section 7.5.1 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.

Évitez d'avoir la partie locale de votre adresse électronique trop générique : des termes comme mail, admin, root, master ou debian devraient être évités. Veuillez consulter `https://www.debian.org/MailingLists/` pour plus de détails.

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 Section 4.1 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.

De plus, si vous avez des paquets prêts à être inclus dans Debian, mais que vous attendez que votre demande pour devenir responsable soit acceptée, vous pouvez trouver un parrain pour envoyer vos paquets pour vous. Les parrains sont des développeurs Debian officiels qui sont volontaires pour critiquer et envoyer vos paquets pour vous. Veuillez lire en premier la FAQ de debian-mentors à `https://wiki.debian.org/DebianMentorsFaq`.

Pour devenir mentor ou parrain, plus d'informations sont disponibles en Section 7.5.

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 developer. For example, before you apply, you have to read the Debian Social Contract. Registering as a developer means that you agree with and pledge to uphold the Debian Social Contract; it is very important that maintainers 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. Le nombre de personnes travaillant pour Debian a atteint 1000 et 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 !

Pour votre candidature, vous devrez être familiarisé avec la philosophie du projet Debian et avec sa documentation technique. Il vous faudra aussi une clé GnuPG signée par un responsable Debian. Si votre clé GnuPG n'est pas encore signée, vous devriez essayer de rencontrer un responsable Debian pour le faire. La page de coordination des signatures de clé GnuPG devrait aider à trouver un responsable Debian près de chez vous. (S'il n'y a pas de responsable près de chez vous, il peut y avoir des moyens alternatifs pour valider votre identité en tant qu'exception absolue étudiée au cas par cas. Reportez-vous à la page d'identification pour en savoir plus.)

If you do not have an OpenPGP key yet, generate one. Every developer needs an OpenPGP key in order to sign and verify package uploads. You should read the manual for the software you are using, since it has much important information that is critical to its security. Many more security failures are due to human error than to software failure or high-powered spy techniques. See Section 3.2.2 for more information on maintaining your public key.

Debian utilise `GNU Privacy Guard` (paquet `gnupg` version 1 ou supérieure) comme standard de base. Vous pouvez aussi utiliser une autre implémentation d'OpenPGP. OpenPGP est un standard ouvert basé sur la RFC 2440.

You need a version 4 key for use in Debian Development. Your key length must be greater than 2048 bits (4096 bits is preferred); there is no reason to use a smaller key, and doing so would be much less secure. [1]

Si votre clé publique n'est pas sur un serveur public tel que `subkeys.pgp.net`, reportez-vous à la documentation disponible à 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 authentification est interdite, contactez-nous pour que nous prenions des dispositions particulières.

Pour faire acte de candidature, il vous faut un responsable Debian qui soutiendra votre candidature (un intercesseur ou « `advocate` » en anglais). Après avoir contribué au projet Debian pendant un 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.

Une fois trouvé un intercesseur, votre clé GnuPG signée et que vous avez déjà contribué au projet, vous êtes prêt à faire acte de candidature. Il vous suffit pour cela de vous enregistrer sur la page de candidature. Ensuite, votre intercesseur devra confirmer votre candidature. Quand il aura accompli cette tâche, un responsable de candidature (« `application manager` ») sera désigné pour vous accompagner dans le processus d'enregistrement. Vous

1. Les clés en version 4 sont conformes au standard OpenPGP défini dans la RFC 2440. La version 4 est le type de clé qui a toujours été créé avec GnuPG. Les versions de PGP depuis la version 5.x peuvent également créer des clés version 4, l'autre choix ayant été des clés compatibles pgp 2.6.x (également appelées « legacy RSA » par PGP).

Les clés (primaires) en version 4 peuvent soit utiliser l'algorithme RSA, soit l'algorithme DSA, cela n'a donc rien à voir avec la question de GnuPG à propos de la question du type de clé que vous désirez : (1) DSA et Elgamal, (2) DSA (signature seule), (5) RSA (signature seule). Si vous n'avez pas des besoins spécifiques, choisissez simplement la valeur par défaut.

Le moyen le plus simple de dire si une clé existante est une clé v4 ou une clé v3 (ou v2) est de regarder son empreinte : les empreintes des clés en version 4 sont des sommes de contrôle SHA-1 d'une partie de la clé, il s'agit donc d'une suite de 40 chiffres hexadécimaux, habituellement groupés par blocs de quatre. Les empreintes des anciennes versions de clé utilisaient MD5 et sont généralement affichées par blocs de 2 chiffres hexadécimaux. Par exemple, si votre empreinte ressemble à `5B00 C96D 5D54 AEE1 206B  AF84 DE7A AF6E 94C0 9C7F` alors il s'agit d'une clé v4.

Une autre possibilité est d'envoyer la clé dans **pgpdump**, qui dira quelque chose comme « Public Key Packet - Ver 4 ».

Remarquez également que votre clé doit être auto-signée (c'est-à-dire qu'elle doit signer tous ses propres identifiants d'utilisateur ; cela empêche la falsification d'identité). Tous les logiciels OpenPGP modernes font cela automatiquement, mais si vous avez une ancienne clé, il se peut que vous deviez ajouter manuellement ces signatures.

pouvez toujours consulter le tableau de bord des candidatures pour connaître l'état de votre candidature.

For more details, please consult New Members Corner at the Debian web site. Make sure that you are familiar with the necessary steps of the New Maintainer process before actually applying. If you are well prepared, you can save a lot of time later on.

Chapitre 3

Devoirs du développeur Debian

3.1 Devoirs du responsable de paquet

As a package maintainer, you're supposed to provide high-quality packages that are well integrated into the system and that adhere to the Debian Policy.

3.1.1 Œuvrer pour la prochaine publication `stable`

Providing high-quality packages in `unstable` is not enough; most users will only benefit from your packages when they are released as part of the next `stable` release. You are thus expected to collaborate with the release team to ensure your packages get included.

Plus concrètement, vous devriez surveiller si vos paquets migrent vers `testing` (consultez Section 5.14). 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 Section 5.8.5). 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 Section 5.5.1).

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 Section 5.8. 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.

Si pour une raison ou une autre, vous ne pouvez pas corriger un bogue critique pour la publication dans un de vos paquets en moins de deux semaines (par exemple à cause de contraintes de temps, ou parce que c'est compliqué à corriger) vous devriez le signaler clairement dans le rapport de bogue en l'étiquetant `help` pour encourager d'autres volontaires à s'impliquer. Sachez que les bogues critiques pour la publication sont souvent les cibles de mises à jour indépendantes (consultez Section 5.11) car ils peuvent bloquer la migration vers `testing` de plusieurs paquets.

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 Section 7.4).

3.1.4 Coordination avec les développeurs amont

Une grande part du travail de responsable Debian sera de rester en contact avec les développeurs amont. Parfois, les utilisateurs signalent des bogues qui ne sont pas spécifiques à Debian. Vous devez transmettre ces rapports de bogue aux développeurs amont pour qu'ils soient corrigés dans les versions suivantes.

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 responsables Debian proposent souvent des correctifs pour corriger des bogues amont, il vous faudra alors évaluer ce correctif puis le transmettre aux développeurs amont.

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.

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 Section 4.5.

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 Section 4.4). 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.

If you add signatures to your public key, or add user identities, you can update the Debian key ring by sending your key to the key server at `keyring.debian.org`. Updates are processed at least once a month by the `debian-keyring` package maintainers.

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 invalide, vous devez également ajouter le certification 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 `http://keyring.debian.org/replacing_keys.html`.

Les mêmes routines d'extraction de clé décrites en Section 2.3 s'appliquent.

You can find a more in-depth discussion of Debian key maintenance in the documentation of the `debian-keyring` package and the http://keyring.debian.org/ site.

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.

Vous n'avez pas besoin de suivre les discussions précédant le vote car le secrétaire enverra plusieurs appels au vote sur la liste debian-devel-announce@lists.debian.org (et tous les développeurs devraient être inscrits à cette liste). La démocratie ne fonctionne pas si les personnes ne prennent pas part au vote, c'est pourquoi nous encourageons tous les développeurs à voter. Le vote est conduit par messages signés ou chiffrés par GPG.

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 Départ en vacances poli

Il est courant pour les développeurs de s'absenter, 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 Section 5.11) 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 Debian LDAP (l'accès à cette information est réservé aux développeurs). N'oubliez pas de retirer l'indicateur `on vacation` à votre retour !

Dans l'idéal, vous devriez vous connecter sur les pages de coordination GPG quand vous prévoyez un départ et vérifier si quelqu'un recherche un échange de signatures. Cela est particulièrement important quand des personnes vont à des endroits exotiques où nous n'avons pas encore de développeurs, mais où il y a des personnes intéressées pour poser leur candidature.

3.2.5 Démission

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

1. abandonnez tous vos paquets comme décrit en Section 5.9.4 ;
2. Send an gpg-signed email announcing your retirement to debian-private@lists.debian.org.
3. Notify the Debian key ring maintainers that you are leaving by opening a ticket in Debian RT by sending a mail to keyring@rt.debian.org with the words "Debian RT" somewhere in the subject line (case doesn't matter).
4. 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.

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

3.2.6 Revenir après démission

Le compte d'un développeur est marqué « emeritus » (honoraire) quand le processus précédent en Section 3.2.5 est suivi, et « disabled » (désactivé) sinon. Un développeur ayant démissionné avec un compte « emeritus » peut réactiver son compte de la façon suivante :

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

— contacter da-manager@debian.org ;

— 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 ») ;

— démontrer qu'il possède toujours la clef GPG associée au compte, ou fournir des preuves d'identité sur une nouvelle clef GPG, avec au moins deux signatures d'autres développeurs Debian.

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

Chapitre 4

Ressources pour les responsables et développeurs Debian

In this chapter you will find a very brief roadmap of the Debian mailing lists, the Debian machines which may be available to you as a developer, and all the other resources that are available to help you in your maintainer work.

4.1 Listes de diffusion

Une grande partie des discussions entre les développeurs Debian (et les utilisateurs) a lieu dans un vaste 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-nomdalias sur master.debian.org, qui se traduit en une adresse raisonnablement agréable *vous-nomdalias@debian.org*) ou d'une liste de diffusion autogérée sur Alioth.

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)`. 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 semblables 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`.

There are other additional channels dedicated to specific subjects. `#debian-bugs` is used for coordinating bug squashing parties. `#debian-boot` is used to coordinate the work on the debian-installer. `#debian-doc` is occasionally used to talk about documentation, like the document you are reading. Other channels are dedicated to an architecture or a set of packages: `#debian-kde`, `#debian-dpkg`, `#debian-perl`, `#debian-python`...

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

Des canaux dédiés à Debian existent sur d'autres réseaux IRC, notamment sur le réseau IRC Freenode, sur lequel pointait l'alias `irc.debian.org` jusqu'au 4 juin 2006.

Pour obtenir un masquage (« `cloak` ») sur freenode, envoyez un message signé à Jörg Jaspert <joerg@debian.org> où vous indiquerez votre pseudonyme (« `nick` »). Indiquez « cloak » dans le sujet. Votre pseudonyme doit être enregistré conformément à la page de configuration des pseudonymes. Le message doit être signé avec une clé du porte-clés Debian. Veuillez consulter la documentation de Freenode sur les masquages pour plus d'informations.

4.3 Documentation

This document contains a lot of information which is useful to Debian developers, but it cannot contain everything. Most of the other interesting documents are linked from The Developers' Corner. Take the time to browse all the links; you will learn many more things.

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 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 Section 7.1 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`

The `ftp-master.debian.org` server holds the canonical copy of the Debian archive. Generally, packages uploaded to ftp.upload.debian.org end up on this server; see Section 5.6.

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 Section 5.9 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.

