Référence du développeur Debian

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

19 août 2012

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 2012-08-19

Si vous désirez imprimer cette référence, vous devriez utiliser la version PDF. Cette page est également disponible en allemand, anglais et japonais.

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

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

Si vous ne l'avez pas encore fait, commencez par vous inscrire à la liste debian-devel@lists.debian.org. Pour cela, envoyez un courrier à debian-devel-REQUEST@lists.debian.org avec le mot `subscribe` en objet (champ `Subject`) de message. En cas de problème, contactez l'administrateur de la liste listmaster@lists.debian.org. Vous trouverez plus d'informations en Section 4.1. debian-devel-announce@lists.debian.org est une autre liste incontournable pour suivre les développements de Debian.

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 GNU/Linux, 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 `http://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 proposer 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 évitées. Veuillez consulter `http://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 `http://wiki.debian.org/DebianMentorsFaq`.

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

2.3 Enregistrement comme responsable Debian

Avant de décider de devenir responsable Debian, il vous faudra lire toute la documentation disponible dans le coin du nouveau responsable. Elle décrit en détail toutes les étapes préparatoires qu'il vous faudra franchir avant de déposer votre candidature. Par exemple, avant d'être candidat, il vous faudra lire le contrat social Debian. Devenir responsable Debian implique que vous adhériez à ce contrat social et que vous vous engagiez à le soutenir ; il est très important que les responsables soient en accord avec les principes fondamentaux qui animent le projet Debian GNU/Linux. Lire le Manifeste GNU est aussi une bonne idée.

Le processus d'enregistrement a pour but de vérifier votre identité, vos intentions et vos compétences. Le nombre de personnes travaillant pour Debian GNU/Linux 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.)

Si vous n'avez pas de clé OpenPGP, créez-la. Tout responsable a besoin d'une clé OpenPGP pour signer et vérifier les mises à jour de paquets. Vous lirez la documentation du logiciel de cryptographie que vous utiliserez car elle contient des informations indispensables pour la sécurité de votre clé. Les défaillances de sécurité sont bien plus souvent dues à des erreurs humaines qu'à des problèmes logiciels ou à des techniques d'espionnage avancées. Voir Section 3.2.2 pour plus d'informations sur la gestion de votre clé publique.

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.

Vous avez besoin d'une clé en version 4 à utiliser pour le développement Debian. La longueur de votre clé doit être plus grande que 1024 bits ; il n'y a pas de raison d'utiliser une clé plus petite et faire cela serait bien moins sûr. [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

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 besoin 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.

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

Pour en savoir plus, consultez le coin des nouveaux responsables sur le site Debian. Assurez-vous de bien connaître les étapes nécessaires au processus d'enregistrement avant de vous porter candidat. Vous gagnerez beaucoup de temps si vous êtes bien préparé.

Chapitre 3

Devoirs du développeur Debian

3.1 Devoirs du responsable de paquet

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

3.1.1 Œuvrer pour la prochaine publication `stable`

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

Plus concrètement, vous devriez surveiller si vos paquets migrent vers `testing` (consultez Section 5.13). 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 amonts sont ou deviennent hostiles envers Debian ou la communauté 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

Soyez très vigilant en utilisant votre clé privée. Ne la placez pas sur un serveur public ou sur des machines multi-utilisateurs telles que les serveurs Debian (voir Section 4.4). Sauvegardez vos clés et gardez-en une copie hors connexion. Lisez la documentation fournie avec votre logiciel et la FAQ PGP.

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

Si vous ajoutez des signatures ou des identifiants à votre clé publique, vous pouvez mettre à jour le porte-clés Debian en envoyant votre clé publique à `keyring.debian.org`.

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.

Une présentation approfondie de la gestion de clé Debian peut être trouvée dans la documentation du paquet `debian-keyring`.

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. envoyez un courrier électronique signé par GnuPG à debian-private@lists.debian.org indiquant pourquoi vous quittez le projet ;
3. signalez aux responsables du porte-clés Debian que vous quittez le projet en ouvrant un ticket en écrivant à keyring@rt.debian.org avec les mots « Debian RT » dans le sujet (peu importe la casse).

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 :

– contacter da-manager@debian.org ;
– passer le 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.

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

Chapitre 4

Ressources pour les responsables Debian

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

4.1 Listes de diffusion

Une grande partie des discussions entre les développeurs Debian (et les utilisateurs) a lieu dans un 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 `http://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 lue.

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 `http://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`.

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

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 uniforme (« `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 uniformes pour plus d'informations.

4.3 Documentation

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

4.4 Serveurs Debian

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

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

Ces machines peuvent être utilisées à votre discrétion pour des buts liés à 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 : `http://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 au `DSA` (administrateurs de système Debian « `Debian System Administrators` ») du gestionnaire de demandes (« `request tracker` ») : `https://rt.debian.org/` (identifiant « debian » ; mot de passe 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.

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`

Le serveur `ftp-master.debian.org` est le serveur maître de l'archive Debian. En général, les paquets envoyés à ftp.upload.debian.org aboutissent sur ce serveur, voir Section 5.6.

Ce serveur est à accès restreint ; un miroir est disponible sur `ries.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.

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

4.4.4 Serveur web pour pages personnelles `people`

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

Si vous avez des informations spécifiques à Debian que vous voulez rendre disponibles sur le web, vous pouvez le faire en les plaçant dans le répertoire `public_html` de votre répertoire personnel sur `people.debian.org`. Elles seront accessibles à l'adresse `http://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 ou demander un nouveau projet avec le système de gestion de versions de votre choix. Alioth gère CVS (cvs.alioth.debian.org/cvs.debian.org), Subversion (svn.debian.org), Arch (tla/baz, tout deux sur arch.debian.org), Bazaar (bzr.debian.org), Darcs (darcs.debian.org), Mercurial (hg.debian.org) et Git (git.debian.org). Consultez http://wiki.debian.org/Alioth/PackagingProject si vous avez l'intention de maintenir un paquet à l'aide d'un logiciel de gestion de versions. 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
Exécution de l'interpréteur de commandes initial 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 http://db.debian.org/machines.cgi.

4.5 Base de données des développeurs

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

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

- l'adresse de 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 le shell 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 http://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 http://db.debian.org/doc-mail.html.

4.6 Archive Debian

La distribution Debian GNU/Linux 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. Ceux-ci sont gérés dans la base de données de l'archive et les logiciels qui l'accompagnent. 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'un répertoire de 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 GNU/Linux 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 GNU/Linux.

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).

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

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 était disponible uniquement pour `i386`. Debian 2.0 gérait les architectures `i386` and `m68k`. Debian 2.1 gérait les architectures `i386`, `m68k`, `alpha`, et `sparc`. Depuis, Debian a considérablement évolué. Debian 6 gère un total de neuf architectures Linux (`amd64`, `armel`, `i386`, `ia64`, `mips`, `mipsel`, `powerpc`, `s390` et `sparc`) et deux architectures kFreeBSD (`kfreebsd-i386` et `kfreebsd-amd64`).

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

4.6.3 Paquets

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

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

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

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

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

4.6.4 Distributions

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

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

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

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). Ceci 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 5.13.

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 publiées 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 ceci ne cause aucun souci aux responsables de l'archive (« `ftpmasters` »), les paquets expérimentaux sont périodiquement enlevés quand vous envoyez le paquet dans `unstable` avec un numéro de version supérieur.

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

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

4.6.5 Noms de code des distributions

Chaque distribution Debian diffusée a un `nom de code` : Debian 1.1 s'appelle `Buzz` ; Debian 1.2, `Rex` ; Debian 1.3, `Bo` ; Debian 2.0, `Hamm` ; Debian 2.1, `Slink` ; Debian 2.2, `Potato` ; Debian 3.0, `Woody` ; Debian 3.1, `Sarge` ; Debian 4.0, `Etch` ; Debian 5.0, `Lenny` ; Debian 6.0, `Squeeze` et la prochaine publication sera appelée `Wheezy`. Il existe aussi une « pseudodistribution » nommée `Sid`, il s'agit de la distribution `unstable` ; comme les paquets sont déplacés d'`unstable` vers `testing` quand ils sont suffisamment stables, la distribution `Sid` n'est jamais publiée. En plus du contenu habituel d'une distribution Debian, `Sid` contient des paquets pour des architectures qui ne sont pas encore officiellement prises en charge ou pour lesquelles la distribution n'a pas encore été publiée. Ces architectures seront intégrées ultérieurement à la distribution principale.

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

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

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

4.7 Miroirs Debian

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

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

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

4.8 Système « Incoming »

Le système « Incoming » est responsable de la collecte des paquets mis à jour et 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 déclanche 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 envoyée à debian-devel-changes@lists.debian.org.

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

4.9 Informations sur un paquet

4.9.1 Sur le web

Chaque paquet a plusieurs pages web dédiées. http://packages.debian.org/*nom-de-paquet* affiche chaque version du paquet disponible dans les différentes distributions. Les informations détaillées par version 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 http://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 ries.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 recompilé sur toutes les architectures.

4.10 Système de suivi des paquets (`PTS`)

Le système de suivi des paquets (« `Package Tracking System` » ou `PTS`) est un outil de suivi par courrier de l'activité d'un paquet source. Cela signifie que l'on peut vraiment recevoir les mêmes courriers que le responsable, simplement en s'inscrivant au paquet dans le PTS.

