Référence du développeur Debian

L'équipe de la Référence du développeur
<developers-reference@packages.debian.org>
Andreas Barth
Adam Di Carlo
Raphaël Hertzog
Christian Schwarz
Ian Jackson

version française par Frédéric Bothamy (traducteur actuel)
et Antoine Hulin (ancien traducteur)
et les membres de la liste <debian-l10n-french@lists.debian.org>

Version 3.3.8, 11 novembre 2006 (version française 20061130).

Copyright

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 ('Devenir responsable Debian' page 5), comment créer de nouveaux paquets ('Nouveaux paquets' page 35) et comment les installer dans l'archive ('Mettre à jour un paquet' page 39), comment gérer les rapports de bogues ('Gérer les bogues' page 42), comment déplacer, effacer ou abandonner un paquet ('Déplacer, effacer, changer le nom, adopter et abandonner des paquets' page 50), comment faire le portage d'un paquet ('Le portage' page 53), quand et comment faire la mise à jour du paquet d'un autre responsable ('Mise à jour indépendante' page 59).

Les ressources présentées dans ce manuel incluent les listes de diffusion ('Les listes de diffusion' page 13) et les serveurs ('Les serveurs Debian' page 16), une présentation de la structure de l'archive Debian ('L'archive Debian' page 19), des explications sur les serveurs qui acceptent les mises à jour de paquets ('Installer un paquet sur `ftp-master`' page 39) et une présentation des outils qui peuvent aider un responsable à améliorer la qualité de ses paquets ('Aperçu des outils du responsable Debian' page 107).

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. Cette information est disponible dans la charte Debian (`http://www.debian.org/doc/debian-policy/`).

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.

1.1 Introduction à la version française

1.1.1 Avertissement

Bien que ce document soit en français, l'activité de responsable Debian implique une maîtrise de la langue anglaise. Le projet Debian est un projet international auquel participent des japonais, néo-zélandais, américains, allemands, finlandais, français, espagnols, danois, etc.

En conséquence, la langue utilisée dans toutes les listes de diffusion destinées aux développeurs (à l'exception de la liste `<debian-devel-french@lists.debian.org>`) est l'anglais et les rapports de bogue doivent être rédigés en anglais. En fait, sauf exception rare, tout dialogue avec les autres responsables Debian se fera en anglais quel que soit le média.

1.1.2 Glossaire

Cette section liste les termes techniques qui ont un sens spécifique dans le projet Debian. Pour chacune de ces expressions, vous trouverez la traduction française utilisée dans ce manuel, une définition et une référence à la section du manuel qui traite de ce sujet.

- *Upload* : mise à jour, téléchargement (parfois livraison). Opération qui consiste à télécharger un nouveau paquet ou une nouvelle version de paquet dans l'archive Debian ('Mettre à jour un paquet' page 39).
- *Non-maintainer upload (NMU)* : mise à jour indépendante. Téléchargement d'une nouvelle version de paquet dans l'archive Debian par une personne qui n'est pas officiellement responsable de ce paquet ('Mise à jour indépendante' page 59).
- *Source NMU* : mise à jour indépendante source. Il s'agit d'une mise à jour indépendante pour un paquet source ('Terminologie' page 63).
- *Binary NMU* : mise à jour indépendante binaire. Mise à jour indépendante pour un paquet binaire ('Terminologie' page 63).
- *Bug Tracking System (BTS)* : système de suivi des bogues. Il s'agit du système utilisé par le projet Debian pour suivre les bogues et leur correction ('Gérer les bogues' page 42).
- *Bug submitter* : rapporteur du bogue. Personne qui envoie un rapport de bogue au système de suivi des bogues ('Rapporter des bogues' page 95).
- *Release critical bug* : bogue empêchant l'intégration du paquet dans la distribution. Bogues de gravité *critique*, *grave* et *sérieuse*. Ces bogues ne doivent pas apparaître dans une distribution *stable*. Ils doivent être corrigés ou bien les paquets en cause doivent être supprimés ('Comment gérer les bogues empêchant l'intégration du paquet dans la distribution ?' page 11).
- *Package Tracking System (PTS)* : système de suivi des paquets. Il s'agit du système utilisé par le projet Debian pour suivre les paquets sources des différentes distributions ('Système de suivi des paquets' page 28).
- *Unstable* : nom de la distribution en cours de développement. Cette distribution contient les paquets envoyés par les développeurs. Ceux-ci n'étant que des êtres humains, elle est parfois cassée ('*Stable*, *testing* et *unstable*' page 23).
- *Testing* : nom de la distribution en test. Cette distribution reçoit les paquets des développeurs qui ont passé une période de deux semaines dans *unstable* et pour lesquels aucun bogue remettant en cause la distribution (cf. *Release critical bug*) n'a été découvert. Cette distribution