If you find a problem with the Debian web server, you should generally submit a bug against the pseudo-package `www.debian.org`. Remember to check whether or not someone else has already reported the problem to the Bug Tracking System.

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 Serveurs de gestion de versions (VCS)

Si vous avez besoin d'une gestion de versions (« Version Control System » ou VCS) pour tout travail relatif à Debian, vous pouvez utiliser un des dépôts existants hébergés sur Alioth (p. ex. le projet collab-maint) ou demander un nouveau projet avec le dépôt VCS de votre choix. Alioth gère CVS (cvs.alioth.debian.org/cvs.debian.org), Subversion (svn.debian.org), Arch (tla/baz, tous deux sur arch.debian.org), Bazaar (bzr.debian.org), Darcs (darcs.debian.org), Mercurial (hg.debian.org) et Git (git.debian.org). Consultez https://wiki.debian.org/Alioth/PackagingProject si vous avez l'intention de maintenir un paquet dans un dépôt VCS. Voir Section 4.12 pour plus d'informations sur les services fournis par Alioth.

4.4.6 Chroots de différentes distributions

Sur certaines machines, des chroots de différentes distributions sont disponibles. Vous pouvez les utiliser comme ceci :

```
vore$ dchroot unstable
Interpréteur de commande dans le chroot : /org/vore.debian.org/chroots/user/ ↩
    unstable
```

Dans chaque chroot, les répertoires normaux des utilisateurs sont disponibles. Vous pouvez trouver quels chroots sont disponibles sur https://db.debian.org/machines.cgi.

4.5 Base de données des développeurs

The Developers Database, at https://db.debian.org/, is an LDAP directory for managing Debian developer attributes. You can use this resource to search the list of Debian developers. Part of this information is also available through the finger service on Debian servers; try **finger yourlogin@db.debian.org** to see what it reports.

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

— l'adresse de suivi pour leur adresse debian.org ;

— 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 15000) 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/testing/
dists/testing/main/
     ...
dists/testing/contrib/
     ...
dists/testing/non-free/
     ...

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

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/f/
pool/non-free/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`.

Le répertoire `dists/stable` contient trois répertoires nommés `main`, `contrib`, et `non-free`.

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 de « 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`.

Les paquets qui ne sont pas conformes aux DFSG sont classés dans la section `non-free`. Bien qu'ils soient prêts à être utilisés sur un système Debian, et qu'ils bénéficient des infrastructures de Debian (système de suivi de bogues, listes de diffusion, etc.), ces paquets non libres ne font pas partie de la distribution Debian.

La Charte Debian (« `Debian Policy Manual` ») donne des définitions plus précises de ces trois sections. Les paragraphes précédents ne constituent qu'une introduction.

La séparation de l'archive en trois sections est importante pour toute personne qui désire distribuer Debian, que ce soit par Internet ou sur CD-ROM : il suffit de distribuer les sections `main` et `contrib` pour éviter tout problème légal. Certains paquets de la section `non-free` interdisent leur distribution à titre commercial par exemple.

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).

Note that the term section is also used to refer to categories which simplify the organization and browsing of available packages: `admin`, `net`, `utils`, etc. Once upon a time, these sections (subsections, rather) existed in the form of subdirectories within the Debian archive. Nowadays, these exist only in the Section header fields of packages.

4.6.2 Architectures

À ses débuts, le noyau Linux existait seulement pour les architectures Intel i386 (et compatible) ; 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 was only available as `i386`. Debian 2.0 shipped for `i386` and `m68k` architectures. Debian 2.1 shipped for the `i386`, `m68k`, `alpha`, and `sparc` architectures. Since then Debian has grown hugely. Debian 9 supports a total of ten Linux architectures (`amd64`, `arm64`, `armel`, `armhf`, `i386`, `mips`, `mips64el`, `mipsel`, `ppc64el`, and `s390x`) and two kFreeBSD architectures (`kfreebsd-i386` and `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) ».

If a package is developed specially for Debian and is not distributed outside of Debian, there is just one `.tar.{gz,bz2,xz}` file, which contains the sources of the program; it's called a "native" source package. If a package is distributed elsewhere too, the `.orig.tar.{gz,bz2,xz}` file stores the so-called `upstream source code`, that is the source code that's distributed by the `upstream maintainer` (often the author of the software). In this case, the `.diff.gz` or the `debian.tar.{gz,bz2,xz}` contains the changes made by the Debian maintainer.

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

The directory system described in the previous chapter is itself contained within `distribution directories`. Each distribution is actually contained in the `pool` directory in the top level of the Debian archive itself.

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 Section 5.14.

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 envois dans la distribution stable pour plus de détails.

Pendant la période de gel, les développements continuent sur la distribution `unstable` car cette distribution reste en place parallèlement à `testing`.

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 les informations à propos de 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://ftp.xy.debian.org/debian/ experimental main
deb-src http://ftp.xy.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

Every released Debian distribution has a code name: Debian 7.0 is called wheezy; Debian 8, jessie; Debian 9, stretch; the next release, Debian 10, will be called buster and Debian 10 will be called buster. There is also a pseudo-distribution", called sid, which is the current unstable distribution; since packages are moved from unstable to testing as they approach stability, sid itself is never released. As well as the usual contents of a Debian distribution, sid contains packages for architectures which are not yet officially supported or released by Debian. These architectures are planned to be integrated into the mainstream distribution at some future date. The codenames and versions for older releases are listed on the website.

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.)

Thus, the names of the distribution directories in the archive are determined by their code names and not their release status (e.g., stretch'). These names stay the same during the development period and after the release; symbolic links, which can be changed easily, indicate the currently released stable distribution. That's why the real distribution directories use the code names, while symbolic links for stable, testing, and unstable point to the appropriate release directories.

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.

Note that mirrors are generally run by third parties who are interested in helping Debian. As such, developers generally do not have accounts on these machines.

4.8 Système « Incoming »

Le système « Incoming » est responsable de la collecte des paquets mis à jour et 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 http://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.

Le logiciel de maintenance de l'archive enverra également le fichier .changes signé avec OpenPGP/GnuPG à la liste de diffusion appropriée. Pour un paquet avec le champ Distribution à « stable », l'annonce sera envoyée à debian-changes@lists.debian.org. Pour un paquet avec le champ Distribution à « unstable » ou « experimental », l'annonce sera plutôt envoyée à debian-devel-changes@lists.debian.org ou debian-experimental-changes@lists.debian.org.

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 disponibles de paquet 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 | 0.1.5-2sarge1 |     oldstable | source, alpha, arm, hppa, i386, ia64,  ↩
    m68k, mips, mipsel, powerpc, s390, sparc
evince |    0.4.0-5 |     etch-m68k | source, m68k
evince |    0.4.0-5 |        stable | source, alpha, amd64, arm, hppa, i386, ia64 ↩
    , mips, mipsel, powerpc, s390, sparc
evince |   2.20.2-1 |       testing | source
evince | 2.20.2-1+b1 |       testing | alpha, amd64, arm, armel, hppa, i386, ia64 ↩
    , mips, mipsel, powerpc, s390, sparc
evince |   2.22.2-1 |      unstable | source, alpha, amd64, arm, armel, hppa,  ↩
    i386, ia64, m68k, mips, mipsel, powerpc, s390, sparc
```

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.

4.10 The 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.

The package tracker has a web interface at https://tracker.debian.org/ that puts together a lot of information about each source package. It features many useful links (BTS, QA stats, contact information, DDTP translation status, buildd logs) and gathers much more information from various places (30 latest changelog entries, testing status, etc.). It's a very useful tool if you want to know what's going on with a specific source package. Furthermore, once authenticated, you can subscribe and unsubscribe from any package with a single click.

You can jump directly to the web page concerning a specific source package with a URL like `https://tracker.debian.org/p`

For more in-depth information, you should have a look at its documentation. Among other things, it explains you how to interact with it by email, how to filter the mails that it forwards, how to configure your VCS commit notifications, how to leverage its features for maintainer teams, 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`

Alioth is a Debian service based on a slightly modified version of the FusionForge software (which evolved from SourceForge and GForge). This software offers developers access to easy-to-use tools such as bug trackers, patch manager, project/task managers, file hosting services, mailing lists, VCS repositories, etc. All these tools are managed via a web interface.

Alioth a pour but de fournir une infrastructure pour des projets de logiciels libres soutenus ou dirigés par Debian, de faciliter les contributions de développeurs externes aux projets initiés par Debian et d'aider des projets dont les buts sont de promouvoir Debian ou ses dérivés. Il est largement utilisé par de nombreuses équipes et fournit l'hébergement pour toutes sortes de systèmes de gestion de versions.

Tous les développeurs Debian ont automatiquement un compte sur Alioth. Ils peuvent l'activer en utilisant la fonctionnalité de récupération des mots de passe. Les développeurs externes peuvent demander un compte invité sur Alioth.

Des informations supplémentaires sont disponibles sur les liens suivants :

- https://wiki.debian.org/Alioth
- https://wiki.debian.org/Alioth/FAQ
- https://wiki.debian.org/Alioth/PackagingProject
- https://alioth.debian.org/

4.13 Cadeaux pour les développeurs et responsables Debian

Les avantages auxquels les développeurs et responsables Debian peuvent prétendre sont décrits dans 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 Section 7.1) 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.

Le sujet du rapport de bogue pour déclarer votre intention d'empaqueter (« Intent To Package » ou ITP) devra être `ITP: NomDuPaquet -- description courte`, en remplaçant *NomDuPaquet* par le nom du paquet. La gravité du bogue sera wishlist. Si vous le jugez nécessaire, envoyez une copie à debian-devel@lists.debian.org en mettant cette adresse dans le champ X-Debbugs-CC de l'en-tête du message. N'utilisez pas le champ CC sinon le sujet du message ne contiendrait pas le numéro du bogue. Si vous empaquetez tellement de paquets (plus de dix) que les signaler sur la liste de diffusion soit trop perturbant, envoyez plutôt un résumé sur la liste debian-devel après avoir rempli les rapports de bogue. Cela informera les autres développeurs de l'arrivée de nouveaux paquets et permettra une relecture des description et nom de paquet.

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 Section 5.8.4).

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.

Veuillez consulter https://ftp-master.debian.org/REJECT-FAQ.html pour les raisons courantes de rejet des nouveaux paquets.

5.2 Enregistrement des modifications

Les modifications apportées au paquet doivent être consignées dans le fichier debian/changelog. 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 Section 5.5. 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 Section 5.8.4.

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