Chaque courrier envoyé par le PTS est classé sous l'un des mots-clés listés ci-dessous. Ceci permet de sélectionner les courriers à recevoir.

Par défaut, sont reçus :

`bts` tous les rapports de bogue et les discussions qui suivent ;

`bts-control` les courriers d'information de control@bugs.debian.org lors des changements d'état de rapport de bogue ;

`upload-source` le courrier d'information de **dak** quand un paquet source est accepté ;

`katie-other` les autres courriers d'avertissement et d'erreur de **dak** (comme une incohérence de modification des champs de section ou de priorité) ;

`buildd` les courriers d'information envoyées par le réseau des démons de compilation, qui mentionnent un lien vers les journaux de compilation afin de les analyser ;

`default` tout courrier non automatique envoyé au PTS pour contacter les inscrits au paquet. Ceci peut être fait en envoyant un courrier à *`paquet-source`*`@packages.qa.debian.org`. Pour prévenir l'envoi de pourriels, tous les courriers envoyés à ces adresses doivent contenir l'en-tête `X-PTS-Approved` avec une valeur non vide ;

`contact` les courriers envoyés au responsable via l'alias `*@packages.debian.org` ;

`summary` les courriers de résumé réguliers sur l'état du paquet, comme la progression du paquet dans `testing`, les notifications de l'état de santé extérieur à Debian (« `Debian External Health Status` » ou `DEHS`) lors qu'une nouvelle version amont est disponible, et la notification si un paquet est enlevé de l'archive ou orphelin.

Il est également possible de recevoir des informations supplémentaires :

`upload-binary` les courriers d'information de **katie** quand un paquet binaire est accepté. En d'autres termes, à chaque fois qu'un démon de compilation ou un porteur envoie le paquet pour une architecture, un courrier est envoyé ce qui permet de suivre comment le paquet est recompilé pour toutes les architectures ;

`cvs` les annonces de nouvelle révision dans le système de gestion de versions (« `VCS commit` »), si le paquet est maintenu avec un tel système et que le responsable a mis en place un suivi de révisions vers le PTS. Le nom `cvs` est historique, la plupart du temps les notifications de révision proviendront d'autres logiciels de gestion de versions comme `Subversion` ou `Git` ;

`ddtp` les traductions de descriptions ou de questionnaires debconf soumis au projet de traduction des descriptions de paquets (« `Debian Description Translation Project` » ou `DDTP` ;

`derivatives` des informations sur les changements effectués sur le paquet dans les distributions dérivées (`Ubuntu` par exemple).

`derivatives-bugs` les rapports de bogue et leurs commentaires dans les distributions dérivées (`Ubuntu` par exemple).

4.10.1 Interface de courrier du PTS

Les inscriptions au PTS peuvent être administrées en envoyant différentes commandes à pts@qa.debian.org.

`subscribe <paquet-source> [<adresse>]` Inscrit l'*`adresse`* aux communications liées au paquet source *`paquet-source`*. L'adresse de l'expéditeur est utilisée si le second paramètre n'est pas présent. Si *`paquet-source`* n'est pas un paquet source valable, vous recevrez un avertissement. Cependant, s'il s'agit d'un paquet binaire valable, le PTS vous inscrira pour le paquet source correspondant.

unsubscribe <paquet-source> [<adresse>] Supprime une inscription au paquet source *paquet-source* en utilisant l'adresse spécifiée ou l'adresse de l'expéditeur si le second paramètre n'est pas rempli.

unsubscribeall [<adresse>] Supprime toutes les inscriptions de l'adresse spécifiée ou de l'adresse de l'expéditeur si le second paramètre n'est pas rempli.

which [<adresse>] Liste les inscriptions pour l'expéditeur ou pour l'adresse indiquée si elle est spécifiée.

keyword [<adresse>] Donne les mots-clés acceptés. Pour une explication de ces mots-clés, voir ci-dessus. Voici un rapide résumé :

- `bts` : courriers venant du système de gestion de bogues (BTS) Debian ;
- `bts-control` : réponses aux courriers envoyés à control@bugs.debian.org ;
- `summary` ; courriers de résumé automatique sur l'état d'un paquet ;
- `contact` ; courriers envoyés au responsable via l'alias `*@packages.debian.org` ;
- `cvs` : annonces de nouvelle révision (« `VCS commit` ») ;
- `ddtp` : traductions des descriptions et questionnaires debconf ;
- `derivatives` : changements effectués dans des distributions dérivées ;
- `derivatives-bugs` : rapports de bogue et leurs commentaires dans les distributions dérivées ;
- `upload-source` : annonce lorsqu'un nouveau paquet source a été accepté ;
- `upload-binary` : annonce lorsqu'un nouveau paquet binaire a été accepté ;
- `katie-other` : autres courriers des responsables de l'archive (incohérence de modification des champs, etc.) ;
- `buildd` : notifications d'erreur des démons de compilation ;
- `default` : tout autre courrier (non automatique).

keyword <paquet-source> [<adresse>] Identique à l'élément précédent, mais pour un paquet source donné car il est possible de sélectionner un ensemble de mots-clés différent pour chaque paquet source.

keyword [<adresse>] {+|-|=} <liste de mots-clés> Accepte (+) ou refuse (-) les courriers classés dans la liste de mots-clés. Définit la liste (=) des mots-clés acceptés. Ceci change l'ensemble par défaut des mots-clés acceptés par un utilisateur.

keywordall [<adresse>] {+|-|=} <liste de mots-clés> Accepte (+) ou refuse (-) les courriers classés dans la liste de mots-clés. Définit la liste (=) des mots-clés acceptés. Ceci change les mots-clés de toutes les inscriptions en cours d'un utilisateur.

keyword <sourcepackage> [<adresse>] {+|-|=} <liste de mots-clés> Identique à l'élément précédent, mais remplace la liste des mots-clés pour le paquet source indiqué.

quit | thanks | -- Arrête le traitement des commandes. Toutes les lignes suivantes sont ignorées par le robot.

L'utilitaire en ligne de commande **pts-subscribe** (du paquet `devscripts`) peut être pratique pour s'inscrire temporairement à certains paquets, par exemple après avoir fait une mise à jour indépendante (NMU).

4.10.2 Filtrer les courriers du PTS

Une fois inscrit à un paquet, vous recevrez les courriers envoyés à *paquet-source*`@packages.qa.debian.org`. Ces courriers ont des en-têtes spéciaux ajoutés pour vous permettre de les filtrer dans des boîtes aux lettres (avec **procmail** par exemple). Les en-têtes ajoutés sont `X-Loop`, `X-PTS-Package`, `X-PTS-Keyword` et `X-Unsubscribe`.

Voici un exemple d'en-têtes ajoutés pour une notification d'envoi de source sur le paquet `dpkg` :

```
X-Loop :dpkg@packages.qa.debian.org
X-PTS-Package :dpkg
X-PTS-Keyword :upload-source
List-Unsubscribe :<mailto pts@qa.debian.org ?body=unsubscribe+dpkg>
```

4.10.3 Faire suivre les annonces de révision vers le PTS

Si vous utilisez un système de gestion de versions accessible publiquement pour maintenir votre paquet Debian, vous pouvez faire suivre les notifications de modifications vers le PTS pour que les inscrits (ainsi que de possibles co-responsables) puissent suivre de près l'évolution du paquet.

Une fois le système de gestion de versions configuré pour générer des notifications de modifications, il suffit d'en envoyer une copie à *paquet-source*_cvs@packages.qa.debian.org. Seules les personnes ayant accepté le mot-clé cvs recevront les notifications. Si le message n'est pas envoyé depuis une machine du domaine debian.org, il faut ajouter l'en-tête X-PTS-Approved:1.

Pour les dépôts Subversion, il est conseillé d'utiliser svnmailer. Voir http://wiki.debian.org/Alioth/PackagingProject pour un exemple de mise en place.

4.10.4 Interface web du PTS

Le PTS possède une interface web sur http://packages.qa.debian.org/ qui réunit beaucoup d'informations pour chaque paquet source. Plusieurs liens utiles sont proposés (BTS, statistiques QA, informations de contact, état de traduction DDTP, journaux de compilation automatique) et beaucoup d'autres informations provenant de différents endroits sont regroupés (les 30 dernières entrées de changelog, l'état dans testing, etc.). C'est un outil très pratique pour connaître ce qu'il en est d'un paquet source spécifique. De plus, un formulaire permet de s'inscrire facilement au PTS par courrier.

Il est possible d'aller directement à la page web concernant un paquet source avec une URL comme http://packages.qa.debian.org/*paquet-source*.

Cette interface a été conçue comme un portail pour le développement des paquets : vous pouvez ajouter du contenu personnalisé aux pages de vos paquets. Vous pouvez ajouter des informations statiques (« static information » : annonces destinées à rester disponibles indéfiniment) et des nouvelles récentes (« latest news »).

Les annonces statiques peuvent être utilisées pour indiquer :

- la disponibilité d'un projet hébergé sur Alioth pour la co-maintenance du paquet ;
- un lien vers le site web amont ;
- un lien vers le suivi de bogues amont ;
- l'existence d'un canal IRC dédié au logiciel ;
- toute autre ressource disponible éventuellement utile à la maintenance du paquet.