n'a pas été testée en profondeur. Elle est cependant censée être plus stable qu'*unstable* (‘*Stable*, *testing* et *unstable*’ page 23).
– *Stable* : nom de la distribution stable. Cette distribution a été testée, validée et diffusée (‘*Stable*, *testing* et *unstable*’ page 23).
– *Debian maintainer* : responsable Debian, développeur Debian (parfois mainteneur). Personne qui fait officiellement partie du projet Debian. Elle a accès aux serveurs Debian et participe au développement — au sens large — de la distribution (‘Les charges du responsable Debian’ page 9). La plupart des responsables font de la mise en paquet, mais il existe d'autres activités telles que la documentation, la gestion du site web, la création des cédéroms, l'administration des serveurs, etc.
– *Upstream version* : version amont. Il s'agit du logiciel tel qu'il est fourni par le responsable amont — par opposition à la version fournie par la distribution Debian. (Voir ‘Coordination avec les développeurs amont’ page 11.)
– *Upstream maintainer* : responsable amont ou développeur amont. Personne responsable du développement ou de la maintenance d'un logiciel. En général, le responsable amont n'a rien à voir avec le projet Debian (‘Coordination avec les développeurs amont’ page 11).
– *Debian keyring* : porte-clés Debian. Le porte-clés Debian contient les clés publiques de tous les responsables Debian (‘Gérer votre clé publique’ page 9).
– *Work-needing and prospective packages (WNPP)* : paquets en souffrance et paquets souhaités. La liste des paquets en souffrance indique les paquets qui n'ont plus de responsable. La liste des paquets souhaités indique les logiciels que des utilisateurs aimeraient bien voir dans la distribution (‘Mettre à jour un paquet’ page 39).

Chapitre 2

Devenir responsable Debian

2.1 Pour commencer

Vous avez lu toute la documentation, vous avez examiné le guide du nouveau responsable (http://www.debian.org/doc/maint-guide/), vous comprenez l'intérêt de tout ce qui se trouve dans le paquet d'exemple `hello` et vous vous apprêtez à mettre en paquet 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 l'adresse <debian-devel-REQUEST@lists.debian.org> avec le mot `subscribe` dans la ligne *Objet*[1] de votre message. En cas de problème, contactez l'administrateur de la liste <listmaster@lists.debian.org>. Vous trouverez plus d'informations dans la section 'Les listes de diffusion' page 13. <debian-devel-announce@lists.debian.org> est une autre liste incontournable pour qui veut suivre les développements de Debian.

Vous suivrez 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 la section 'Mentors et parrains Debian' page suivante pour les détails. Le canal IRC `#debian` pourra aussi être utile ; voir 'Canaux IRC' page 15.

Quand vous avez choisi la manière dont vous contribuerez au projet Debian GNU/Linux, prenez contact avec les responsables Debian qui travaillent sur des tâches similaires. Ainsi, vous pourrez apprendre auprès de personnes expérimentées. Si, par exemple, vous voulez mettre en paquet des logiciels existants, trouvez-vous un parrain. Un parrain est une personne qui travaillera sur vos paquets avec vous et les téléchargera dans l'archive Debian une fois qu'il sera satisfait de votre mise en paquet. 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 'Parrainer des paquets' page 99 et http://people.debian.org/

[1] *Subject* en anglais

~mpalmer/debian-mentors_FAQ.html 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), vous pouvez contacter les responsables qui travaillent déjà sur ces tâches et proposer des correctifs et des améliorations.

Un écueil à éviter est d'avoir la partie locale de votre adresse électronique trop générique : des termes comme mail, admin, root ou master 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 'Les listes de diffusion' page 13 pour les détails).

Ceux qui préfèrent recevoir une aide plus personnalisée (par exemple, par courriels privés) 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 non officielle de debian-mentors à http://people.debian.org/~mpalmer/debian-mentors_FAQ.html.

Si vous désirez être un mentor et/ou un parrain, vous pouvez trouver plus d'informations à 'Interagir avec de futurs développeurs Debian' page 99.