```
  * New upstream release.
```

Certains outils peuvent aider à éditer et finaliser le fichier changelog — voir Section A.6.1 et Section A.6.5. Voir aussi Section 6.3.

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 :

— installer le paquet et vérifier que le logiciel fonctionne. Si le paquet existait déjà dans une version plus ancienne, faire une mise à niveau ;

— exécuter **lintian** sur le paquet. Il est possible d'exécuter **lintian** comme suit : lintian -v *paquet-version*.changes. Cette commande provoquera une vérification des paquets source et binaire. En cas de difficultés pour comprendre les messages de retour, utiliser l'option -i de **lintian**. Cette option rendra **lintian** beaucoup plus explicite dans la description des problèmes.

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 Section A.2.1 ;

— facultativement exécuter **debdiff** (voir Section A.2.2) pour analyser les modifications depuis une ancienne version si celle-ci existe ;

— revenir à la version précédente du paquet (si elle existe) — cela permet de tester les scripts postrm et prerm ;

— retirer le paquet et le réinstaller à nouveau ;

— 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.

The first time a version is uploaded that corresponds to a particular upstream version, the original source tar file must be uploaded and included in the `.changes` file. Subsequently, this very same tar file should be used to build the new diffs and `.dsc` files, and will not need to be re-uploaded.

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.

Please notice that, in non-native packages, permissions on files that are not present in the `*.orig.tar.{gz,bz2,xz}` will not be preserved, as diff does not store file permissions in the patch. However, when using source format "3.0 (quilt)", permissions of files inside the `debian` directory are preserved since they are stored in a tar archive.

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`.

Packages are normally uploaded into `unstable`. Uploads to `unstable` or `experimental` should use these suite names in the changelog entry; uploads for other supported suites should use the suite codenames, as they avoid any ambiguity.

En fait, il y a d'autres distributions possibles : *nomdecode*`-security`, consultez Section 5.8.5 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-r` 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.

To ensure that your upload will be accepted, you should discuss the changes with the stable release team before you upload. For that, file a bug against the `release.debian.org` pseudo-package using **reportbug**, including the patch you want to apply to the package version currently in `stable`. The patch should be a source **debdiff** (see Section A.2.2) against the current version in `stable`. The changelog entry should be against the `stable` distribution (e.g. stretch') and should be detailed, including `Closes` statements for bugs that are fixed by the upload.

Une mise à jour de paquet pour la distribution `stable` requiert des soins supplémentaires. Un paquet de cette distribution ne devrait être mis à jour que dans les cas suivants :

— un problème fonctionnel vraiment critique ;

— un paquet devenu non installable ;

— un paquet indisponible pour une architecture.

In the past, uploads to `stable` were used to address security problems as well. However, this practice is deprecated, as uploads used for Debian security advisories (DSAs) are automatically copied to the appropriate `proposed-updates` archive when the advisory is released. See Section 5.8.5 for detailed information on handling security problems. If the security team deems the problem to be too benign to be fixed through a `DSA`, the stable release managers are usually willing to include your fix nonetheless in a regular upload to `stable`.

Il est fortement déconseillé de changer quoi que ce soit de non important car même une modification triviale peut provoquer un bogue.

Les paquets à destination de `stable` doivent être compilés sur un système qui tourne sous `stable`, afin de limiter les dépendances aux bibliothèques (et autres paquets) disponibles dans `stable` ; par exemple, un paquet pour `stable` qui dépend de bibliothèques uniquement disponibles dans `unstable` sera rejeté. Modifier les dépendances d'autres paquets (en semant la pagaille avec les champs `Provides` ou les fichiers `shlibs`), au risque de rendre d'autres paquets impossibles à installer, est fortement déconseillé.

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.

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

Veuillez consulter les informations de la section relative à testing pour plus de détails.

5.6 Envois de paquets

5.6.1 Envois sur `ftp-master`

To upload a package, you should upload the files (including the signed changes and dsc file) with anonymous ftp to `ftp.upload.debian.org` in the directory /pub/UploadQueue/. To get the files processed there, they need to be signed with a key in the Debian Developers keyring or the Debian Maintainers keyring (see `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.

Pour supprimer des paquets, veuillez lire le fichier `ftp://ftp.upload.debian.org/pub/UploadQueue/README` et le paquet Debian dcut.

5.6.2 Envois différés

Il peut être utile d'envoyer un paquet à un moment donné, mais vouloir que ce paquet n'entre dans l'archive que quelques jours plus plus tard. Par exemple, lors de la préparation d'une mise à jour indépendante (« Non-Maintainer Upload » ou NMU), vous pourriez vouloir donner quelques jours au responsable pour réagir.

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`.

Avec **dput**, le paramètre `--delayed` *`DELAY`* permet de placer le paquet dans une des files d'attente.

5.6.3 Envois de sécurité

Do **NOT** upload a package to the security upload queue (on `*.security.upload.debian.org`) without prior authorization from the security team. If the package does not exactly meet the team's requirements, it will cause many problems and delays in dealing with the unwanted upload. For details, please see Section 5.8.5.

5.6.4 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.

5.6.5 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.

Also note that new uploads are announced on the IRC channel `#debian-devel-changes`. If your upload fails silently, it could be that your package is improperly signed, in which case you can find more explanations on `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 pseudo-paquet `ftp.debian.org` demandant la modification de la section ou de la priorité de votre paquet. Utilisez un sujet comme `override: PACKAGE1:section/pri [...], 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 Section 4.6.1. 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 Manipulation des bogues

Chaque développeur doit être capable de travailler avec le système de suivi des bogues (« bug tracking system » ou BTS) Debian. Il faut savoir comment remplir des rapports de bogue correctement (voir Section 7.1), 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.

5.8.1 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 :

```
# Synthèse hebdomadaire des rapports de bogue qui me concernent
0 17 * * fri   echo "index maint address" | mail request@bugs.debian.org
```

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

5.8.2 Réponses aux bogues

Lorsque vous répondez à des rapports de bogue, assurez-vous que toutes vos discussions concernant les bogues sont envoyées au rapporteur du bogue et au bogue lui-même (*123*@bugs.debian.org par exemple). Si vous rédigez un nouveau courrier et si vous ne vous souvenez plus de l'adresse du rapporteur de bogue, vous pouvez utiliser l'adresse *123*-submitter@bugs.debian.org pour contacter le rapporteur *et* enregistrer votre courrier dans le journal du bogue (ce qui signifie que vous n'avez pas besoin d'envoyer une copie du courrier à *123*@bugs.debian.org).

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 Section 5.8.4.

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.

5.8.3 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 Section 7.1 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 aux personnes que le bogue existe, mais ne sera pas corrigé. Si cette situation n'est pas acceptable, vous (ou le rapporteur) pouvez vouloir demander une décision par le comité technique en réassignant le bogue à `tech-ctte` (vous pouvez utiliser la commande `clone` du BTS si vous désirez garder le bogue comme rapporté sur votre paquet). 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 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. si le bogue est lié à l'empaquetage, vous devez simplement le corriger. Si vous ne pouvez pas le corriger vous-même, marquez alors le bogue avec `help` (aide). Vous pouvez également demander de l'aide sur debian-devel@lists.debian.org ou debian-qa@lists.debian.org. S'il s'agit d'un problème amont, vous devez faire suivre le rapport à l'auteur amont. Faire suivre un bogue n'est pas suffisant, vous devez vérifier à chaque version si le bogue a été corrigé ou non. S'il a été corrigé, il vous suffit de le clôturer, sinon vous devez le rappeler à l'auteur. Si vous avez les compétences nécessaires, vous pouvez préparer un correctif pour le bogue et l'envoyer en même temps à l'auteur. Assurez-vous d'envoyer le correctif au BTS et marquez le bogue avec `patch` (correctif) ;
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. Once a corrected package is available in the archive, the bug should be closed indicating the version in which it was fixed. This can be done automatically; read Section 5.8.4.

5.8.4 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
```

We prefer the `closes: #`*XXX* syntax, as it is the most concise entry and the easiest to integrate with the text of the `changelog`. Unless specified differently by the `-v`-switch to **dpkg-buildpackage**, only the bugs closed in the most recent changelog entry are closed (basically, exactly the bugs mentioned in the changelog-part in the `.changes` file are closed).

Historiquement, les envois identifiés comme mise à jour indépendante (« non-maintainer upload » ou 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`.

Si vous entrez un numéro de bogue incorrect ou si vous oubliez un bogue dans les entrées du fichier `changelog`, n'hésitez pas à annuler tout dommage que l'erreur a entraîné. Pour rouvrir un bogue fermé par erreur, envoyez une commande `reopen` *XXX* à l'adresse de contrôle du système de suivi des bogues, control@bugs.debian.org. Pour fermer tous les bogues restants qui ont été corrigés par votre envoi, envoyez le fichier `.changes` à *XXX*-done@bugs.debian.org où *XXX* est le numéro du bogue et placez « Version: *YYY* » et une ligne vide comme deux premières lignes du corps du courrier où *YYY* est la première version dans laquelle le bogue a été corrigé.

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 Section 6.3.

5.8.5 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) ;

— Any fixed packages that you have prepared yourself (send the resulting debdiff or alternatively only the `.diff.gz` and `.dsc` files and read Section 5.8.5.4 first)

— toute assistance possible pour aider à tester (exploitation de faille, tests de régression, etc.) ;

— toute information utile pour l'annonce de sécurité (voir Section 5.8.5.3).

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.

5.8.5.1 Gestionnaire de sécurité (« `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 possibles à propos des problèmes de sécurité connus : quels sont les paquets et versions affectés et non affectés, et par conséquent si `stable`, `testing` et `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.

5.8.5.2 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.

5.8.5.3 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 (usurpation 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) ;

— des références à des annonces amont, des identifiants CVE et toute autre information utile pour recouper les références de la vulnérabilité.

5.8.5.4 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.

When an update is made to the stable release, care must be taken to avoid changing system behavior or introducing new bugs. In order to do this, make as few changes as possible to fix the bug. Users and administrators rely on the exact behavior of a release once it is made, so any change that is made might break someone's system.

This is especially true of libraries: make sure you never change the API (Application Program Interface) or ABI (Application Binary Interface), no matter how small the change.

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 Section A.2.2).

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

— **Target the right distribution** in your `debian/changelog`: *`codename`*`-security` (e.g. `stretch-security`). Do not target *`distribution`*`-proposed-updates` or `stable`!

— l'envoi devra être fait avec **urgency=high** ;

— 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 ;

— Make sure the **version number** is proper. It must be greater than the current package, but less than package versions in later distributions. If in doubt, test it with `dpkg --compare-versions`. Be careful not to re-use a version number that you have already used for a previous upload, or one that conflicts with a binNMU. The convention is to append `+deb`*`X`*`u1` (where *`X`* is the major release number), e.g. `1:2.4.3-4+deb9u1`, of course increasing 1 for any subsequent uploads.

— 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 Section 4.4) ou mettre en place un chroot (voir Section A.4.3 et Section A.4.2).

5.8.5.5 Mise à jour du paquet corrigé

Do **NOT** upload a package to the security upload queue (on `*.security.upload.debian.org`) without prior authorization from the security team. If the package does not exactly meet the team's requirements, it will cause many problems and delays in dealing with the unwanted upload.

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.

Once you have created and tested the new package and it has been approved by the security team, it needs to be uploaded so that it can be installed in the archives. For security uploads, the place to upload to is `ftp://ftp.security.upload.deb`

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.

Uploads that are waiting for acceptance or verification are only accessible by the security team. This is necessary since there might be fixes for security problems that cannot be disclosed yet.

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`.

5.9 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.

5.9.1 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 5.7.

5.9.2 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.

If you want to remove a package you maintain, you should note this in the bug title by prepending `ROM` (Request Of Maintainer). There are several other standard acronyms used in the reasoning for a package removal; see `https://ftp-master.debian.org/removals.html` for a complete list. That page also provides a convenient overview of pending removal requests.

Note that removals can only be done for the `unstable`, `experimental` and `stable` distributions. Packages are not removed from `testing` directly. Rather, they will be removed automatically after the package has been removed from `unstable` and no package in `testing` depends on it. (Removals from `testing` are possible though by filing a removal bug report against the `release.debian.org` pseudo-package. See Section 5.14.2.2.)

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 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 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.

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

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.

Plus d'informations sur ce sujet et autres sujets connexes sont disponibles sur https://wiki.debian.org/ftpmaster_Removals et https://qa.debian.org/howto-remove.html.

Si vous ne savez pas bien si un paquet peut être supprimé, demandez l'avis des autres développeurs sur la liste debian-devel@lists.debian.org. Le programme **apt-cache** du paquet `apt` pourra aussi vous être utile. La commande `apt-cache showpkg` *`paquet`* vous indiquera, entre autres, les paquets qui dépendent de *`paquet`*. Parmi les programmes utiles, citons **apt-cache rdepends**, **apt-rdepends**, **build-rdeps** (du paquet `devscripts`) et **grep-dctrl**. La suppression de paquets orphelins est discutée sur 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`.