Les nouvelles usuelles peuvent être utilisées pour annoncer que :

- des paquets bêta sont disponibles pour tester ;
- des paquets finaux sont attendus pour la semaine prochaine ;
- l'empaquetage est sur le point d'être intégralement refait ;
- des rétroportages sont disponibles ;
- le responsable est en vacances (s'il désire publier cette information) ;
- une mise à jour indépendante (NMU) est en cours de réalisation ;
- quelque chose d'important va affecter le paquet.

Les deux types d'informations sont fabriqués de façon similaire : il suffit d'envoyer un courrier à pts-static-news@qa.debian.org (pour les annonces statiques) ou pts-news@qa.debian.org (pour les nouvelles usuelles). Le courrier devrait indiquer quel paquet est concerné par la nouvelle en donnant le nom du paquet source dans un en-tête de courrier X-PTS-Package ou un pseudo-en-tête Package (comme pour les rapports de bogue du BTS). Si une URL est disponible dans l'en-tête de courrier X-PTS-Url ou dans un pseudo-en-tête Url, le résultat est un lien vers cette URL au lieu d'une nouvelle complète.

Voici quelques exemples de courriers valables utilisés pour générer des nouvelles dans le PTS. Le premier ajoute un lien vers l'interface viewsvn de debian-cd dans la section des informations statiques :

```
From :Raphael Hertzog <hertzog@debian.org>
To :pts-static-news@qa.debian.org
Subject :Browse debian-cd SVN repository

Package :debian-cd
Url :http ://svn.debian.org/viewsvn/debian-cd/trunk/
```

Le second est une annonce envoyée à une liste de diffusion et également envoyée au PTS pour qu'elle soit publiée sur la page web du PTS du paquet. Notez l'utilisation du champ BCC pour éviter que des réponses ne soient envoyées par erreur au PTS.

```
From :Raphael Hertzog <hertzog@debian.org>
To :debian-gtk-gnome@lists.debian.org
Bcc :pts-news@qa.debian.org
Subject :Galeon 2.0 backported for woody
X-PTS-Package :galeon

Hello gnomers !

I'm glad to announce that galeon has been backported for woody. You'll find
everything here :
...
```

Réfléchissez-y à deux fois avant d'ajouter une nouvelle au PTS car vous ne pourrez pas l'enlever par la suite et vous ne pourrez pas non plus la modifier. La seule chose que vous puissiez faire est d'envoyer une deuxième nouvelle qui va rendre la première obsolète.

4.11 Vue d'ensemble des paquets d'un développeur

Un portail web pour l'assurance qualité (« quality assurance » ou QA) sur http://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 est un service de Debian basé sur une version légèrement modifiée du logiciel FusionForge (qui a évolué à partir de SourceForge et GForge). Ce logiciel offre aux développeurs l'accès à des outils faciles d'utilisation comme un gestionnaire de suivi de bogues, un gestionnaire de correctifs, un gestionnaire de tâches et de projets, un service d'hébergement de fichiers, des listes de diffusion, des systèmes de gestion de versions, etc. Tous ces outils sont gérés à l'aide d'une interface web.

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 :

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

4.13 Avantages pour les développeurs

4.13.1 Abonnements à LWN

Depuis octobre 2002, HP parraine l'abonnement à LWN pour tous les développeurs Debian intéressés. Des détails sur les moyens d'accéder à cet avantage sont expliqués dans le message http://lists.debian.org/debian-devel-announce/2002/10/msg00018.html.

4.13.2 Remise sur l'hébergement `Gandi.net`

Depuis Novembre 2008, `Gandi.net` offre une remise sur leurs serveur dédié virtuel pour les développeurs Debian : http://lists.debian.org/debian-devel-announce/2008/11/msg00004.html.

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* ». Ceci 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 http://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.6.6. 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 éprouvées :

– 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 contenues dans le fichier .diff.gz.

5.4 Agencement du paquet source

Il existe deux types de paquets source Debian :

– les paquets natifs (« native ») pour lesquels il n'y a pas de distinction entre les sources d'origine et les correctifs appliqués pour Debian ;

– les paquets (plus courants) avec au moins une archive, contenant les sources d'origine, accompagnée d'un fichier, contenant les modifications pour Debian.

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

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

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

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

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

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

5.5 Choix de distribution

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

Il existe plusieurs valeurs possibles pour ce champ : `stable`, `unstable`, `testing-proposed-updates` et `experimental`. En principe, les paquets sont destinés à `unstable`.

En fait, il y a deux autres possibilités : `stable-security` et `testing-security`, voir 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-new` pour être revu par les responsables de la publication `stable`. Une fois accepté, le paquet sera installé dans le répertoire `stable-proposed-updates` de l'archive Debian. Il sera ensuite ajouté à `stable` lors de la prochaine mise à jour de la distribution.

Pour qu'un paquet soit accepté, vous devriez contacter l'équipe de publication `stable` avant de l'envoyer. Pour ce faire, soumettez un bogue sur le pseudopaquet `release.debian.org` en utilisant **reportbug** avec le correctif que vous avez l'intention d'appliquer à la version du paquet présent dans `stable`. Il faut toujours détailler précisément le journal de modification pour un envoi vers la distribution `stable`.

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.

Par le passé, les envois vers `stable` étaient également utilisés pour corriger les problèmes de sécurité. Cependant, cette pratique est déconseillée car les mises à jour pour les avis de sécurité Debian (« `Debian security advisory` » ou `DSA`) sont automatiquement copiés dans l'archive `proposed-updates` appropriée quand l'avis est publié. Reportez-vous en Section 5.8.5 pour des informations plus détaillées sur la gestion des problèmes de sécurité. Si l'équipe en charge de la sécurité estime le problème trop insignifiant pour justifier un `DSA`, les responsables de la publication `stable` seront cependant plus facilement disposés à intégrer votre correctif via un envoi ordinaire vers `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 impossible 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

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

Attention, il est préférable de transférer le fichier changes en dernier. Dans le cas contraire, votre envoi pourrait être rejetée 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 donner quelques jours pour 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. Ceci 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 de ces files d'attente.

5.6.3 Envois de sécurité

N'envoyez **jamais** un paquet vers la file d'envoi de sécurité oldstable-security, stable-security, etc.) sans avoir obtenu au préalable l'autorisation de l'équipe de sécurité. Si le paquet ne correspond pas tout à fait aux besoins de cette équipe, il entraînera beaucoup de problèmes et de retards dans la gestion de cet envoi non désiré. Pour plus de détails, voir 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 via 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 Notification d'installation de nouveau paquet

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 via les files d'attente, le démon vous enverra également une notification par courrier électronique.

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 pourrez corriger votre fichier `debian/control` avant votre prochain envoi ou alors vous pourrez vouloir 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/priorité, [...], PACKAGEX:section/priorité`, et exposez bien les raisons qui vous amènent à demander ces changements dans le corps de texte.

Pour en savoir plus sur les fichiers `override`, reportez-vous à dpkg-scanpackages(1) et http://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 http://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 : `http://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 http://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 paquet, 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 bogue ;