2.3 S'enregistrer comme responsable Debian

Avant de décider de devenir responsable Debian, il vous faudra lire toute la documentation disponible dans le coin du nouveau responsable (http://www.debian.org/devel/join/newmaint). 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 (http://www.debian.org/social_contract). 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 (http://www.gnu.org/gnu/manifesto.html) 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 900 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 (`http://nm.debian.org/gpg.php`) devrait vous 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. Veuillez vous reporter à la page d'identification (`http://www.debian.org/devel/join/nm-step2`) 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 'Gérer votre clé publique' page 9 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 (`http://www.rfc-editor.org/rfc/rfc2440.txt`).

Vous avez besoin d'une clé en version 4 à utiliser pour le développement Debian. La longueur de votre clé doit être d'au moins 1024 ;bits ; il n'y a pas de raison d'utiliser une clé plus petite et faire cela serait bien moins sûr. [2]

[2] Les clés en version 4 sont des clés 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 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 « quel type de clé voulez-vous : (1) DSA et Elgamal, (2) DSA (signature seule), (5) RSA (signature seule). Si vous n'avez pas des besoins spécifiques, choisissez simplement la valeur par défaut ». Le moyen le plus simple de dire si une clé existante est une clé v4 ou une clé v3 (ou v2) est de regarder son empreinte : les empreintes des clés en version 4 sont des hachés SHA-1 d'une partie de la clé, il s'agit donc d'une suite de 40 chiffres hexadécimaux, habituellement groupés par blocs de 4. 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 ceci : `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 ». Notez également que votre clé doit être auto-signée (c.-à-d. elle doit signer tous ses

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 (`http://www.debian.org/devel/join/nm-step2`). Cette documentation explique comment mettre votre clé publique sur un serveur. L'équipe *New maintainer* mettra 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 *avocat*). 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.

Quand vous avez trouvé un *avocat*, quand votre clé GnuPG est signée et quand vous avez déjà contribué au projet, vous êtes prêt à faire acte de candidature. Il vous suffit pour cela de vous enregistrer sur notre page de candidature (`http://nm.debian.org/newnm.php`). Ensuite, votre avocat devra confirmer votre candidature. Quand il aura accompli cette tâche, un responsable de candidature[3] sera désigné pour vous accompagner dans le processus d'enregistrement. Vous pouvez toujours consulter le tableau de bord des candidatures (`http://nm.debian.org/`) pour connaître l'état de votre candidature.

Pour en savoir plus, consultez le coin des nouveaux responsables (`http://www.debian.org/devel/join/newmaint`) 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.

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.

[3] Application manager

Chapitre 3

Les charges du responsable Debian

3.1 Mise à jour de vos références Debian

Il existe une base de données LDAP contenant des informations concernant les développeurs Debian à 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 redirigée 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 à propos de cette base de données, veuillez consulter 'La base de données des développeurs' page 18.

3.2 Gérer votre 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 'Les serveurs Debian' page 16). Sauvegardez vos clés et gardez-en une copie hors connexion. Lisez la documentation fournie avec votre logiciel et la FAQ PGP (http://www.cam.ac.uk.pgp.net/pgpnet/pgp-faq/).

Vous devez vous assurer que non seulement votre clé est à l'abri des vols, mais également à l'abri d'une perte. Générez et faites une copie (et également sur papier) de votre certificat de révocation ; il est nécessaire si votre clé est perdue.

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.

Si vous voulez 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 la certification de révocation. S'il n'y pas de vraie raison pour une nouvelle clé, les responsables du trousseau de clés peuvent rejeter la nouvelle clé. Vous pouvez trouver plus de détails à http://keyring.debian.org/replacing_keys.html.

Les mêmes routines d'extraction de clé décrites dans 'S'enregistrer comme responsable Debian' page 6 s'appliquent.

Vous pouvez trouver une présentation approfondie de la gestion de clé Debian dans la documentation du paquet `debian-keyring`.

3.3 Voter

Bien que Debian ne soit pas vraiment une démocratie, nous disposons d'un processus démocratique pour élire nos chefs et pour approuver les résolutions générales. Ces procédures sont définies par la charte Debian (`http://www.debian.org/devel/constitution`).

En dehors de l'élection annuelle du chef, les votes ne se tiennent pas régulièrement et ils 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 elle 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 à la page Informations sur les votes Debian (`http://www.debian.org/vote/`), ainsi que des informations complémentaires sur la procédure à suivre pour effectuer une proposition, la soutenir et voter pour elle.