5.9.2.1 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. 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 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.

5.9.3 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 Section 5.9.2). 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.

5.9.4 Abandon de paquet

Si vous ne pouvez plus maintenir un paquet, il faut en informer les autres et faire le nécessaire pour le marquer `orphaned` (orphelin). Vous devriez remplacer votre nom par `Debian QA Group <packages@qa.debian.org>` dans le champ `Maintainer` du paquet et faire un rapport de bogue sur le pseudopaquet `wnpp`. Le titre de votre rapport de bogue devrait être « `O:` *`paquet -- description courte`* » pour indiquer que le paquet est orphelin (`O` signifie « `Orphaned` » : orphelin). La gravité du bogue sera `normal` ; si le paquet a une priorité standard ou supérieure, elle devrait être `important`. Si vous le jugez nécessaire, envoyez une copie à debian-devel@lists.debian.org en mettant cette adresse dans le champ `X-Debbugs-CC` de l'en-tête du message. N'utilisez pas le champ `CC` sinon le sujet du message ne contiendra pas le numéro du bogue.

Si vous avez simplement l'intention de donner le paquet, mais que vous pouvez conserver sa maintenance pour le moment, vous devriez plutôt soumettre un rapport de bogue sur `wnpp` intitulé `RFA:` *`package -- description courte`*. `RFA` signifie « `Request For Adoption` » (demande d'adoption).

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

5.9.5 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.

It is not OK to simply take over a package without assent of the current maintainer — that would be package hijacking. You can, of course, contact the current maintainer and ask them for permission to take over the package.

However, when a package has been neglected by the maintainer, you might be able to take over package maintainership by following the package salvaging process as described in Section 5.12. If you have reason to believe a maintainer is no longer active at all, see Section 7.4.

Complaints about maintainers should be brought up on the developers' mailing list. If the discussion doesn't end with a positive conclusion, and the issue is of a technical nature, consider bringing it to the attention of the technical committee (see the technical committee web page for more information).

If you take over an old package, you probably want to be listed as the package's official maintainer in the bug system. This will happen automatically once you upload a new version with an updated `Maintainer` field, although it can take a few hours after the upload is done. If you do not expect to upload a new version for a while, you can use Section 4.10 to get the bug reports. However, make sure that the old maintainer has no problem with the fact that they will continue to receive the bugs during that time.

5.9.6 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 comaintenir le paquet ou parrainer le paquet si nécessaire.

Toutes les étapes nécessaires à l'introduction d'un nouveau paquet (Section 5.1) devraient être suivies.

You should base your work on the latest packaging available that is suitable. That might be the latest version from `unstable`, which will still be present in the snapshot archive.

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.

5.10 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.

Porting is the act of building Debian packages for architectures that are different from the original architecture of the package maintainer's binary package. It is a unique and essential activity. In fact, porters do most of the actual compiling of Debian packages. For instance, when a maintainer uploads a (portable) source package with binaries for the `i386` architecture, it will be built for each of the other architectures, amounting to 10 more builds.

5.10.1 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 sec-

tion contient une liste d'erreurs régulièrement commises par les responsables Debian — problèmes courants qui bloquent souvent les porteurs et compliquent inutilement leur travail.

The first and most important thing is to respond quickly to bugs or issues raised by porters. Please treat porters with courtesy, as if they were in fact co-maintainers of your package (which, in a way, they are). Please be tolerant of succinct or even unclear bug reports; do your best to hunt down whatever the problem is.

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. Make sure that your `Build-Depends` and `Build-Depends-Indep` settings in `debian/control` are set properly. The best way to validate this is to use the `debootstrap` package to create an `unstable` chroot environment (see Section A.4.2). Within that chrooted environment, install the `build-essential` package and any package dependencies mentioned in `Build-Depends` and/or `Build-Depends-Indep`. Finally, try building your package within that chrooted environment. These steps can be automated by the use of the **pbuilder** program, which is provided by the package of the same name (see Section A.4.3).

   En cas de difficultés pour configurer un environnement `chrooté`, **dpkg-depcheck** pourra peut-être vous aider (voir Section A.6.6).

   Consultez la Charte Debian pour en savoir plus sur les dépendances de fabrication ;
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. vérifiez que le paquet source est correct. Faites `dpkg-source -x` *`paquet`*`.dsc` pour vous assurer que le paquet se décompresse correctement. En utilisant le résultat de ce test, construisez votre paquet binaire à l'aide de la commande **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. Make sure you don't rely on locally installed or hacked configurations or programs. For instance, you should never be calling programs in `/usr/local/bin` or the like. Try not to rely on programs being set up in a special way. Try building your package on another machine, even if it's the same 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 demande 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 fabrication 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**.

5.10.2 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 Section 5.11.1.

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`.

La manière d'invoquer **dpkg-buildpackage** est la suivante : `dpkg-buildpackage -B -m`*`adresse-porteur`*. Bien sûr, remplacez *`adresse-porteur`* par votre adresse électronique. Cette commande construira les parties du paquet qui dépendent de l'architecture, en utilisant la cible `binary-arch` de `debian/rules`.

Si vous travaillez sur une machine Debian pour vos efforts de portage et que vous devez signer l'envoi localement pour être accepté dans l'archive, vous pouvez exécuter **debsign** sur le fichier `.changes` pour qu'il soit signé convenablement, ou utiliser le mode de signature à distance de **dpkg-sig**.

5.10.2.1 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]

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

5.10.2.2 Quand utiliser une NMU source pour un portage

Les porteurs faisant des NMU source suivent normalement les instructions de Section 5.11, 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 actuels 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.

5.10.3 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.

5.10.3.1 Listes de diffusion et pages web

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

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 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.

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.

5.10.3.2 Outils pour les porteurs

Les descriptions de plusieurs outils de portage peuvent être trouvées en Section A.7.

5.10.3.3 wanna-build

The wanna-build system is used as a distributed, client-server build distribution system. It is usually used in conjunction with build daemons running the buildd program. Build daemons are "slave" hosts, which contact the central wanna-build system to receive a list of packages that need to be built.

wanna-build is not yet available as a package; however, all Debian porting efforts are using it for automated package building. The tool used to do the actual package builds, sbuild, is available as a package; see its description in Section A.4.4. Please note that the packaged version is not the same as the one used on build daemons, but it is close enough to reproduce problems.

Most of the data produced by wanna-build that is generally useful to porters is available on the web at https://buildd.debian.org/. This data includes nightly updated statistics, queueing information and logs for build attempts.

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.

5.10.4 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://anonscm.debian.org/cgit/mirror/packages-arch-specific.git/tree/Packages-arch-specific ; veuillez consulter le début du fichier pour savoir qui contacter pour le modifier.

Please note that it is insufficient to only add your package to Packages-arch-specific without making it fail to build on unsupported architectures: A porter or any other person trying to build your package might accidentally upload it without noticing it doesn't work. If in the past some binary packages were uploaded on unsupported architectures, request their removal by filing a bug against ftp.debian.org.

5.10.5 Paquets non libres pouvant être automatiquement construits

By default packages from the `non-free` section 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 nonfree@release.debian.org et expliquer pourquoi le paquet peut légalement et automatiquement être construit.

5.11 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).*

5.11.1 NMU : quand et comment

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

— Avez-vous adapté le NMU de façon à aider le responsable ? Puisqu'il pourrait exister un désaccord sur le fait que le responsable ait vraiment besoin d'aide ou pas, la file d'attente DELAYED est là pour donner au responsable le temps de réagir et permet, comme effet de bord, de permettre des révisions indépendantes du diff de la NMU.

— Est-ce que votre NMU corrige réellement le bogue ? (« bogue » signifie n'importe quelle sorte de bogue, p. ex. un bogue `wishlist` (souhait) pour empaqueter une nouvelle version amont, mais l'impact sur le responsable devra être minimisé avec soin). Corriger des problèmes superficiels ou changer la façon d'empaqueter (p. ex. utiliser cdbs au lieu de dh) dans les NMU est déconseillé.

— 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.

— Have you clearly expressed your intention to NMU, at least in the BTS? If that didn't generate any feedback, it might also be a good idea to try to contact the maintainer by other means (email to the maintainer addresses or private email, 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 ;

— autres NMU : dix jours.

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'accepter 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` »), 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.

Après une NMU, vous êtes responsable d'éventuels problèmes introduits. Vous devez garder un œil sur le paquet (s'abonner au paquet dans le PTS est un bon moyen).

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.

5.11.2 NMU et `debian/changelog`

Just like any other (source) upload, NMUs must add an entry to `debian/changelog`, telling what has changed with this upload. The first line of this entry must explicitly mention that this upload is an NMU, e.g.:

```
  * 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.

If you upload a package to testing or stable, you sometimes need to "fork" the version number tree. This is the case for security uploads, for example. For this, a version of the form `+deb`*X*`u`*Y* should be used, where *X* is the major release number, and *Y* is a counter starting at `1`. For example, while stretch (Debian 9) is stable, a security NMU to stable for a package at version `1.5-3` would have version `1.5-3+deb9u1`, whereas a security upload to buster would get version `1.5-3+deb10u1`.

5.11.3 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 Section 5.6.2) 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.

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.

5.11.4 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 co-responsable 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.

5.11.5 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 Section 5.10.2.1. 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.

5.11.6 NMU et envoi de QA

NMUs are uploads of packages by somebody other than their assigned maintainer. There is another type of upload where the uploaded package is not yours: QA uploads. QA uploads are uploads of orphaned packages.

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 PTS 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.

Instead of doing a QA upload, you can also consider adopting the package by making yourself the maintainer. You don't need permission from anybody to adopt an orphaned package; you can just set yourself as maintainer and upload the new version (see Section 5.9.5).

5.11.7 NMU et envoi d'équipe

Sometimes you are fixing and/or updating a package because you are member of a packaging team (which uses a mailing list as Maintainer or Uploader; see Section 5.13) but you don't want to add yourself to Uploaders because you do not plan to contribute regularly to this specific package. If it conforms with your team's policy, you can perform a normal upload without being listed directly as Maintainer or Uploader. In that case, you should start your changelog entry with the following line:

```
* Team upload.
```

5.12 Package Salvaging

Package salvaging is the process by which one attempts to save a package that, while not officially orphaned, appears poorly maintained or completely unmaintained. This is a weaker and faster procedure than orphaning a package officially through the powers of the MIA team. Salvaging a package is not meant to replace MIA handling, and differs in that it does not imply anything about the overall activity of a maintainer. Instead, it handles a package maintainership transition for a single package only, leaving any other package or Debian membership or upload rights (when applicable) untouched.

Note that the process is only intended for actively taking over maintainership. Do not start a package salvaging process when you do not intend to maintain the package for a prolonged time. If you only want to fix certain things, but not take over the package, you must use the NMU process, even if the package would be eligible for salvaging. The NMU process is explained in Section 5.11.

Another important thing to remember: It is not acceptable to hijack others' packages. If followed, this salvaging process will help you to ensure that your endeavour is not a hijack but a (legal) salvaging procedure, and you can counter any allegations of hijacking with a reference to this process. Thanks to this process, new contributors should no longer be afraid to take over packages that have been neglected or entirely forgotten.

The process is split into two phases: In the first phase you determine whether the package in question is *eligible* for the salvaging process. Only when the eligibility has been determined you may enter the second phase, the *actual* package salvaging.

For additional information, rationales and FAQs on package salvaging, please visit the Salvaging Packages page on the Debian wiki.

5.12.1 When a package is eligible for package salvaging

A package becomes eligible for salvaging when it has been neglected by the current maintainer. To determine that a package has really been neglected by the maintainer, the following indicators give a rough idea what to look for:

— NMUs, especially if there has been more than one NMU in a row.

— Bugs filed against the package do not have answers from the maintainer.

— Upstream has released several versions, but despite there being a bug entry asking for it, it has not been packaged.

— There are QA issues with the package.

You will have to use your judgement as to whether a given combination 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.

5.12.2 How to salvage a package

If and *only* if a package has been determined to be eligible for package salvaging, any prospective maintainer may start the following package salvaging procedure.

1. Open a bug with the severity "important" against the package in question, expressing the intent to take over maintainership of the package. For this, the title of the bug should start with `ITS: package-name`[3]. You may alternatively offer to only take co-maintenance of the package. When you file the bug, you must inform all maintainers, uploaders and if applicable the packaging team explicitly by adding them to `X-Debbugs-CC`. Additionally, if the maintainer(s) seem(s) to be generally inactive, please inform the MIA team by adding `mia@qa.debian.org` to `X-Debbugs-CC` as well. As well as the explicit expression of the intent to salvage, please also take the time to document your assessment of the eligibility in the bug report, for example by listing the criteria you've applied and adding some data to make it easier for others to assess the situation.
2. In this step you need to wait in case any objections to the salvaging are raised; the maintainer, any current uploader or any member of the associated packaging team of the package in question may object publicly in response to the bug you've filed within `21 days`, and this terminates the salvaging process.

   The current maintainers may also agree to your intent to salvage by filing a (signed) public response to the the bug. They might propose that you become a co-maintainer instead of the sole maintainer. On team

3. ITS is shorthand for *"Intend to Salvage"*

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. After the 21 days delay, if no answer has been sent to the bug from the maintainer, one of the uploaders or team, you may upload the new release of the package into the `DELAYED` queue with a minimum delay of `seven days`. You should close the salvage bug in the changelog and you must also send an nmudiff to the bug ensuring that copies are sent to the maintainer and any uploaders (including teams) of the package by `CC'ing` them in the mail to the BTS.

   During the waiting time of the `DELAYED` queue, the maintainer can accept the salvaging, do an upload themselves or (ask to) cancel the upload. The latter two of these will also stop the salvaging process, but the maintainer must reply to the salvaging bug with more information about their action.

5.13 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 co-responsables.

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

Dans sa forme la plus simple, ajouter un nouveau co-responsable est assez facile :

— Set up the co-maintainer with access to the sources you build the package from. Generally this implies you are using a network-capable version control system, such as `CVS` or `Subversion`. Alioth (see Section 4.12) provides such tools, amongst others.

— ajouter les nom et adresse correctes du co-responsable au champ `Uploaders` dans le premier paragraphe du fichier `debian/control` ;

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

— Using the PTS (Section 4.10), the co-maintainers should subscribe themselves to the appropriate source package.

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 parmi 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. Put the mailing list address in the `Maintainer` field. In the `Uploaders` field, put the team members who care for the package. In this case, you must make sure the mailing list accepts bug reports without any human interaction (like moderation for non-subscribers).

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 Section 4.11) 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 Section 5.11.7). 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`.

5.14 La distribution testing

5.14.1 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.

5.14.2 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 :

— le paquet doit avoir été disponible dans unstable depuis deux, cinq ou dix jours selon l'urgence de l'envoi (élevée, moyenne ou basse). Veuillez noter que cette urgence est « collante » (« sticky »), ce qui signifie qu'il est tenu compte de l'urgence la plus élevée des envois depuis la précédente transition dans testing.

— 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. 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 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 inter-relations est trop compliqué et ne peut être résolu par le script. Voir ci-dessous pour des détails.

Some further dependency analysis is shown on https://release.debian.org/migration/ — but be warned: this page also shows build dependencies that are not considered by britney.

5.14.2.1 Désynchronisation

For the testing migration script, outdated means: There are different versions in unstable for the release architectures (except for the architectures in fuckedarches; fuckedarches is a list of architectures that don't keep up (in update_out.py), but currently, it's empty). Outdated has nothing whatsoever to do with the architectures this package has in testing.

Considérons cet exemple :

	alpha	**arm**
testing	1	-
unstable	1	2

The package is out of date on alpha in unstable, and will not go to testing. Removing the package would not help at all; the package is still out of date on alpha, and will not propagate to 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).

5.14.2.2 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`.

Of course, there is another reason to remove a package from `testing`: it's just too buggy (and having a single RC-bug is enough to be in this state).

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.

5.14.2.3 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; depends: b=1	2; depends: b=2
b	1; depends: a=1	2; depends: a=2

Aucun des paquets `a` et `b` ne sera considéré pour 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.

5.14.2.4 Influence d'un paquet dans `testing`

Généralement, l'état d'un paquet dans `testing` ne change rien pour la transition de la prochaine version d'`unstable` vers `testing`, avec deux exceptions : si le nombre de bogues critiques pour la publication du paquet diminue, le paquet peut migrer même s'il a encore des bogues critiques pour la publication. La seconde exception est que si la version du paquet dans `testing` est désynchronisée entre les différentes architectures, alors toute architecture peut être mise à jour vers la version du paquet source ; cependant, cela ne peut se produire que si le paquet a été précédemment forcé, si l'architecture est dans `fuckedarches` ou s'il n'y avait pas du tout de paquet binaire de cette architecture présent dans `unstable` lors de la migration dans `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.

5.14.2.5 Détails

Suivent 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://ftp-master.debian.org/testing/update_output/.

Les coups de pouce (« hints ») sont disponibles sur https://ftp-master.debian.org/testing/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 testing-proposed-updates ou outrepasser l'urgence.

5.14.3 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.

Keep in mind that packages uploaded there are not automatically processed; they have to go through the hands of the release manager. So you'd better have a good reason to upload there. In order to know what a good reason is in the release managers' eyes, you should read the instructions that they regularly give on 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 facilité, mais il est recommandé de demander l'autorisation des responsables de publication auparavant. 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.

Version numbers are usually selected by appending +debXuY, where X is the major release number of Debian and Y is a counter starting at 1. e.g. 1:2.4.3-4+deb9u1.

Veuillez vous assurer 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 n'avoir intégré qu'un minimum de changements ;

— vérifiez avoir ajouté une explication appropriée dans le journal de modification (changelog) ;

— Make sure that you've written the testing code name (e.g. buster) into your target distribution;

— vérifiez 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 testing-proposed-updates, et moins élevé que celle de unstable ;

— 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.

5.14.4 Foire aux questions

5.14.4.1 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.

The unstable bug count comprises all release-critical bugs that are marked to apply to *package*/*version* combinations available in unstable for a release architecture. The testing bug count is defined analogously.

5.14.4.2 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.

However, the old version may have provided a binary package with an old soname of a library, such as libacme-foo0. Removing the old acmefoo will remove libacme-foo0, which will break any packages that depend on it.

Evidently, this mainly affects packages that provide changing sets of binary packages in different versions (in turn, mainly libraries). However, it will also affect packages upon which versioned dependencies have been declared of the ==, <=, or << varieties.

When the set of binary packages provided by a source package changes in this way, all the packages that depended on the old binaries will have to be updated to depend on the new binaries instead. Because installing such a source package into testing breaks all the packages that depended on it in testing, some care has to be taken now: all the depending packages must be updated and ready to be installed themselves so that they won't be broken, and, once everything is ready, manual intervention by the release manager or an assistant is normally required.

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.

Chapitre 6

Meilleures pratiques d'empaquetage

Debian's quality is largely due to the Debian Policy, which defines explicit baseline requirements that all Debian packages must fulfill. Yet there is also a shared history of experience which goes beyond the Debian Policy, an accumulation of years of experience in packaging. Many very talented people have created great tools, tools which help you, the Debian maintainer, create and maintain excellent packages.

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`

The following recommendations apply to the `debian/rules` file. Since `debian/rules` controls the build process and selects the files that go into the package (directly or indirectly), it's usually the file maintainers spend the most time on.

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.

Annexe A 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`.

Pour démarrer avec `debhelper`, il est conseillé de lire debhelper(1) et de consulter les exemples fournis avec le paquet. **dh_make**, fourni avec le paquet `dh-make` (voir Section A.3.2) peut être utilisé pour convertir un paquet source originel en paquet géré par `debhelper`. Cette méthode rapide ne doit cependant pas se substituer à une compréhension individuelle des commandes **dh_***. Si vous utilisez un assistant, vous devez prendre le temps de le connaître, pour comprendre ses besoins et son comportement.

6.1.2 Séparation des correctifs (« `patches` ») en plusieurs fichiers

Les paquets complexes ont souvent de nombreux bogues qui doivent être gérés par le responsable. Si certains de ces bogues sont corrigés par des modifications effectuées directement dans le code source, sans discernement, il

peut devenir difficile de retrouver l'origine et la motivation de ces correctifs. Cela peut également rendre bien plus complexe l'intégration d'une nouvelle version amont qui pourrait inclure certains de ces correctifs (mais pas tous). Il est en effet alors quasiment impossible de reprendre le jeu initial de changements (par exemple dans le fichier `.diff.gz`) et supprimer ceux qui correspondent à des correctifs appliqués par le responsable amont.

Fortunately, with the source format "3.0 (quilt)" it is now possible to keep patches separate without having to modify `debian/rules` to set up a patch system. Patches are stored in `debian/patches/` and when the source package is unpacked patches listed in `debian/patches/series` are automatically applied. As the name implies, patches can be managed with **quilt**.

Avec l'ancien format source « 1.0 », il est aussi possible de séparer les correctifs mais un système de correctif dédié doit être utilisé : les correctifs individualisés sont embarqués dans le fichier général de correctifs Debian (`.diff.gz`), en général à l'intérieur du répertoire `debian/`. La seule différence est que ces correctifs ne sont pas appliqués directement par **dpkg-source** mais par la règle `build` de `debian/rules`, à travers une dépendance à la règle `patch`. À l'inverse, les correctifs sont retirés par la règle `clean`, à travers une dépendance à la règle `unpatch`.

quilt is the recommended tool for this. It does all of the above, and also allows one to manage patch series. See the `quilt` package for more information.

D'autres outils existent pour gérer les correctifs, comme **dpatch**, et le système de correctif intégré à `cdbs`.

6.1.3 Paquets binaires multiples

Un simple paquet source créera souvent plusieurs paquets binaires, soit pour fournir plusieurs variantes du même logiciel (par exemple le paquet source `vim`) ou pour répartir les fichiers en plusieurs paquets plus petits au lieu d'un seul paquet monolithique (ce qui peut permettre à un utilisateur de n'installer que les éléments nécessaires et donc préserver l'espace disque).

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`.

The first case is a bit more difficult since it involves multiple recompiles of the same software but with different configuration options. The `vim` source package is an example of how to manage this using a hand-crafted `debian/rules` file.

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. Section 6.2.1 donne des indications communes à ces deux parties, Section 6.2.2 donne des indications spécifiques pour le résumé et Section 6.2.3 donne des indications pour la description.

6.2.1 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.

6.2.2 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 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`* »).

6.2.3 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.

The following paragraphs should answer the following questions: Why do I as a user need this package? What other features does the package have? What outstanding features and deficiencies are there compared to other packages (e.g., if you need X, use Y instead)? Is this package related to other packages in some way that is not handled by the package manager (e.g., is this the client for the foo server)?

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.

— How is this package different from the competition? Is it a better implementation? more features? different features? Why should I choose this package?

6.2.4 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.

6.2.5 Emplacement du système de gestion de versions

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

6.2.5.1 Vcs-Browser

La valeur de ce champ doit être une URL `http://` 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.

6.2.5.2 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). It is allowed to specify different VCS fields for the same package: they will all be shown in the PTS web interface.

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 Subversion du paquet `vim`. Veuillez noter que l'URL a la forme `svn://` (au lieu de `svn+ssh://`) et pointe sur la branche `trunk/`. Une utilisation des champs `Vcs-Browser` et `Homepage`, décrits précédemment, est aussi indiquée.