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 version, il peut être marqué `pending` (en attente) pour signaler qu'il est corrigé, et sera fermé à la prochaine mise à jour (ajouter « `closes:` » dans `changelog`). C'est d'autant plus utile si plusieurs développeurs travaillent sur le même paquet ;
8. une fois le paquet corrigé disponible dans l'archive, le bogue devrait être fermé en précisant dans quelle version du paquet il a été réglé. Ceci peut être fait automatiquement, voir 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
```

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

Historiquement, les envois identifiés comme 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, de préférence en remplir un ticket sur leur système de suivi de requêtes. Consultez `http://wiki.debian.org/rt.debian.org#Security_Team`. Sinon, vous pourriez envoyer un courrier à team@security.debian.org. **n'envoyez pas** de paquet pour `stable` sans contacter l'équipe de sécurité. Les informations utiles comprennent, par exemple :

- si le bogue a déjà été rendu public on non ;
- les versions du paquet affectées par le bogue. Vérifiez chaque version présente dans les distributions maintenues par Debian ainsi que dans `testing` et `unstable` ;
- la nature d'une solution si elle existe (les correctifs sont particulièrement utiles) ;
- tout paquet corrigé préparé par vous-même (envoyez seulement les fichiers `.diff.gz` et `.dsc` et lisez d'abord Section 5.8.5.4) ;
- 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 : quelles 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. Ceci 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 publique, 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 permettant 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.

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

Cela signifie que passer à une version amont supérieure n'est pas une bonne solution. À la place, les changements pertinents devraient être rétroportés vers la version présente dans 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, ceci 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 liés 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 :

- **ciblez la bonne distribution** dans votre fichier `debian/changelog`. Pour `stable` il s'agit de `stable-security`, pour `testing` il s'agit de `testing-security`, et pour l'ancienne distribution stable, il s'agit de `oldstable-security`. Ne ciblez ni *`distribution`*`-proposed-updates`, ni `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. Déclarez `closes:` pour tout **bogue Debian**. Intégrez toujours une référence externe, de préférence un **identifiant CVE**, pour qu'elle puisse être recoupée. Néanmoins, si aucun identifiant CVE n'a encore été assigné, ne l'attendez pas et continuez le processus. L'identifiant pourra être référencé plus tard ;
- assurez-vous que le **numéro de version** est correct. Il doit être plus élevé que celui du paquet actuel, mais moins que celui des paquets des distributions suivantes. En cas de doute, testez-le avec `dpkg --compare-versions`. Soyez attentif à ne pas réutiliser un numéro de version déjà utilisé pour un précédent envoi, ou qui entrerait en conflit avec une mise à jour indépendante binaire (`binNMU`). Par convention, ajoutez `+`*`nomdecode`*`1`, par exemple `1:2.4.3-4+lenny1`, bien sûr, incrémentez le nombre qui suit le nom de code (1 ici) lors des mises à jour suivantes ;
- à moins que l'archive sources 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 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

Vous ne devez **jamais** envoyer un paquet dans la file d'attente des envois de sécurité (`oldstable-security`, `stable-security`, etc.) sans l'accord préalable de l'équipe de sécurité. Si le paquet ne remplit pas exactement les exigences de l'équipe, il causera beaucoup de problèmes, ainsi que des retards dans la gestion de l'envoi indésirable.

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

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

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

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

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

5.9 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 distribuée 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 le placer 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 valide, 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

Pour supprimer complètement un paquet de l'archive (une vieille bibliothèque de compatibilité devenue inutile par exemple), il faudra envoyer un rapport de bogue sur le pseudopaquet `ftp.debian.org` et demander la suppression du paquet ; comme chaque bogue, il devrait être de gravité normale. Le titre du rapport devrait être de la forme `RM:`*`paquet [liste d'architectures] --raison`*, où *`paquet`* est le paquet à supprimer et *`raison`* un court résumé de la raison de la demande. *`[liste d'architectures]`* est facultatif, il n'est requis que si la demande ne concerne pas toutes les architectures. Remarquez que **reportbug** préparera un titre conforme à ces règles lors de la création d'un bogue sur le pseudopaquet `ftp.debian.org`.

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

Seuls les paquets d'`unstable`, `experimental` ou `stable` peuvent être supprimés. Les paquets de `testing` ne sont pas supprimés directement. Ils sont plutôt enlevés automatiquement après suppression d'`unstable` et si aucun paquet de `testing` n'en dépend. (Les suppressions de `testing` sont possibles aussi en soumettant un bogues de suppression sur le pseudopaquet `release.debian.org`. Consultez la section Section 5.13.2.2.)

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. Ceci a pour but d'éviter les suppressions indésirables et de garder une trace de la raison pour laquelle un paquet a été supprimé. Par exemple, vous pouvez fournir le nom du paquet qui remplace celui à supprimer.

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

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

Plus d'informations sur ce sujet et autres sujets connexes sont disponibles sur http://wiki.debian.org/ftpmaster_Removals et http://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 désirer 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 ceci 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.

Prendre un paquet parce qu'il vous semble négligé n'est pas correct — ce serait un détournement de paquet. Vous pouvez prendre contact avec le responsable actuel et lui demander si vous pouvez prendre en charge ce paquet. Si vous avez le sentiment qu'un responsable est parti sans prévenir (« `absent without leave` » ou `AWOL`), veuillez vous reporter à Section 7.4.

Normalement, vous ne pouvez pas adopter un paquet sans l'accord de son responsable. Même s'il vous ignore, ce n'est pas une raison pour le faire. Les plaintes à propos des responsables devraient être portées sur la liste de diffusion des développeurs. Si la discussion ne se termine pas par une conclusion positive et que le problème est de nature technique, envisagez de porter le cas à l'attention du comité technique (voir la page web du comité technique pour plus d'informations).

Si vous reprenez un vieux paquet, vous voudrez sûrement que le système de suivi des bogues indique que vous êtes le responsable du paquet. Cela se produira automatiquement une fois installé une nouvelle version du paquet dans l'archive avec le champ `Maintainer` à jour. Cela peut prendre quelques heures après l'envoi. Si vous ne pensez pas faire de mise à jour avant un moment, vous pouvez utiliser le Section 4.10 pour recevoir les rapports de bogue. Cependant, assurez-vous que l'ancien responsable n'est pas embêté de recevoir les rapports de bogues en attendant.

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.

Porter un paquet consiste à compiler un paquet binaire pour des architectures différentes de celle du paquet binaire du responsable du paquet. C'est une activité remarquable et essentielle. En fait, les porteurs sont à l'origine de la plupart des compilations de paquets Debian. Par exemple, quand un paquet source (portable) est envoyé avec les paquets binaires `i386`, il faut compter une compilation pour chaque autre architecture, soit un total de 11 compilations.

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

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

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

1. vérifiez que les champs `Build-Depends` et `Build-Depends-Indep` du fichier `debian/control` sont corrects. Le meilleur moyen de le vérifier est d'utiliser le paquet `debootstrap` pour créer un environnement `unstable chrooté` (voir Section A.4.2). Dans cet environnement `chrooté`, installez le paquet `build-essential` et tous les paquets mentionnés dans les champs `Build-Depends` ou `Build-Depends-Indep`. Ensuite, essayez de fabriquer le paquet dans cet environnement. Ces étapes peuvent être automatisées en utilisant le programme **pbuilder** fourni par le paquet de même nom (voir Section A.4.3).

   En cas de difficultés pour configurer un environnement `chrooté`, **dpkg-depcheck** pourra peut-être vous aider (voir Section A.6.7).

   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. assurez-vous de ne pas dépendre d'éléments de configuration, ou de logiciels installés ou modifiés localement. Par exemple, vous ne devriez jamais appeler des programmes du répertoire `/usr/local/bin` ou de répertoires équivalents. Essayez de ne pas dépendre de logiciels configurés de manière spéciale. Essayez de construire votre paquet sur une autre machine, même s'il s'agit de la même architecture ;
6. ne vous appuyez pas sur une installation préexistante du paquet (un sous-cas de la remarque précédente). Il existe, bien sûr, des exceptions à cette règle, mais soyez conscient que chaque cas comme celui-ci 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 car certaines architectures pourraient choisir un compilateur différent ;
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 dépendent les uns des autres via `$(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).

La « magie » d'une NMU par seule recompilation est déclenchée par l'utilisation d'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 le 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-buildpackage -B` pour ne construire que les parties dépendant de l'architecture du paquet.

2. Par le passé, ces NMU utilisaient le numéro de troisième niveau de la partie Debian de la révision pour indiquer l'état de seule recompilation ; cependant, cette syntaxe était ambiguë pour les paquets natifs et ne permettait pas d'ordonner correct les NMU par seule recompilation, les NMU source et les NMU de sécurité sur le même paquet, elle a donc été abandonnée en faveur de cette nouvelle syntaxe.

5.10.2.2 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. Ceci 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 d'implémenter des contournements pour des bogues de l'environnement de compilation, du noyau ou de la `libc`. Parfois, ces contournements sont inévitables. 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 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 `http://www.debian.org/ports/`.

Chaque portage de Debian possède sa propre liste de diffusion. La liste des listes de diffusion de portage peut être trouvée à `http://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`

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

`wanna-build` n'est pas encore disponible sous forme de paquet ; pourtant, tous les efforts de portage l'utilisent pour automatiser la compilation de paquets. L'outil de compilation vraiment utilisé est dans le paquet `sbuild`, voir la description en Section A.4.4. La version empaquetée n'est pas la même que celle utilisée sur les automates de compilation, mais suffisamment proche pour reproduire les problèmes.

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

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

L'équipe de wanna-build, en charge des empaqueteurs (« buildd »), peut être contactée à l'adresse électronique debian-wb-team@lists.debian.org. Pour savoir qui (équipe de wanna-build, équipe de publication) et comment (courrier électronique, BTS) contacter, se reporter à http://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 http://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 stable, 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 http://buildd.debian.org/quinn-diff/Packages-arch-specific ; veuillez consulter le début du fichier pour savoir qui contacter pour le modifier.

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

5.10.5 Paquets non libres pouvant être automatiquement construits

Par défaut, les paquet de la section non-free ne sont pas construits sur le réseau d'empaqueteurs automatiques (surtout parce que la licence des paquets pourrait l'interdire). Pour permettre à un paquet d'être construit, vous devez suivre les étapes suivantes :

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.

– Votre NMU corrige-t-elle vraiment des bogues ? La correction de problème superficiel ou la modification du mode d'empaquetage lors d'une NMU est déconseillée.

– Avez-vous laissé suffisamment de temps au responsable ? Quand le bogue a-t-il été reporté au BTS ? Être occupé pendant une semaine ou deux n'est pas exceptionnel. Le bogue est-il si grave qu'il doive être corrigé immédiatement, ou cela peut-il attendre encore quelques jours ?

– Quelle confiance avez-vous dans vos modifications ? Souvenez-vous du serment d'Hippocrate : « je m'abstiendrai de tout mal ». Il est préférable de laisser un paquet avec un bogue ouvert grave plutôt qu'appliquer un correctif non fonctionnel ou un correctif qui cache le bogue sans le résoudre. Si vous n'êtes pas absolument sûr de vous, il pourrait être judicieux de chercher des conseils autour de vous. Rappelez-vous que si quelque chose est cassé par votre NMU, de nombreuses personnes seront mécontentes.

– Avez-vous clairement manifesté votre intention de procéder à une NMU, au moins dans le BTS ? C'est aussi une bonne idée d'essayer de contacter le responsable par d'autres moyen (courrier personnel, IRC).

– Si le responsable est habituellement actif et réactif, avez-vous tenté de le contacter ? En général il est préférable que le responsable prenne en charge lui-même un problème et qu'il lui soit offert une chance d'examiner et corriger votre correctif, car il est censé être mieux placé pour découvrir d'éventuels problèmes que vous pourriez rater. C'est souvent un gain de temps pour tout le monde si le responsable a la possibilité d'envoyer lui même une correction.

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

Lors de la préparation du correctif, vous devriez connaître les pratiques spécifiques au paquet potentiellement utilisées par le responsable. Les prendre en compte réduit le fardeau d'intégrer les modifications dans les tâches quotidiennes (« `workflow` ») du paquet et augmente ainsi les chances d'intégration. Un bon endroit pour chercher d'éventuelles pratiques spécifiques est `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 réaction 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 ou important : cinq jours ;

– autres NMU : dix jours.

Ces délais sont simplement donnés à titre indicatifs. Dans certains cas, de tels 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'inscrire au paquet via 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 recommendations 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`

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

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

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

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

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

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

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

Lors d'un envoi de paquet vers `testing` ou `stable`, il est parfois nécessaire de créer une branche (« `fork` ») dans l'arbre de numérotation des versions. Pour cela, une version de la forme `+debXYuZ` devrait être utilisée, où *X* et *Y* sont les numéros de publication majeur et mineur et *Z* est un compteur qui commence à `1`. Lorsqu'un numéro de version n'est pas encore connu (c'est souvent le cas de `testing`, au début du cycle de publication), le plus petit numéro de publication plus grand que la dernière publication `stable` doit être utilisé. Par exemple, alors que `Lenny` (Debian 5.0) est stable, une NMU de sécurité pour un paquet dont la version est `1.5-3` devrait avoir la version `1.5-3+deb50u1`, alors qu'une NMU de sécurité vers `Squeeze` devrait prendre la version `1.5-3+deb60u1`. Après la publication de `Squeeze`, les envois de sécurité vers la distribution `testing` prendront comme version `+deb61uZ`, jusqu'à ce qu'il soit établi que la publication devienne Debian 6.1 ou Debian 7.0 (dans ce cas, les envois prendront comme version `+deb70uZ`).

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 une commutation 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`) et 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 binaire. 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

Les NMU sont des envois effectués par quelqu'un d'autre que le responsable attitré. Il existe un autre type d'envoi où le paquet n'est pas le sien : les envois de QA, qui sont des envois pour les paquets orphelins.

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