3.4 Prendre des vacances courtoisement

Il est courant pour les développeurs de s'absenter, que ce soit pour des vacances prévues ou parce qu'ils sont submergés de travail. La chose importante à noter est que les autres développeurs ont besoin de savoir que vous êtes en vacances pour qu'ils puissent agir en conséquence si un problème se produit pendant vos vacances.

Habituellement, cela veut dire que les autres développeurs peuvent faire des NMU (voir 'Mise à jour indépendante' page 59) 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.

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

L'autre chose à faire est de vous signaler comme « en vacances »[2] dans la base de données Debian LDAP (l'accès à cette information est réservé aux développeurs). N'oubliez pas de retirer votre indicateur « en vacances » lorsque celles-ci sont terminées !

Idéalement, vous devriez vous connecter sur le site de coordination GPG (`http://nm.debian.org/gpg.php`) quand vous réservez pour des vacances 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.

3.5 Coordination avec les développeurs amont

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

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, alors 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.

3.6 Comment gérer les bogues empêchant l'intégration du paquet dans la distribution ?

Habituellement, vous devriez traiter les rapports de bogue sur vos paquets tel que cela est décrit dans 'Gérer les bogues' page 42. Cependant, il y a une catégorie spéciale de bogues qui nécessite particulièrement votre attention : les bogues empêchant l'intégration du paquet dans la distribution[3]. Tous les rapports de bogue de gravité *critique*, *grave* et *sérieuse*[4] sont considérés comme ayant un impact sur la présence du paquet dans la prochaine version stable de Debian. Ces bogues peuvent retarder la publication d'une distribution Debian ou peuvent justifier la suppression d'un paquet d'une distribution gelée. C'est pourquoi ces bogues doivent être corrigés au plus vite.

[2] *on vacation* en anglais
[3] Traduction de l'anglais *Release-critical bug (RC Bugs)*
[4] Respectivement *critical*, *grave* et *serious* en anglais

Les développeurs faisant partie de l'équipe d'assurance qualité Debian (http://qa.debian.org/) surveillent ces bogues et essaient de vous aider chaque fois qu'ils le peuvent. Si vous ne pouvez pas corriger un bogue de ce type dans les deux semaines, vous devriez soit demander de l'aide en envoyant un courrier à l'équipe d'assurance qualité (*QA group*) à <debian-qa@lists.debian.org>, soit expliquer vos difficultés et présenter un plan pour corriger le problème en répondant au rapport de bogue concerné. Si rien n'est fait, des personnes de l'équipe d'assurance qualité pourraient chercher à corriger votre paquet (voir 'Mise à jour indépendante' page 59) après avoir tenté de vous contacter (ils pourraient attendre moins longtemps que d'habitude pour faire leur mise à jour s'il n'y a pas trace d'activité récente de votre part dans le système de suivi des bogues).

3.7 Démissionner

Si vous choisissez de quitter le projet Debian, procédez comme suit :

1. abandonnez tous vos paquets comme décrit dans 'Abandonner un paquet' page 52 ;
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 écrivant à <keyring-maint@debian.org>.

Chapitre 4

Ressources pour le responsable Debian

Dans ce chapitre, vous trouverez une brève description des listes de diffusion, des machines Debian qui seront à votre disposition en tant que responsable Debian ainsi que toutes les autres ressources à votre disposition pour vous aider dans votre travail de responsable.

4.1 Les listes de diffusion

Une grande partie des discussions entre les développeurs Debian (et les utilisateurs) est gérée par un vaste éventail de listes de diffusion que nous hébergeons à `lists.debian.org` (`http://lists.debian.org/`). Pour en savoir plus sur comment s'abonner ou se désabonner, comment envoyer un message et comment ne pas en envoyer, comment retrouver d'anciens messages et comment les rechercher, comment contacter les responsables des listes et pour savoir d'autres informations sur les listes de diffusion, veuillez lire `http://www.debian.org/MailingLists/`. Cette section ne couvrira que les aspects des listes de diffusion qui ont un intérêt particulier pour les développeurs.

4.1.1 Règles de base d'utilisation

Lorsque vous répondez sur une liste de diffusion, veuillez ne pas envoyer de copie (`CC`) à l'auteur d'origine sauf s'il le demande explicitement. Toute personne envoyant un message à une liste de diffusion devrait la suivre pour voir les réponses.