```
Source: vim
Section: editors
Priority: optional
<snip>
Vcs-Svn: svn://svn.debian.org/svn/pkg-vim/trunk/packages/vim
Vcs-Browser: https://svn.debian.org/wsvn/pkg-vim/trunk/packages/vim
Homepage: http://www.vim.org
```

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 Section 5.8.4 pour plus d'informations.

6.3.2 Selecting the upload urgency

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 Section 5.14.2). For example, you might select **urgency=low** if the changes since the last upload are large and might be disruptive in unanticipated ways.

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 Section 5.8.4.

Les entrées du journal des modifications ne devraient **pas** être utilisées pour des discussions variées avec les émetteurs des rapports de bogues (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 Section 5.8.2 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 FU w/ ancient CRTs.
```

Trop d'acronymes qui rendent difficile de savoir ce qu'était le « merdoyage » (NdT : FU signifie « fsckup », donc cet exemple ajoute la vulgarité à l'incompréhensibilité) 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 de simplement 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.
```

If for some reason you didn't mention the bug number in a previous changelog entry, there's no problem, just close the bug normally in the BTS. There's no need to touch the changelog file, presuming the description of the fix is already in (this applies to the fixes by the upstream authors/maintainers as well; you don't have to track bugs that they fixed ages ago in your changelog).

```
  * 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 Meilleures pratiques pour les scripts du responsable

Maintainer scripts include the files `debian/postinst`, `debian/preinst`, `debian/prerm` and `debian/postrm`. These scripts take care of any package installation or deinstallation setup that isn't handled merely by the creation or removal of files and directories. The following instructions supplement the Debian Policy.

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 which 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`, **type**, et **which** ne sont pas conformes à POSIX.

Bien que **which** soit acceptable car fourni dans le paquet requis `debianutils`, il n'est pas disponible sur la partition racine mais est situé dans le répertoire `/usr/bin` au lieu de `/bin`, ce qui rend son utilisation impossible si `/usr` n'est pas encore monté. La plupart des scripts ne seront toutefois pas affectés par cela.

6.5 Gestion de la configuration avec `debconf`

`Debconf` is a configuration management system that can be used by all the various packaging scripts (`postinst` mainly) to request feedback from the user concerning how to configure the package. Direct user interactions must now be avoided in favor of `debconf` interaction. This will enable non-interactive installations in the future.

`debconf` est un outil très pratique mais souvent mal utilisé. De nombreuses erreurs classiques sont mentionnées dans la page de manuel debconf-devel(7). Il est indispensable de lire cette page de manuel avant de décider d'utiliser debconf. Quelques bonnes pratiques sont également indiquées dans le présent document.

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).

6.5.1 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.

Keep usage notes to what they belong: the `NEWS.Debian`, or `README.Debian` file. Only use notes for important notes that may directly affect the package usability. Remember that notes will always block the install until confirmed or bother the user by email.

Carefully choose the questions' priorities in maintainer scripts. See debconf-devel(7) for details about priorities. Most questions should use medium and low priorities.

6.5.2 Recommandations générales pour les auteurs et les traducteurs

6.5.2.1 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.

6.5.2.2 Courtoisie avec les traducteurs

Debconf templates may be translated. Debconf, along with its sister package **po-debconf**, offers a simple framework for getting templates translated by translation teams or even individuals.

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).

Avoid changing templates too often. Changing template text induces more work for translators, which will get their translation fuzzied. A fuzzy translation is a string for which the original changed since it was translated, therefore requiring some update by a translator to be usable. When changes are small enough, the original translation is kept in PO files but marked as `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"
```

This command will first synchronize the PO and POT files in `debian/po` with the template files listed in `debian/po/POTFILES.in`. Then, it will send a call for new translations, in the debian-i18n@lists.debian.org mailing list. Finally, it will also send a call for translation updates to the language team (mentioned in the `Language-Team` field of each PO file) as well as the last translator (mentioned in `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.

6.5.2.3 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_orig; cp *.po po_orig
   ```

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_orig/*.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_orig
```

6.5.2.4 Proscrire toute supposition sur les interfaces utilisateurs

Templates text should not make reference to widgets belonging to some debconf interfaces. Sentences like *If you answer Yes...* have no meaning for users of graphical interfaces that use checkboxes for boolean questions.

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.

6.5.2.5 Proscrire l'utilisation de la première personne

You should avoid the use of first person (*I will do this...* or *We recommend...*). The computer is not a person and the Debconf templates do not speak for the Debian developers. You should use neutral construction. Those of you who already wrote scientific publications, just write your templates like you would write a scientific paper. However, try using the active voice if still possible, like *Enable this if ...* instead of *This can be enabled if....*

6.5.2.6 Neutralité en genre

As a way of showing our commitment to our diversity statement, please use gender-neutral constructions in your writing. This means avoiding pronouns like he/she when referring to a role (like "maintainer") whose gender is unknown. Instead, you should use the plural form (singular they).

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

Les informations présentées dans cette partie proviennent pour l'essentiel de la page de manuel debconf-devel(7).

6.5.3.1 `Type`

6.5.3.1.1 `string`

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

6.5.3.1.2 `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.

6.5.3.1.3 `boolean`

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

6.5.3.1.4 `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: truc/bidule
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.

Le système d'indicateur (« `flag` ») d'écrans `debconf` permet de faire de telles choses. La page de manuel po-debconf(7) documente toutes ces possibilités.

6.5.3.1.5 `multiselect`

similaire au type `select`, mais permet de choisir plusieurs (ou aucune) valeurs parmi la liste de choix.

6.5.3.1.6 `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.

6.5.3.1.7 `text`

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

6.5.3.1.8 `error`

This type is designed to handle error messages. It is mostly similar to the note type. Front ends may present it differently (for instance, the dialog front end of cdebconf draws a red screen instead of the usual blue one).

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.

6.5.3.2 `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.

The short description should be able to stand on its own. Some interfaces do not show the long description by default, or only if the user explicitly asks for it or even do not show it at all. Avoid things like: "What do you want to do?"

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.

Ne soyez pas trop verbeux. Les utilisateurs ont tendance à ne pas lire les écrans trop longs. Une vingtaine de lignes est une limite que vous ne devriez pas dépasser car, avec l'interface `dialog` standard, les utilisateurs devront monter et descendre avec des ascenseurs, ce que la plupart des utilisateurs ne font simplement pas.

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.).

6.5.3.3 `Choices`

This field should be used for select and multiselect types. It contains the possible choices that will be presented to users. These choices should be separated by commas.

6.5.3.4 `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.

6.5.4 Template fields specific style guide

6.5.4.1 Champ `Type`

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

6.5.4.2 Champ `Description`

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

6.5.4.2.1 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.

6.5.4.2.2 Modèles `boolean`

— The short description should be phrased in the form of a question, which should be kept short and should generally end with a question mark. Terse writing style is permitted and even encouraged if the question is rather long (remember that translations are often longer than original versions).

— 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... »).

6.5.4.2.3 Modèles `select` et `multiselect`

— The short description is a prompt and **not** a title. Do **not** use useless "Please choose..." constructions. Users are clever enough to figure out they have to choose something... :)

— 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).

6.5.4.2.4 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 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.

6.5.4.3 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.

6.5.4.4 Champ `Default`

If the default value for a select template is likely to vary depending on the user language (for instance, if the choice is a language choice), please use the _Default trick.

This special field allows translators to put the most appropriate choice according to their own language. It will become the default choice when their language is used while your own mentioned Default Choice will be used when using English.

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:
```

Note the use of brackets, which allow internal comments in debconf fields. Also note the use of comments, which will show up in files the translators will work with.

Les commentaires sont très utiles car l'astuce « `_Default` » est parfois déroutante pour les traducteurs qui doivent y mettre leur propre choix et non une simple traduction.

6.5.4.5 Champ `Default`

Do NOT use an empty default field. If you don't want to use default values, do not use Default at all.

If you use po-debconf (and you **should**; see Section 6.5.2.2), consider making this field translatable, if you think it may be translated.

Si la valeur par défaut peut dépendre de la langue ou du pays (par exemple une langue par défaut dans un programme), pensez à utiliser le type « `_Default` » documenté dans la page de manuel po-debconf(7).

6.6 Internationalisation

This section contains global information for developers to make translators' lives easier. More information for translators and developers interested in internationalization are available in the Internationalisation and localisation in Debian documentation.

6.6.1 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.

The goal of `debconf` was to make package configuration easier for maintainers and for users. Originally, translation of debconf templates was handled with **debconf-mergetemplate**. However, that technique is now deprecated; the best way to accomplish `debconf` internationalization is by using the `po-debconf` package. This method is easier both for maintainer and translators; transition scripts are provided.

Avec `po-debconf`, les traductions sont gérées dans des fichiers `.po` (hérités des techniques de traduction utilisées avec **gettext**). Des fichiers modèles contiennent les messages d'origine et les champs à traduire y sont marqués spécifiquement. Lorsque le contenu d'un champ traduisible est modifié, l'emploi de la commande **debconf-updatepo** permet d'indiquer que la traduction a besoin d'une mise à jour par les traducteurs. Ensuite, au moment de la construction du paquet, le programme **dh_installdebconf** s'occupe des opérations nécessaires pour ajouter

le modèle avec les traductions à jour dans les paquets binaires. Vous pouvez consulter la page de manuel de po-debconf(7) pour plus d'informations.

6.6.2 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.

If you maintain XML or SGML documentation, we suggest that you isolate any language-independent information and define those as entities in a separate file that is included by all the different translations. This makes it much easier, for instance, to keep URLs up to date across multiple files.

Some tools (e.g. `po4a`, `poxml`, or the `translate-toolkit`) are specialized in extracting the translatable material from different formats. They produce PO files, a format quite common to translators, which permits seeing what needs to be re-translated when the translated document is updated.

6.7 Situations courantes de gestion de paquets

6.7.1 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.

6.7.2 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.

Les bonnes pratiques pour la maintenance de paquets fournissant des bibliothèques ont été rassemblées dans le guide de gestion des paquets de bibliothèque.

6.7.3 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.

The manpages do not need to be written directly in the troff format. Popular source formats are DocBook, POD and reST, which can be converted using **xsltproc**, **pod2man** and **rst2man** respectively. To a lesser extent, the **help2man** program can also be used to write a stub.

6.7.4 Catégories particulières de paquets

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

— Perl related packages have a Perl policy; some examples of packages following that policy are `libdbd-pg-perl` (binary perl module) or `libmldbm-perl` (arch independent perl module).

— Python related packages have their Python policy; see `/usr/share/doc/python/python-policy.txt.gz` in the `python` package.

— Les paquets liés à Emacs utilisent une charte Emacs.

— Les paquets liés à Java utilisent une charte Java.

— OCaml related packages have their own policy, found in `/usr/share/doc/ocaml/ocaml_packaging_policy.gz` from the `ocaml` package. A good example is the `camlzip` source package.

— Les paquets fournissant des DTD XML ou SGML devraient suivre les recommandations données dans le paquet `sgml-base-doc`.

— Les paquets Lisp doivent s'enregistrer avec `common-lisp-controller`, pour lequel plus d'information est disponible dans `/usr/share/doc/common-lisp-controller/README.packaging`.

6.7.5 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.

However, if the size of the data is considerable, consider splitting it out into a separate, architecture-independent package (`_all.deb`). By doing this, you avoid needless duplication of the same data into ten or more .debs, one per each architecture. While this adds some extra overhead into the `Packages` files, it saves a lot of disk space on Debian mirrors. Separating out architecture-independent data also reduces processing time of **lintian** (see Section A.2) when run over the entire Debian archive.