```
* QA upload.
```

Si vous voulez faire une NMU, et que le responsable ne semble pas actif, il est judicieux de vérifier le paquet pour voir s'il est orphelins (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 http://qa.debian.org/orphaned.html.

Plutôt que faire un envoi de QA, vous pouvez envisager l'adoption du paquet en devenant son responsable. Vous n'avez besoin de la permission de personne pour adopter un paquet orphelin, il suffit de vous configurer comme responsable et d'envoyer la nouvelle version (voir Section 5.9.5).

5.11.7 NMU et envoi d'équipe

Parfois, vous corrigez ou envoyez un paquet car vous êtes membre d'une équipe de responsables (qui utilise une liste de diffusion comme responsable (Maintainer ou Uploader), voir Section 5.12), mais vous ne voulez pas vous ajouter comme co-responsable (Uploaders) car vous n'avez pas l'intention de participer régulièrement à ce paquet. Si cela est conforme avec la politique de votre équipe, vous pouvez procéder à un envoi normal sans être listé parmi les responsables (Maintainer ou Uploader). Dans ce cas, vous devriez commencer l'entrée du journal de modification (changelog) par la ligne suivante :

```
* Team upload.
```

5.12 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 champs Uploaders du fichier debian/control.

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

- donner au co-responsable un accès aux sources à partir desquelles vous construisez le paquet. Habituellement, cela implique que vous utilisiez un système de gestion de version comme CVS ou Subversion. Alioth (voir Section 4.12) fournit entre autres de tels outils ;
- 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>
```

– en utilisant le PTS (Section 4.10), les co-responsables devraient s'inscrire eux-mêmes aux paquet source.

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

En tout cas, il faut éviter de placer automatiquement tous les membres de l'équipe dans le champ Uploaders. Cela encombre la vue d'ensemble des paquets d'un développeur (voir 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.

5.13 La distribution testing

5.13.1 Bases

Les paquets sont habituellement installés dans la distribution testing après avoir été suffisamment éprouvés 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.

5.13.2 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 le champ d'urgence de l'envoi (élevée, moyenne ou basse). Veuillez noter que cette urgence est « collante » (« sticky »), ce qui signifie que l'envoi avec l'urgence la plus élevée depuis la précédente transition dans testing est prise en compte ;

– 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.

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

5.13.2.1 Désynchronisation

Pour le script de migration dans testing, « désynchronisé » (« outdated ») signifie : il y a différentes versions dans unstable pour les architectures de publication (à l'exception des architectures dans fuckedarches ; fuckedarches est une liste des architectures qui ne suivent pas le rythme (dans update_out.py), mais actuellement cette liste est vide). « Désynchronisé » n'a rien à voir avec les architectures que le paquet fournit pour testing.

Considérons cet exemple :

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

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

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

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

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

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

5.13.2.2 Suppression de `testing`

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

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

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

5.13.2.3 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, ceci 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.

5.13.2.4 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.

5.13.2.5 Détails

Suivent quelques informations sur le fonctionnement de britney.

Les paquets sont examinés pour savoir si ce sont des candidats valides. 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 valides. Pour ce faire, britney essaye d'ajouter chaque candidat valide 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 tel 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 y jetez un œil en http://ftp-master.debian.org/testing/update_output/.

Les coups de pouce (« hints ») sont disponibles sur http://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 remplacer l'urgence.

5.13.3 Mises à jour directes dans testing

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

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

Vous ne devriez pas envoyer de paquet à testing-proposed-updates quand vous pouvez le mettre à jour via 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 tire pas de nouvelles dépendances.

Les numéros de version sont habituellement choisis en ajoutant le nom de code de la distribution testing et un numéro incrémenté, comme 1.2squeeze1 pour le premier envoi dans testing-proposed-updates du paquet en version 1.2.

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) ;
- vérifiez avoir bien indiqué testing ou testing-proposed-updates comme distribution cible ;
- 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.

5.13.4 Foire aux questions

5.13.4.1 Quels sont les bogues bloquant l'intégration dans la version stable et comment sont-ils comptés ?

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

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

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

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

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

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

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

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

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

Chapitre 6

Meilleures pratiques d'empaquetage

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

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

6.1 Meilleures pratiques pour `debian/rules`

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

6.1.1 Scripts d'assistance

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

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

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.

Certains responsables pensent que l'utilisation de fichiers `debian/rules` sans assistants est préférable car elle évite d'avoir à apprendre les subtilités de ces systèmes d'assistance. Utiliser l'une ou l'autre méthode est entièrement à la discrétion du responsable d'un paquet qui devrait choisir la méthode qui lui convient le mieux. De nombreux

exemples de fichiers debian/rules qui n'utilisent pas d'assistants sont disponibles à l'adresse http://arch.debian.org/arch/private/srivasta/.

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.

Heureusement, avec le format source « 3.0 (quilt) », il est dorénavant possible de séparer des correctifs (« patches ») sans avoir à modifier debian/rules pour configurer un système de correctif. Les correctifs sont conservés dans debian/patches/ et, lorsque le paquet source est dépaqueté (« unpacked »), les correctifs énumérés dans debian/patches/series sont automatiquement appliqués. Comme le nom le suggère, les correctifs peuvent être gérés avec **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, via une dépendance à la règle patch. À l'inverse, les correctifs sont retirés par la règle clean, via une dépendance à la règle unpatch.

quilt est l'outil recommandé pour ce besoin. Il effectue l'ensemble des actions mentionnées précédemment et offre la possibilité de gérer des ensembles de correctifs. Veuillez regarder le paquet quilt pour plus d'informations.

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.

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

6.2 Meilleures pratiques pour debian/control

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

La description d'un paquet telle que définie par le champ correspondant du fichier control, comprend à la fois le résumé et la description longue du paquet. 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 inappropriés : 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 toute les informations nécessaires pour déterminer si le paquet doit être installé. Elle complète le résumé qui est donc supposé avoir été lu précédemment.

La description longue est constituée de phrases complètes.

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

Les paragraphes suivants devraient répondre aux questions : « Pourquoi, en tant qu'utilisateur, ai-je besoin de ce paquet ? », « Quelles autres fonctionnalités ce paquet apporte-t-il ? », « Quelles fonctionnalités et défauts comporte-t-il par rapport à d'autres paquets (par exemple, « si vous avez besoin de X, utilisez plutôt Y ») ? », « Ce paquet est-il lié à d'autres paquets d'une manière non gérée par le système de gestion des paquets (par exemple, « ceci est le client destiné au serveur toto ») ? ».