Le multi-postage (cross-posting) (envoyer le même message à plusieurs listes) est découragé. Comme toujours sur le net, veuillez réduire la citation des articles auxquels vous répondez. En général, veuillez adhérez aux conventions usuelles d'envoi de messages.

Veuillez lire le code de conduite (`http://www.debian.org/MailingLists/#codeofconduct`) pour plus d'informations.

4.1.2 Principales listes pour les responsables

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

– <debian-devel-announce@lists.debian.org>, utilisée pour les annonces importantes faites aux responsables. Tous les responsables Debian sont censés être inscrits à cette liste,
– <debian-devel@lists.debian.org>, utilisée pour discuter de diverses questions techniques relatives au développement,
– <debian-policy@lists.debian.org> où l'on discute et vote les modifications de la charte Debian,
– <debian-project@lists.debian.org>, utilisée pour discuter de questions non techniques.

Il existe d'autres listes de diffusion spécialisées dans différents thèmes. Reportez-vous à la page http://lists.debian.org/ pour en obtenir la liste complète.

4.1.3 Listes spéciales

<debian-private@lists.debian.org> est une liste de diffusion destinée aux discussions privées entre développeurs Debian. Elle doit être utilisée pour tout message qui ne doit pas être publié, quelle qu'en soit la raison. C'est une liste à faible trafic et les utilisateurs sont priés de ne l'utiliser qu'en cas de réelle nécessité. De plus, il ne faut *jamais* faire suivre un courrier de cette liste à qui que ce soit. Pour des raisons évidentes, les archives de cette liste ne sont pas disponibles sur la toile. Vous pouvez les consulter en visitant le répertoire ~debian/archive/debian-private avec votre compte sur lists.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 relative au développement

Avant de demander une liste de diffusion liée au développement d'un paquet (ou d'un petit groupe de paquets liés), veuillez considérer si l'utilisation d'un alias (via un fichier .forward-aliasname sur master.debian.org, ce qui se traduit en une adresse raisonnablement agréable *you-aliasname@debian.org*) ou une liste de diffusion auto-gérée sur Alioth serait plus appropriée.

Si vous décidez qu'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 (http://www.debian.org/MailingLists/HOWTO_start_list).

4.2 Canaux IRC

Plusieurs canaux IRC sont dédiés au développement Debian. Ils sont principalement hébergés sur le réseau Open and free technology community (OFTC) (http://www.oftc.net/oftc/). L'entrée DNS irc.debian.org est simplement 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 Debian est *#debian-devel*. C'est un canal très actif avec habituellement plus de 150 personnes 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 toujours rempli d'informations intéressantes.

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 canal protégé par clé *#debian-private* dans ce but. La clé est disponible dans les archives de debian-private à master.debian.org:~debian/archive/debian-private/, effectuez simplement un zgrep avec *#debian-private* dans tous les fichiers.

Il existe d'autres canaux dédiés à des sujets spécifiques. *#debian-bugs* est utilisé pour la coordination des *chasses aux bogues*. *#debian-boot* est utilisé pour la coordination du travail sur l'installateur Debian. *#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-bsd*, *#debian-kde*, *#debian-jr*, *#debian-edu*, *#debian-sf* (paquet SourceForge), *#debian-oo* (paquet OpenOffice), etc.

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

Il existe également des canaux dédiés pour Debian sur d'autres réseaux IRC, notamment sur le réseau IRC Freenode (http://www.freenode.net/), sur lequel pointait l'alias irc.debian.org jusqu'au 4 juin 2006.

Pour obtenir un uniforme (« cloak ») sur freenode, vous devez envoyer un courriel signé à Jörg Jaspert <joerg@debian.org> dans lequel vous indiquez quel est votre pseudonyme (« nick »). Mettez « cloak » quelque part dans le sujet. Votre pseudonyme devra être enregistré : Page de configuration des pseudonymes (http://freenode.net/faq.shtml#nicksetup). Le courriel doit être signé par une clé du porte-clés Debian. Veuillez consulter la documentation de Freenode (http://freenode.net/faq.shtml#projectcloak) pour plus d'informations sur les uniformes.

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 (http://www.debian.org/devel/). Prenez le temps de parcourir tous les liens, vous apprendrez encore beaucoup de choses.