6.7.6 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
```

6.7.7 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, le paramètre `--guess-dummy` de la commande **deborphan** permet de rechercher les paquets de transition qui étaient nécessaires lors de mises à niveau mais peuvent être supprimés sans problème. Pour cela, il recherche la chaîne « `dummy` » ou « `transitional` » dans leur description courte.

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

Also, it is recommended to adjust its section to `oldlibs` and its priority to `optional` in order to ease **deborphan**'s job.

6.7.8 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` »).

6.7.8.1 Source originelle (« `pristine` »)

La caractéristique définissant une archive source originelle et 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.

There are no universally accepted guidelines that upstream authors follow regarding the directory structure inside their tarball, but **dpkg-source** is nevertheless able to deal with most upstream tarballs as pristine source. Its strategy is equivalent to the following:

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

   ```
   zcat path/to/nomdupaquet_version-amont.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)`.

6.7.8.2 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. We cannot prevent upstream authors from changing the tarball they distribute without also incrementing the version number, so there can be no guarantee that a pristine tarball is identical to what upstream *currently* distributing at any point in time. All that can be expected is that it is identical to something that upstream once *did* distribute. If a difference arises later (say, if upstream notices that they weren't using maximal compression in their original distribution and then re-**gzip** it), that's just too bad. Since there is no good way to upload a new `.orig.tar.{gz,bz2,xz}` for the same version, there is not even any point in treating this situation as a bug.

1. **devrait** être documenté dans le fichier source. Des informations détaillées sur la façon dont les sources ont été obtenues et comment il est possible de refaire l'opération devraient être fournies dans le fichier `debian/copyright`. Il est également suggéré de fournir une cible `get-orig-source` dans le fichier `debian/rules`, qui permette de refaire cette opération, comme indiqué dans la Charte Debian à propos du script de construction principal : `debian/rules` ;
2. **ne devrait pas** contenir de fichier non distribué par les auteurs amont, ou dont vous avez modifié le contenu ; [2]
3. **devrait**, sauf si c'est impossible pour des raisons légales, préserver l'intégralité de l'infrastructure de construction et de portabilité fournie par l'auteur amont. Par exemple, il ne faut pas enlever un fichier sous prétexte qu'il ne sert qu'à la compilation sur MS-DOS. De même, un `Makefile` fourni en amont n'a pas de raison d'être enlevé si la première action de `debian/rules` est de l'écraser en exécutant un script de configuration.

   (*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. **devrait** utiliser *`nomdupaquet-version-amont`*`(.orig)` comme nom de répertoire racine de l'archive. Cela permet de distinguer les sources originelles des sources reconstruites ;
5. **devrait** utiliser le taux de compression maximal.

6.7.8.3 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).

6.7.9 Meilleures pratiques pour les paquets de débogage

Un paquet de débogage est un paquet dont le nom se termine par « -dbg », et qui contient des informations supplémentaires que **gdb** peut utiliser. Puisque les informations de débogage, comme les noms de fonction et de numéro de ligne, sont par défaut absentes des paquets binaires Debian, elles ne pourraient autrement pas être disponibles lors de l'utilisation de **gdb**. Les paquets de débogage permettent aux utilisateurs qui le désirent d'ajouter ces informations de débogage supplémentaires, sans augmenter la taille d'un système normal avec ces informations.

It is up to a package's maintainer whether to create a debug package or not. Maintainers are encouraged to create debug packages for library packages, since this can aid in debugging many programs linked to a library. In general, debug packages do not need to be added for all programs; doing so would bloat the archive. But if a maintainer finds that users often need a debugging version of a program, it can be worthwhile to make a debug package for it. Programs that are core infrastructure, such as Apache and the X server are also good candidates for debug packages.

Certains paquets de débogage peuvent contenir une compilation spécifique de débogage complète d'une bibliothèque ou d'un autre programme, mais la plupart peuvent préserver de la place et du temps de compilation en contenant plutôt séparément les symboles de débogage 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 gardés 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`.

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.

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`, il suffit d'appeler **dh_strip --dbg-package=libtruc-dbg**, et d'ajouter une entrée à `debian/control` pour le paquet de débogage.

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

2. As a special exception, if the omission of non-free files would lead to the source failing to build without assistance from the Debian diff, it might be appropriate to instead edit the files, omitting only the non-free parts of them, and/or explain the situation in a `README.source` file in the root of the source tree. But in that case please also urge the upstream author to make the non-free components easier to separate from the rest of the source.

```
Depends: libtruc (= ${binary:Version})
```

6.7.10 Meilleures pratiques pour les métapaquets

Un métapaquet est un paquet principalement 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`).

The long description of the meta-package must clearly document its purpose so that the user knows what they will lose if they remove the package. Being explicit about the consequences is recommended. This is particularly important for meta-packages that are installed during initial installation and that have not been explicitly installed by the user. Those tend to be important to ensure smooth system upgrades and the user should be discouraged from uninstalling them to avoid potential breakages.

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 de la simple création et maintenance 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 bogues 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-adre`*

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 — 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.

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: titre-du-bogue

Package: nom-de-paquet
[ ... ]
User: adresse-mail
Usertags: nom-d-etiquette [ nom-d-etiquette ... ]

description-du-bogue ...
```

Note that tags are separated by spaces and cannot contain underscores. If you are filing bugs for a particular group or team it is recommended that you set the `User` to an appropriate mailing list after describing your intention there.

Pour voir les bogues marqués par une étiquette d'utilisateur en particulier, rendez-vous sur la page `https://bugs.debian.org/`

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, 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 Section A.2.1). Si cela vous paraît impossible, vous devriez alors envisager d'abandonner certains de vos paquets (voir Section 5.9.4). 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 Section 5.13).

7.2.2 Chasses aux bogues

From time to time the QA group organizes bug squashing parties to get rid of as many problems as possible. They are announced on debian-devel-announce@lists.debian.org and the announcement explains which area will be the focus of the party: usually they focus on release critical bugs but it may happen that they decide to help finish a major upgrade (like a new **perl** version that requires recompilation of all the binary modules).

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 bug 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.

People participating in the party have special rules for NMU; they can NMU without prior notice if they upload their NMU to DELAYED/3-day at least. All other NMU rules apply as usual; they should send the patch of the NMU to the BTS (to one of the open bugs fixed by the NMU, or to a new bug, tagged fixed). They should also respect any particular wishes of the maintainer.

Si vous ne vous sentez pas à l'aise avec une NMU, envoyez simplement un correctif au BTS. C'est de loin meilleur qu'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 d'un 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.

You may also be interested in contacting the persons who are subscribed to a given source package via Section 4.10. You can do so by using the *package*`@packages.qa.debian.org` email address.

7.4 Gestion des responsables non joignables

If you notice that a package is lacking maintenance, you should make sure that the maintainer is active and will continue to work on their packages. It is possible that they are not active anymore, but haven't registered out of the system, so to speak. On the other hand, it is also possible that they just need a reminder.

Il y a un système simple (la base de données `MIA`) dans laquelle les informations sur les responsables supposés manquant à l'appel (« `Missing In Action` ») sont enregistrées. Quand un membre du groupe `QA` contacte un responsable inactif ou trouve plus d'informations sur celui-ci, un enregistrement dans la base de données MIA a lieu. Ce système est disponible dans `/org/qa.debian.org/mia` sur l'hôte `qa.debian.org` et peut être interrogé avec **mia-query**. Utilisez **mia-query --help** pour voir comment interroger la base de données. Si aucune information n'a encore été enregistrée pour un responsable inactif ou si vous pouvez ajouter plus d'informations, vous devriez utiliser la procédure suivante.

La première étape est de contacter poliment le responsable et d'attendre une réponse pendant un temps raisonnable. Il est assez difficile de définir le « 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.

A non-functional e-mail address is a violation of Debian Policy . If an e-mail "bounces", please file a bug against the package and submit this information to the MIA database.

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 `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 — 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 des conditions 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 !

On the other hand, although we are volunteers, a package maintainer has made a commitment and therefore has a responsibility to maintain the package. So you can stress the importance of the greater good — if a maintainer does not have the time or interest anymore, they should let go and give the package to someone with more time and/or interest.

If you are interested in working on the MIA team, please have a look at the `README` file in `/org/qa.debian.org/mia` on `qa.debian.org`, where the technical details and the MIA procedures are documented, and contact mia@qa.debian.org.

7.5 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.

7.5.1 Parrainage de paquets

Sponsoring a package means uploading a package for a maintainer who is not able to do it on their own. It's not a trivial matter; the sponsor must verify the packaging and ensure that it is of the high level of quality that Debian strives to have.

Les développeurs Debian peuvent parrainer des paquets, mais pas les mainteneurs Debian.

Le processus de parrainage d'un paquet est :

1. le responsable prépare un paquet source (`.dsc`) et le met en ligne quelque part (sur mentors.debian.net par exemple) ou, mieux encore, fournit un lien vers un dépôt de gestion de versions (consultez Section 4.4.5) où le paquet est maintenu ;
2. The sponsor downloads (or checks out) the source package.
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.

Before delving into the details of how to sponsor a package, you should ask yourself whether adding the proposed package is beneficial to Debian.

There's no simple rule to answer this question; it can depend on many factors: is the upstream codebase mature and not full of security holes? Are there pre-existing packages that can do the same task and how do they compare to this new package? Has the new package been requested by users and how large is the user base? How active are the upstream developers?

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 apprendre ?

Il est aussi recommandé de connaître sa position par rapport à Debian : approuve-t-il la philosophie Debian et désire-t-il rejoindre Debian ? Vu comme il est facile de devenir mainteneur Debian, vous pourriez ne parrainer que des personnes ayant l'intention de rejoindre le projet. De cette façon, vous savez au départ que vous n'aurez pas a parrainer indéfiniment.

7.5.1.1 Parrainage d'un nouveau paquet

New maintainers usually have certain difficulties creating Debian packages — this is quite understandable. They will make mistakes. That's why sponsoring a brand new package into Debian requires a thorough review of the Debian packaging. Sometimes several iterations will be needed until the package is good enough to be uploaded to Debian. Thus being a sponsor implies being a 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]

1. You can find more checks in the wiki, where several developers share their own sponsorship checklists.

— 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).

— Run **lintian** (see Section A.2.1). It will catch many common problems. Be sure to verify that any **lintian** overrides set up by the maintainer are fully justified.

— Exécutez **licensecheck** (qui fait partie de Section A.6.1) et vérifiez que `debian/copyright` ait l'air correct et exhaustif. Recherchez des problèmes de licence (comme des fichiers avec « All rights reserved » — tous droits réservés — en en-tête, ou ayant une licence non compatible avec les principes du logiciel libre selon Debian). **grep -ri** peut vous aider ici.

— Construisez le paquet avec **pbuilder** (ou n'importe quel outil du même genre, consultez Section A.4.3) pour vérifier que les dépendances de constructions sont exhaustives.

— Relisez `debian/control` : est-il conforme aux meilleures pratiques (consultez Section 6.2) ? Les dépendances sont elles exhaustives ?

— Relisez `debian/rules` : est-il conforme aux meilleures pratiques (consultez Section 6.1) ? 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 Chapitre 6) ?

— Build the packages, install them and try the software. Ensure that you can remove and purge the packages. Maybe test them with **piuparts**.

If the audit did not reveal any problems, you can build the package and upload it to Debian. Remember that even if you're not the maintainer, as a sponsor you are still responsible for what you upload to Debian. That's why you're encouraged to keep up with the package through Section 4.10.

Note that you should not need to modify the source package to put your name in the `changelog` or in the `control` file. The `Maintainer` field of the `control` file and the `changelog` should list the person who did the packaging, i.e. the sponsee. That way they will get all the BTS mail.

Instead, you should instruct **dpkg-buildpackage** to use your key for the signature. You do that with the `-k` option:

```
dpkg-buildpackage -kidentifiant_de_clef
```

Si vous utilisez **debuild** et **debsign**, vous pouvez même le configurer de façon permanente dans `~/.devscripts` :

```
DEBSIGN_KEYID=identifiant_de_clef
```

7.5.1.2 Parrainage de la mise à jour d'un paquet existant

You will usually assume that the package has already gone through a full review. So instead of doing it again, you will carefully analyze the difference between the current version and the new version prepared by the maintainer. If you have not done the initial review yourself, you might still want to have a deeper look just in case the initial reviewer was sloppy.

To be able to analyze the difference, you need both versions. Download the current version of the source package (with **apt-get source**) and rebuild it (or download the current binary packages with **aptitude download**). Download the source package to sponsor (usually with **dget**).

Read the new changelog entry; it should tell you what to expect during the review. The main tool you will use is **debdiff** (provided by the `devscripts` package); you can run it with two source packages (`.dsc` files), or two binary packages, or two `.changes` files (it will then compare all the binary packages listed in the `.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 modification Debian.

If everything is fine, build the package and compare the binary packages to verify that the changes on the source package have no unexpected consequences (some files dropped by mistake, missing dependencies, etc.).

You might want to check out the Package Tracking System (see Section 4.10) to verify if the maintainer has not missed something important. Maybe there are translation updates sitting in the BTS that could have been integrated. Maybe the package has been NMUed and the maintainer forgot to integrate the changes from the NMU into their package. Maybe there's a release critical bug that they have left unhandled and that's blocking migration to `testing`. If you find something that they could have done (better), it's time to tell them so that they can improve for next time, and so that they have a better understanding of their responsibilities.

Si vous n'avez pas trouvé de problème majeur, envoyez la nouvelle version. Sinon, demandez au responsable de fournir une version corrigée.

7.5.2 Recommandation d'un nouveau développeur

Les recommandations pour un futur développeur sont disponibles sur le site web de Debian.

7.5.3 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 plupart de ce chapitre.

According to Introduction to i18n from Tomohiro KUBOTA, I18N (internationalization) means modification of software or related technologies so that software can potentially handle multiple languages, customs, and so on in the world, 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 actuellement 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. Most of the time, the translation is handled upstream within projects like the Free Translation Project, the GNOME Translation Project or the KDE one. 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, but no real infrastructure to ease the translation process.

Un effort pour traduire les descriptions de paquet a démarré il y a longtemps, même si les outils fournissent très peu de prise en charge pour les utiliser vraiment (seul APT peut les utiliser une fois configuré convenablement). Les responsables n'ont rien à faire de particulier pour gérer les traductions des descriptions de paquets ; les traducteurs devraient utiliser le projet de traduction de descriptions de Debian (`DDTP`).

Pour les questionnaires `debconf`, les responsables devraient utiliser le paquet `po-debconf` pour faciliter le travail des traducteurs, qui peuvent utiliser le DDTP pour faire leur travail (mais les équipes française et brésilienne ne le font pas). Certaines statistiques sont disponibles à la fois sur le site du DDTP (à propos de ce qui est vraiment traduit) et sur le centre de traduction de Debian (à propos de ce qui est intégré dans les paquets).

Pour les pages web, chaque équipe l10n a accès au système de gestion de versions correspondant et les statistiques sont disponibles sur le site des statistiques de traduction Debian centralisées.

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.

Pour la documentation spécifique aux paquets (pages de manuel, documents info, autres formats), presque tout est encore à faire.

En particulier, le projet KDE gère la traduction de ses documentations de la même façon que ses messages de programme.

Il existe un effort pour gérer les pages de manuel spécifiques à Debian au sein d'un système de gestion de versions spécifique.

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 or `debconf` templates, you have nothing to do; the DDTP infrastructure will dispatch the material to translate to volunteers with no need for interaction on your part.

Pour tous les autres matériels (fichiers gettext, pages de manuel ou autre documentation), la meilleure solution est de placer votre texte quelque part sur l'Internet et de demander sur debian-i18n la traduction dans différentes langues. Certains membres des équipes de traduction sont abonnés à cette liste et ils prendront soin de la traduction et du processus de relecture. Quand ils auront fini, ils vous enverront le document traduit.

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 l'étiquette `patch` 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 dûs 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**.

Annexe A

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 ».

Some people prefer to use high-level package maintenance tools and some do not. Debian is officially agnostic on this issue; any tool that gets the job done is fine. Therefore, this section is not meant to stipulate to anyone which tools they should use or how they should go about their duties of maintainership. Nor is it meant to endorse any particular tool to the exclusion of a competing tool.

La plupart des descriptions de ces outils proviennent des descriptions de leurs paquets. Vous trouverez plus d'informations dans les documentations de ces paquets. Vous pouvez aussi obtenir plus d'informations avec la commande `apt-cache show` *`nom_de_paquet`*.

A.1 Outils de base

Les outils suivants sont pratiquement nécessaires à tout responsable.

A.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.

A.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`.

Many feel that this system should be used for all packages that require interactive configuration; see Section 6.5. `debconf` is not currently required by Debian Policy, but that may change in the future.

A.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.

A.2 Contrôle de paquets (« `lint` »)

According to the Free On-line Dictionary of Computing (FOLDOC), lint' is: "A Unix C language processor which carries out more thorough checks on the code than is usual with C compilers." Package lint tools help package maintainers by automatically finding common problems and policy violations in their packages.

A.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 Section 5.3 pour plus d'informations sur comment et quand utiliser Lintian.

Vous pouvez aussi obtenir un résumé de tous les problèmes signalés par Lintian sur vos paquets en https://lintian.debian.org/. Ces rapports contiennent la sortie de la dernière version de **lintian** pour l'ensemble de la distribution de développement (unstable).

A.2.2 debdiff

debdiff (du paquet devscripts, Section A.6.1) 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 paquet_1-1_arch.deb paquet_2-1_arch.deb
```

Ou même sur un couple de fichiers de changements :

```
debdiff paquet_1-1_arch.changes paquet_2-1_arch.changes
```

Pour plus d'informations, veuillez consulter debdiff(1).

A.3 Assistance pour debian/rules

Des outils de construction de paquets facilitent le processus d'écriture du fichier debian/rules. Section 6.1.1 contient plus d'informations sur l'intérêt de les utiliser ou non.

A.3.1 debhelper

debhelper is a collection of programs that can be used in debian/rules to automate common tasks related to building binary Debian packages. debhelper includes programs to install various files into your package, compress files, fix file permissions, and integrate your package with the Debian menu system.

Unlike some approaches, debhelper is broken into several small, simple commands, which act in a consistent manner. As such, it allows more fine-grained control than some of the other debian/rules tools.

Il existe aussi un certain nombre de petites extensions debhelper trop éphémères pour être documentées ici. La plupart seront listés avec apt-cache search ^dh-.

A.3.2 dh-make

The dh-make package contains **dh_make**, a program that creates a skeleton of files necessary to build a Debian package out of a source tree. As the name suggests, **dh_make** is a rewrite of debmake, and its template files use **dh_*** programs from 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.

A.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.

A.4 Construction de paquets

The following packages help with the package building process, general driving of **dpkg-buildpackage**, as well as handling supporting tasks.

A.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 référentiel Git et assiste le responsable lors de l'intégration de modifications amont dans le référentiel.

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 version.

A.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 minimum. Les automates de constructions « `chrootés` » utilisent ce paquet ; voir ci-après.

A.4.3 `pbuilder`

`pbuilder` constructs a chrooted system, and builds a package inside the chroot. It is very useful to check that a package's build dependencies are correct, and to be sure that unnecessary and wrong build dependencies will not exist in the resulting package.

A related package is `cowbuilder`, which speeds up the build process using a COW filesystem on any standard Linux filesystem.

A.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 Section 5.10.3.3 pour plus d'informations et `https://buildd.debian.org/` pour voir le système en fonctionnement.

A.5 Envoi de paquets

Les paquets suivants aident à automatiser ou simplifier le processus d'envoi de paquets dans l'archive officielle.

A.5.1 `dupload`

`dupload` contient un script du même nom pour envoyer des paquets dans l'archive Debian, suivre les envois, et les annoncer par courrier électronique. Il peut être configuré pour envoyer les paquets ailleurs ou avec d'autres méthodes.

A.5.2 `dput`

The `dput` package and script do much the same thing as `dupload`, but in a different way. It has some features over `dupload`, such as the ability to check the GnuPG signature and checksums before uploading, and the possibility of running **dinstall** in dry-run mode after the upload.

A.5.3 dcut

dcut (du paquet `dput`, Section A.5.2) permet de supprimer des fichiers du répertoire d'envoi FTP.

A.6 Automatisation de la maintenance

Les outils suivants permettent d'automatiser les différentes tâches de maintenance en ajoutant des entrées au journal de modification ou des lignes de signatures, en cherchant des bogues depuis Emacs et en utilisant le fichier officiel `config.sub` le plus récent.

A.6.1 devscripts

`devscripts` is a package containing wrappers and tools that are very helpful for maintaining your Debian packages. Example scripts include **debchange** and **dch**, which manipulate 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. **debrsign** can be used to remotely sign a package prior to upload, which is nice when the machine you build the package on is different from where your GPG keys are.

Voir la page de manuel devscripts(1) pour une liste complète des scripts disponibles.

A.6.2 autotools-dev

`autotools-dev` contains best practices for people who maintain packages that use **autoconf** and/or **automake**. Also contains canonical `config.sub` and `config.guess` files, which are known to work on all Debian ports.

A.6.3 dpkg-repack

dpkg-repack creates a Debian package file out of a package that has already been installed. If any changes have been made to the package while it was unpacked (e.g., files in `/etc` were modified), the new package will inherit the changes.

This utility can make it easy to copy packages from one computer to another, or to recreate packages that are installed on your system but no longer available elsewhere, or to save the current state of a package before you upgrade it.

A.6.4 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.

A.6.5 dpkg-dev-el

`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.

A.6.6 dpkg-depcheck

dpkg-depcheck (du paquet `devscripts`, Section A.6.1) 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).

A.7 Outils de portage

Les outils suivants sont pratiques pour les porteurs et la compilation croisée (« `cross-compilation` »).

A.7.1 dpkg-cross

dpkg-cross est un outil qui installe les bibliothèques et les en-têtes nécessaires à une compilation croisée (« cross-compilation ») d'une manière similaire à dpkg. De plus, les fonctionnalités de **dpkg-buildpackage** et **dpkg-shlibdeps** ont été améliorées pour accepter les compilations croisées.

A.8 Documentation et information

Les paquets suivants fournissent des informations pour les responsables ou de l'aide pour construire de la documentation.

A.8.1 docbook-xml

docbook-xml fournit la définition de type de document (« Document Type Definition » ou DTD) XML pour DocBook, souvent utilisé pour la documentation Debian (de même que la plus ancienne DTD SGML pour DebianDoc). Ce manuel, par exemple, est écrit en XML pour DocBook.

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-*.

Pour fabriquer des PDF à partir des FO (« Formatting Objects »), il faut un processeur de FO comme xmlroff ou fop. dblatex est un autre outil pour générer des PDF à partir des XML pour DocBook.

A.8.2 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 devrait être utilisé à la place). Il fournit également des scripts pour construire et décliner les sources en de multiples formats de sortie.

De la documentation sur la DTD est disponible dans le paquet debiandoc-sgml-doc.

A.8.3 debian-keyring

Contains the public GPG keys of Debian Developers and Maintainers. See Section 3.2.2 and the package documentation for more information.

A.8.4 debian-el

debian-el provides an Emacs mode for viewing Debian binary packages. This lets you examine a package without unpacking it.