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

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

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

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

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

6.2.4 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-*

La valeur de ce champ doit être une chaîne identifiant sans équivoque l'emplacement du dépôt de gestion de versions utilisé pour la maintenance de ce paquet, s'il est disponible. `*` doit être remplacé par le système de gestion de versions. Les systèmes suivants sont actuellement gérés par le système de suivi des paquets : `arch`, `bzr` (Bazaar), `cvs`, `darcs`, `git`, `hg` (Mercurial), `mtn` (Monotone), `svn` (Subversion). Il est possible d'indiquer plusieurs champs VCS pour le même paquet : ils seront alors tous mentionnés dans l'interface web du système de suivi des paquets.

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 :http ://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 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.

6.3.3 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.
```

Si, pour une raison donnée, vous avez omis de mentionner un numéro de bogue dans une entrée précédente, ce n'est pas grave : il suffit de clôturer le bogue normalement dans le système de suivi des bogues. Il est inutile de changer le journal des modifications si on suppose que les explications sur la correction du bogue sont dans le bogue lui-même (cela s'applique également au suivi des bogues des auteurs amont : il est inutile de suivre, dans le journal des modifications, les bogues qu'ils ont corrigés depuis longtemps).

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

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

6.3.4 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

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

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 « `sheb ang` » 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 [ -x /usr/sbin/install-docs ]; then ...
```

Pour ne pas coder en dur le chemin d'une commande dans les scripts de responsable, la fonction shell conforme à POSIX suivante peut aider :

```
pathfind() {
    OLDIFS="$IFS"
    IFS= :
    for p in $PATH; do
        if [ -x "$p/$*" ]; then
            IFS="$OLDIFS"
            return 0
        fi
    done
    IFS="$OLDIFS"
    return 1
}
```

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

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.

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

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

6.5.2 Recommandations générales pour les auteurs et les traducteurs

6.5.2.1 Utilisation d'un anglais correct

La plupart des responsables de paquets 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

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

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

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

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

Une façon recommandée de se servir de cet utilitaire est :

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

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

La mention d'une date limite aux traducteurs est toujours appréciée, pour leur permettre d'organiser leur travail. Veuillez ne pas oublier que certaines équipes ont formalisé leur processus de traduction et révision de telle sorte qu'un délai inférieur à dix jours n'est pas considéré comme raisonnable. Un délai plus court met trop de pression sur les équipes de traduction et ne devrait être réservé qu'aux modifications mineures.

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

6.5.2.3 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

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

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

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

6.5.2.5 Proscrire l'utilisation de la première personne

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

6.5.2.6 Neutralité en genre

Le monde est fait d'hommes et de femmes. Veuillez utiliser des constructions neutres en genre dans vos écrits.

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

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

6.5.3.2 `Description` : descriptions courte et étendue

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

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

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

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

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

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

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 feront 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`

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

6.5.3.4 `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 Guide de style spécifique à certains modèles

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 (« `IP address:` »). 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 répéter la même demande. Utilisez des phrases complètes. Un style d'écriture abrégé est déconseillé.

6.5.4.2.2 Modèles `boolean`

- La partie courte devrait utiliser la forme interrogative et se terminer par un point d'interrogation. Un style abrégé est toléré et même encouragé si la question est complexe (les traductions vont être plus longues que la version originale).
- 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`

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

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.
- **N'abusez pas de `debconf`.** Les notes sont un des abus les plus fréquents de debconf. Comme indiqué dans la page de manuel de debconf, elles devraient être réservées pour avertir les utilisateurs de problèmes très importants. Les fichiers `NEWS.Debian` ou `README.Debian` sont les endroits appropriés pour l'information qu'affichent la majorité des notes. Si, à la lecture de ces conseils, vous envisagez de convertir vos modèles de type note en entrée dans `NEWS.Debian` ou `README.Debian`, pensez à conserver d'éventuelles traductions existantes.

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`

Si la valeur par défaut d'un modèle `select` peut être dépendante de la langue utilisée (par exemple s'il s'agit du choix d'une langue par défaut), pensez à utiliser l'astuce « `_Default` ».

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

Exemple, pris dans le paquet geneweb :

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

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

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

6.5.4.5 Champ `Default`

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

Quand vous utilisez `po-debconf`, (et vous **devriez**, voir Section 6.5.2.2), veuillez rendre ce champ traduisible si vous pensez qu'il peut l'être.

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

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

6.6.1 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.

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

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 version.

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

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

6.7 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 bibliothèques.

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.

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

6.7.4 Catégories particulières de paquets

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

- Les paquets liés à Perl utilisent une charte Perl. Des exemples de tels paquets qui appliquent cette charte spécifique sont `libdbd-pg-perl` (module Perl binaire) ou `libmldbm-perl` (module Perl indépendant de l'architecture).
- Les paquets liés à Python utilisent une charte Python. Veuillez consulter le fichier `/usr/share/doc/python/python-policy.txt.gz` du paquet `python` pour plus d'informations.
- Les paquets liés à Emacs utilisent une charte Emacs.
- Les paquets liés à Java utilisent une charte Java.
- Les paquets liés à Ocaml utilisent leur propre charte, que l'on peut trouver dans le fichier `/usr/share/doc/ocaml/ocaml_packaging_policy.gz` du paquet `ocaml`. Un bon exemple est fourni par le paquet source `camlzip`.
- Les paquets fournissant des DTD XML ou SGML devraient suivre les recommandations données dans le paquet `sgml-base-doc`.
- 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.

Cependant, si cette taille est importante, vous devriez réfléchir à les fournir dans un paquet séparé, indépendant de l'architecture (`_all.deb`). Cela permet ainsi d'éviter la duplication des mêmes données dans de nombreux paquets binaires, un par architecture. Bien que cela ajoute des entrées dans les fichiers `Packages`, cela permet d'économiser une place importante sur les miroirs de Debian. La séparation des données indépendantes de l'architecture réduit également le temps de traitement de **lintian** (voir Section A.2) lorsqu'il est utilisé sur l'archive Debian en entier.

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 bibliothèques 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**.

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

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

1. Il est impossible d'empêcher les auteurs amont de modifier l'archive qu'ils distribuent sans également incrémenter le numéro de version. Il est donc impossible de garantir qu'une archive originelle est identique à ce que l'auteur amont *distribue* à un instant donné. Tout ce qu'il est possible de garantir est qu'elle a été identique à ce que les auteurs amont *ont distribué* à un moment donné. Si une différence apparaît plus tard (par exemple si les auteurs amont découvrent ne pas avoir utilisé la compression maximale dans leur distribution d'origine et la recompresse,

vérifier que toutes les modifications effectuées entre la version Debian et la version amont sont contenues dans le fichier de différences Debian. De même, si la taille des sources d'origine est importante, les auteurs amont et tous ceux qui disposent de l'archive amont d'origine peuvent économiser du temps de téléchargement s'ils souhaitent contrôler le paquet en détail.

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

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

```
zcat path/to/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. **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.

c'est tout simplement dommage. Comme il n'existe pas de méthode adaptée pour envoyer un nouveau fichier `.orig.tar.{gz,bz2,xz}` pour la même version, il est même totalement inutile de traiter cette situation comme un bogue.

2. Avec pour exception particulière, si l'omission de fichiers non libres provoque une erreur de compilation sans l'aide du fichier de modification Debian, de pouvoir modifier les fichiers pour enlever les portions non libres, ou d'expliquer la situation dans un fichier `README.source` à la racine de l'arbre des sources. Veuillez dans ce cas solliciter l'auteur amont de rendre les portions non libres faciles à séparer du reste des sources.

6.7.8.3 Modification de fichier binaire

Il est parfois nécessaire de modifier les fichiers binaires contenus dans l'archive d'origine, ou d'ajouter des fichiers binaires. C'est tout à fait possible avec les paquets au format « 3.0 (quilt) ». Consultez la page de manuel dpkg-source(1) pour plus de détails. Avec le plus ancien format « 1.0 », `.diff.gz` ne peut pas contenir de fichiers binaires, ce qui oblige à utiliser **uuencode** (ou une fonction similaire) pour les stocker, puis de les reconstruire lors de la compilation dans `debian/rules` (et les remettre à leur place).

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.

C'est à la discrétion des responsables de paquet de créer ou non un paquet de débogage. Il est conseillé de créer des paquets de débogage pour les bibliothèques, puisque cela peut faciliter le débogage de nombreux programmes liés à ces bibliothèques. Normalement, les paquets de débogage n'ont pas besoin d'être ajoutés systématiquement, sinon la taille de l'archive augmenterait considérablement. En revanche, si un responsable estime que des utilisateurs peuvent avoir souvent besoin d'une version de débogage de son programme, il peut être judicieux de fournir un paquet de débogage. Les programmes faisant partie des applications principales de l'infrastructure, comme Apache ou le serveur X, sont également de bons candidats pour les paquets de débogage.

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

```
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`).

La description longue du métapaquet doit clairement expliquer son objectif, afin d'informer les utilisateurs sur ce qu'ils perdront s'ils suppriment le paquet. Vous devriez être explicite sur les conséquences. C'est tout particulièrement important pour les métapaquets installés lors de l'installation initiale qui n'ont pas été installés explicitement par l'utilisateur. Ils ont tendance à être importants pour garantir les mises à niveau du système et la description devrait essayer de dissuader les utilisateurs de les désinstaller pour éviter d'éventuels dommages.

Chapitre 7

Au-delà de l'empaquetage

Debian, c'est beaucoup plus que de l'empaquetage de logiciels et de la maintenance de paquets. Ce chapitre contient des informations sur les façons, souvent vraiment importantes, de contribuer à Debian au-delà 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 `http://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 `http://bugs.debian.org/from:`*`votre-adresse-de-courrier`*.

7.1.1 Signalement d'un grand nombre de bogues en une fois (« `mass bug filing` »)

Signaler de nombreux bogues pour le même problème sur un grand nombre de paquets — 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 ...
```

Remarquez que les étiquettes sont séparées par des espaces et ne peuvent contenir de tiret bas. Si vous signalez des bogues au nom d'un groupe ou d'une équipe spécifique, il vaut mieux définir `User` comme une liste diffusion appropriée après y avoir décrit votre intention.

Pour voir les bogues marqués par une étiquette d'utilisateur en particulier, rendez-vous sur la page `http://bugs.debian.org/cgi-bin/pkgreport.cgi?users=`*adresse-mail*`&tag=`*nom-d-etiquette*.

7.2 Effort d'assurance qualité

7.2.1 Travail quotidien

Bien qu'il y ait un groupe de personnes dédié à l'assurance qualité, 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 5.12).

7.2.2 Chasses aux bogues

De temps en temps, le groupe d'assurance qualité organise des chasses aux bogues (« `Bug Squashing Party` ») pour essayer de résoudre autant de problèmes que possible. Elles sont annoncées sur debian-devel-announce@lists.debian.org en précisant quel domaine sera visé pendant la chasse : habituellement, il s'agit des bogues empêchant l'intégration du paquet dans la distribution (bogues de gravité « `Release Critical` »), mais il peut être décidé d'aider à finir une transition majeure (comme une nouvelle version de Perl qui demande la recompilation de tous les modules binaires).

Les règles pour les mises à jour indépendantes (`NMU`) sont différentes au cours de la chasse parce que l'annonce de la chasse est considérée comme une annonce préalable pour les NMU. Si vous avez des paquets qui peuvent être affectés par la chasse (parce qu'ils ont des bogues critiques par exemple), vous devriez envoyer une mise à jour pour chaque bogue correspondant pour expliquer leur état actuel et ce que vous attendez de la chasse. Si vous ne voulez pas de NMU, si vous n'êtes intéressé que par un correctif, ou si vous voulez gérer vous-même le bogue, veuillez l'expliquer dans le BTS.

Les personnes qui participent à la chasse ont des règles spécifiques pour les NMU, elles peuvent en faire une sans avertissement préalable si elles envoient leur paquet avec un délai d'au moins trois jours dans `DELAYED/3-day`. Toutes les autres règles de NMU s'appliquent comme d'habitude ; le correctif de la NMU devrait être envoyé

dans le BTS (pour l'un des bogues ouverts corrigé par la NMU ou pour un nouveau bogue marqué corrigé). Les participants devraient également respecter tout souhait du responsable s'il en a exprimé.

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

Vous pouvez également vouloir contacter les personnes inscrites à un paquet source donné, cf. Section 4.10. Vous pouvez le faire en utilisant l'adresse *paquet*`@packages.qa.debian.org`.

7.4 Gestion des responsables non joignables

Si vous remarquez qu'un paquet manque de maintenance, vous devriez vous assurer que le responsable est toujours actif et qu'il continue à travailler sur ses paquets. Il est possible qu'il ne soit plus actif, mais qu'il n'ait pas démissionné du système. D'un autre côté, il est possible qu'il ait simplement besoin d'un rappel.

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.

Si le responsable ne répond pas après quatre semaines (un mois), on peut supposer qu'il n'y aura probablement pas de réponse. Si ceci se produit, vous devriez poursuivre vos investigations et essayer de réunir toutes les informations utiles sur ce responsable. Ceci 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 via 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 !

D'un autre côté, bien que tout le monde soit volontaire, tout le monde est responsable. Vous pouvez donc insister sur l'importance du plus grand intérêt — si un responsable n'a plus le temps ou l'envie, il devrait « laisser filer » et donner le paquet à quelqu'un ayant plus de temps.

Si vous êtes intéressé pour travailler dans l'équipe MIA, veuillez étudier le fichier `README` dans `/org/qa.debian.org/mia` sur `qa.debian.org` où les détails techniques et les procédures MIA sont documentés et contactez mia@qa.debian.org.

7.5 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

Parrainer un paquet signifie envoyer un paquet pour un responsable qui n'est pas encore autorisé à le faire lui-même. Ce n'est pas une mince affaire, le parrain doit vérifier l'empaquetage et s'assurer qu'il respecte le haut niveau d'exigence qualité que Debian s'efforce de conserver.

Les développeurs Debian peuvent parrainer des paquets, mais pas les 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. le parrain télécharge (ou extrait du dépôt) le paquet source ;
3. le parrain vérifie le paquet source. En cas de problème, il informe le responsable et lui demande de fournir une version corrigée (le processus reprend à la première étape) ;
4. le parrain ne trouve plus aucun problème. Il construit le paquet, le signe, et l'envoie dans Debian.

Avant de se plonger dans les détails du parrainage de paquet, vous devriez vous demander si l'ajout du paquet proposé est profitable à Debian.

Il n'y a pas de règle simple pour répondre à cette question, cela peut dépendre de plusieurs facteurs : le code source amont est-il suffisamment achevé et pas miné de trous de sécurité ? Existe-t-il déjà des paquets qui font la même chose et que donne la comparaison avec ce nouveau paquet ? Le paquet a-t-il été demandé par des utilisateurs et le nombre d'utilisateurs est-il significatif ? Les développeurs amont sont-ils actifs ?

Vous devriez également vérifier que le futur responsable sera un bon responsable. A-t-il déjà de l'expérience avec d'autres paquets ? Si oui, réalise-t-il du bon travail avec ceux-ci (contrôlez quelques bogues) ? Est-il familier avec le paquet et son langage de programmation ? A-t-il les compétences nécessaires pour ce paquet ? Si non, est-il capable de les 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

Les nouveaux responsables ont souvent quelques difficultés à créer des paquets Debian — ceci est bien compréhensible. Ils feront des erreurs. C'est pourquoi le parrainage d'un tout nouveau paquet dans Debian demande une inspection minutieuse de l'empaquetage. Parfois plusieurs itérations seront nécessaires avant que le paquet ne soit assez bon pour être envoyé dans Debian. Ainsi, être un parrain signifie être un mentor.

Ne parrainez jamais de paquet sans l'avoir vérifié. La vérification de nouveau paquet réalisée par les responsables de l'archive veille principalement à ce que le logiciel soit vraiment libre. Bien sûr, ils tombent parfois sur des problèmes d'empaquetage, mais ça ne devrait vraiment pas arriver. Il est de votre responsabilité de vérifier que le paquet envoyé est compatible avec les principes du logiciel libre selon Debian et qu'il est de bonne qualité.

La construction du paquet et l'essai du logiciel fait partie de la vérification, mais ça ne suffit pas. La suite de cette section est une liste non exhaustive de points à vérifier lors de votre contrôle. [1]

- Vérifiez que l'archive source fournie est la même que celle distribuée par l'auteur amont (quand les sources ont été réempaquetées pour Debian, créez vous-même l'archive modifiée).
- Exécutez **lintian**. De nombreux problèmes usuels seront découverts. Prenez soin de vérifier que tous les contrôles de **lintian** ignorés par le responsable sont pleinement justifiés.
- Exécutez **licensecheck** (qui fait partie de Section A.6.1) et vérifiez que `debian/copyright` a 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 quand les dépendances ne sont pas installées ? Est-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) ?
- Construisez les paquets, installez-les et essayez le logiciel. Vérifiez de pouvoir supprimer et purger les paquets. Essayez-les si possible avec **piuparts**.

Si la vérification n'a révélé aucun problème, vous pouvez construire le paquet et l'envoyer dans Debian. Rappelez-vous que même sans être le responsable du paquet, le parrain est toujours responsable de ce qu'il envoie dans Debian. C'est pourquoi vous devriez suivre le paquet avec le Section 4.10.

Remarquez que vous ne devriez pas avoir à modifier le paquet source pour mettre votre nom dans les fichiers `changelog` ou `control`. Le champ `Maintainer` du fichier `control` et le fichier `changelog` devraient afficher la personne qui a fait l'empaquetage, c'est-à-dire, le filleul. Ainsi, il recevra tous les courriers du système de suivi des bogues (BTS).

À la place, vous devriez indiquer à **dpkg-buildpackage** d'utiliser votre clef en signature. L'option `-k` le permet :

```
dpkg-buildpackage -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

Vous pourrez en général supposer que le paquet a déjà été vérifié complètement. Au lieu de recommencer depuis le début, vous devriez analyser précautionneusement les différences entre la version actuelle et la nouvelle version préparée par le responsable. Si vous n'avez pas fait la vérification initiale vous-même, vous pourriez tout de même regarder de plus près au cas où elle aurait été négligée.

Afin de comparer les différences, il vous faudra les deux paquets. Téléchargez la version actuelle du paquet source (avec **apt-get source**) et reconstruisez-le (ou téléchargez les paquets binaires actuels avec **aptitude download**). Téléchargez le paquet source à parrainer (normalement avec **dget**).

Lisez la nouvelle entrée du journal de modification, elle devrait vous apprendre ce que vous devriez trouver lors de la vérification. L'outil principal à utiliser est **debdiff** (du paquet `devscripts`), vous pouvez l'exécuter avec deux paquets source (fichiers `.dsc`), deux paquets binaires, ou deux fichiers `.changes` (seront alors comparés tous les paquets binaires présents dans le fichier `.changes`).

1. D'autres vérifications sont disponibles dans le wiki où plusieurs développeurs partagent leurs propres listes de contrôle.

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.

Si tout est correct, construisez le paquet et comparez les paquets binaires pour vérifier que les modifications du paquet source n'ont pas des conséquences inattendues (comme certains fichiers supprimés par erreur, des dépendances manquantes, etc.)

Vous pourriez consulter le système de suivi des paquets (consultez Section 4.10) pour vérifier que le responsable n'a rien oublié d'important. Des mises à jour de traductions pourraient attendre d'être intégrées dans le BTS. Le paquet pourrait avoir été la cible d'une NMU précédente et le responsable pourrait avoir oublié d'intégrer les modifications apportées. Un bogue critique pour la publication oublié pourrait empêcher la migration vers `testing`. Si vous trouvez quelque chose à améliorer, il est temps de le signaler pour que le responsable puisse s'améliorer la prochaine fois, et ainsi lui donner l'opportunité de mieux comprendre ses responsabilités.

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

7.5.2 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 » d' « internationalisation »). C'est pourquoi, même si des programmes seulement en anglais vous suffisent, vous devriez lire la plupart de ce chapitre.

Selon l'introduction à l'i18n de Tomohiro KUBOTA, « I18N (internationalisation) signifie la modification d'un logiciel ou des technologies liées pour qu'un logiciel puisse potentiellement gérer des langues multiples, des conventions multiples et ainsi de suite dans le monde entier » alors que « L10N (localisation) signifie l'implémentation dans une langue spécifique pour un logiciel déjà internationalisé ».

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.

Pour les messages des programmes, l'infrastructure gettext est utilisée pour la plupart d'entre eux. La plupart du temps, la traduction est gérée en amont dans des projets comme le projet de traduction libre, le projet de traduction de Gnome ou celui de KDE. La seule ressource centralisée dans Debian est le centre de traduction de Debian où vous pouvez trouver des statistiques sur les fichiers de traduction trouvés dans les paquets, mais il n'y a aucune infrastructure pour faciliter le processus de traduction.

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 version 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 version), 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 version 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

Pour traduire des descriptions de paquet ou des questionnaires `debconf`, vous n'avez rien à faire, l'infrastructure du DDTP répartira le matériel à traduire aux volontaires sans besoin d'interaction de votre 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 ».

Certaines personnes préfèrent utiliser des outils de haut niveau, d'autres pas. Debian n'a pas de position officielle sur la question ; tout outil conviendra du moment qu'il fait le boulot. C'est pourquoi cette section n'a pas été conçue pour indiquer à chacun quel outil il doit utiliser ou comment il devrait faire pour gérer sa charge de responsable. Elle n'est pas non plus destinée à favoriser l'utilisation d'un outil aux dépens d'un autre.

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`.

Beaucoup pensent que ce système devrait être utilisé pour tout paquet nécessitant une configuration interactive, cf. Section 6.5. `debconf` n'est pas requis par la Charte Debian pour le moment, mais cela pourrait changer.

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`, vous pouvez construire un paquet en tant que simple utilisateur : `dpkg-buildpackage -rfakeroot`.

A.2 Contrôle de paquets (« `lint` »)

Selon le « `Free On-line Dictionary of Computing` » (FOLDOC), « `lint` » est « un outil de traitement de langage C qui contient beaucoup plus de tests complets sur le code que n'en font habituellement les compilateurs C ». Les outils de contrôle de paquets aident les responsables à découvrir automatiquement les problèmes habituels et les violations de Charte dans leurs paquets.

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 http://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 regroupe un ensemble de programmes pouvant être utilisés dans debian/rules pour automatiser les tâches courantes relatives à la fabrication de paquets Debian binaires. debhelper inclut des programmes pour installer différents fichiers, les compresser, ajuster leurs droits et intégrer votre paquet dans le système de menu Debian.

À la différence d'autres approches, debhelper est divisé en plusieurs petits utilitaires simples qui agissent de manière cohérente. Ce découpage permet un contrôle des opérations plus fin que certains des autres outils pour debian/rules.

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

A.3.2 dh-make

dh-make contient **dh_make**, un programme qui crée un squelette de fichiers nécessaires à la construction d'un paquet Debian à partir d'une arborescence source. Comme le nom le suggère, **dh_make** est une réécriture de debmake et ses fichiers modèles utilisent les programmes **dh_*** de debhelper.

Quoique les fichiers de règles fabriqués par **dh_make** constituent en général une base suffisante pour un paquet fonctionnel, ce ne sont que les fondations : la charge incombe toujours au responsable d'affiner les fichiers générés et de rendre le paquet complètement fonctionnel et en conformité avec la Charte.

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

Les paquets suivants facilitent le processus de construction des paquets, en contrôlant globalement **dpkg-buildpackage** ainsi que la gestion des tâches.

A.4.1 `cvs-buildpackage`

`cvs-buildpackage` permet de mettre à jour ou de récupérer des paquets source dans un référentiel CVS, il permet de fabriquer un paquet Debian depuis le référentiel CVS et assiste le responsable lors de l'intégration de modifications amont dans le référentiel.

Ce paquet fournit l'infrastructure facilitant l'utilisation de CVS pour le responsable Debian. Il permet de conserver des branches CVS 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` construit un système « `chrooté` » et compile des paquets dans ce système. C'est très pratique pour vérifier que les dépendances de compilation d'un paquet sont correctes et pour s'assurer qu'aucune dépendance de construction inutile ou incorrecte n'existe dans le paquet résultant.

`pbuilder-uml` est un paquet similaire, qui va même plus loin en réalisant la construction au sein d'un environnement « `User Mode Linux` ».

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 `http://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`

`dput` fait à peu près la même chose que `dupload`, mais autrement. Il possède quelques fonctions supplémentaires comme la possibilité de vérifier la signature GnuPG et les sommes de contrôles avant l'envoi, ainsi que la possibilité de démarrer **dinstall** en mode simulation (« `dry-run` ») après l'envoi.

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` contient des scripts et outils très pratiques pour maintenir les paquets Debian. Parmi ces scripts, **debchange** et **dch** manipulent le fichier `debian/changelog` en ligne de commande et **debuild** est une surcouche à **dpkg-buildpackage**. L'utilitaire **bts** est aussi très pratique pour mettre à jour l'état des rapports de bogue en ligne de commande. **uscan** permet de suivre les nouvelles versions amont de vos paquets. **debrsign** permet de signer un paquet à distance avant de l'envoyer, ce qui est bien utile quand la machine de construction de paquet est différente de celle où résident vos clés GPG.

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

A.6.2 autotools-dev

`autotools-dev` contient les meilleurs pratiques pour les responsables des paquets qui utilisent **autoconf** ou **automake**. Il contient également les fichiers canoniques `config.sub` et `config.guess`, connus pour fonctionner avec tous les portages Debian.

A.6.3 dpkg-repack

`dpkg-repack` crée un paquet Debian à partir d'un paquet déjà installé. Si des changements ont été effectués sur le paquet depuis qu'il a été installé (des fichiers de `/etc` modifiés par exemple), le nouveau paquet héritera de ces changements.

Cet utilitaire peut faciliter la copie de paquet d'un ordinateur à un autre, ou la recréation de paquets installés sur un système qui ne sont plus disponibles ailleurs, ou pour sauvegarder l'état actuel d'un paquet avant de le mettre à jour.

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 debsums

`debsums` vérifie des paquets installés par rapport à leurs sommes de contrôle MD5. Remarquez que les paquets n'ont pas tous des sommes de contrôle MD5 car elles ne sont pas requises par la Charte.

A.6.6 dpkg-dev-el

`dpkg-dev-el` fournit des macros Emacs Lisp pour faciliter l'édition des fichiers du répertoire `debian`. Par exemple, des fonctions pratiques permettent de lister les bogues actuels d'un paquet et de finaliser la dernière entrée d'un fichier `debian/changelog`.

A.6.7 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 quinn-diff

quinn-diff est utilisé pour localiser les différences d'une architecture à l'autre. Il permet de déterminer, par exemple, quels paquets de l'architecture *X* doivent être portés sur l'architecture *Y*.

A.7.2 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 (doc book-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

Contient les clés publiques GPG et PGP des développeurs Debian. Voir Section 3.2.2 et la documentation du paquet pour plus d'informations.

A.8.4 debian-maintainers

Contient les clés publiques GPG des responsables Debian. Voir http://wiki.debian.org/DebianMaintainer et la documentation du paquet pour plus d'informations.

A.8.5 debview

debview fournit un mode Emacs pour voir les paquets binaires Debian. Il vous permet d'examiner un paquet sans le décompresser.