4.4 Les 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 (http://www.debian.org/devel/dmup).

Dans l'ensemble, vous pouvez faire usage de ces machines pour des buts relatifs à Debian comme vous l'entendez. Veuillez cependant être gentil avec les administrateurs système et ne pas utiliser de grandes quantités d'espace disque, de ressource réseau ou CPU sans obtenir auparavant l'accord des administrateurs. Habituellement, ces machines sont administrées par des volontaires.

Veuillez prendre soin de votre mot 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 l'Internet comme telnet, FTP, POP, etc.

Veuillez ne pas déposer de données non relatives à Debian sur les serveurs Debian à moins que vous n'ayez préalablement obtenu la permission de le faire.

La liste actuelle des machines Debian est disponible à http://db.debian.org/machines.cgi. Cette page web contient les noms des machines, les informations de contact, les informations sur qui peut s'y connecter, les clés SSH, etc.

Si vous avez un problème en utilisant un serveur Debian et si vous estimez que les administrateurs système devraient en être avertis, l'équipe des administrateurs système peut être jointe à <debian-admin@lists.debian.org>.

Si votre problème est lié à un certain service ou n'est pas lié au système (paquet à supprimer de l'archive ou suggestion pour le site web par exemple), il vous faudra en général ouvrir un rapport de bogue sur un « pseudo-paquet ». Reportez-vous à la section 'Rapporter des bogues' page 95 pour connaître la procédure à suivre.

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

4.4.1 Le serveur pour les rapports de bogues

`bugs.debian.org` est le serveur maître du système de suivi des bogues (BTS[1]).

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

Si vous envisagez de manipuler les rapports de bogue 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 Le serveur ftp-master

Le serveur `ftp-master.debian.org` est le serveur maître de l'archive Debian (exception faite des paquets non-US). En général, les mises à jour de paquets se font sur ce serveur. Reportez-vous à la section 'Mettre à jour un paquet' page 39 pour en savoir plus.

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

Les problèmes avec l'archive Debian FTP doivent généralement être rapportés comme bogues sur le pseudo-paquet `ftp.debian.org` ou par courrier électronique à `<ftpmaster@debian.org>` ; reportez-vous à la section 'Déplacer, effacer, changer le nom, adopter et abandonner des paquets' page 50 pour connaître la procédure à suivre.

4.4.3 Le serveur non-US

Le serveur non-US `non-us.debian.org` a été interrompu avec la publication de *Sarge*. Le pseudo-paquet `nonus.debian.org` existe encore pour le moment.

4.4.4 Le serveur 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 devez généralement envoyer un rapport de bogue sur le pseudo-paquet `www.debian.org`. Vérifiez d'abord sur le système de suivi des bogues (`http://bugs.debian.org/www.debian.org`) que personne ne l'a déjà rapporté avant vous.

4.4.5 Le serveur web people

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

[1] Système de suivi des bogues se dit *Bug Tracking System* en anglais

Si vous avez des informations spécifiques 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-user-id/`.

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

Habituellement, la seule raison pour utiliser un serveur différent est que vous avez besoin de publier des informations soumises aux restrictions d'exportation américaines, dans ce cas, vous pouvez utiliser l'un des autres serveurs situés en dehors des États-Unis.

Veuillez envoyer un courrier à `<debian-devel@lists.debian.org>` si vous avez une question.

4.4.6 Le serveur CVS

Notre serveur CVS est situé sur `cvs.debian.org`.

Si vous avez besoin d'un serveur CVS accessible par tous pour, par exemple, coordonner le travail de plusieurs développeurs sur un paquet, vous pouvez demander un espace CVS sur ce serveur.

Le serveur `cvs.debian.org` autorise les accès CVS locaux, les accès en lecture seule pour les connexions client-serveur anonymes et les accès client-serveur complets pour les connexions `ssh`. L'espace CVS peut aussi être consulté par la toile à l'adresse `http://cvs.debian.org/`.

Pour obtenir un espace CVS, envoyez une demande à l'adresse `<debian-admin@debian.org>` en précisant le nom de l'espace, le compte Debian propriétaire du répertoire racine et pourquoi vous en avez besoin.

4.4.7 Chroots de différentes distributions

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

```
vore% dchroot unstable
Executing shell in chroot: /org/vore.debian.org/chroots/user/unstable
```

Dans tous les chroots, les répertoires normaux des utilisateurs sont disponibles. Vous pouvez trouver quels chroots sont disponibles à `http://db.debian.org/machines.cgi`.

4.5 La base de données des développeurs

La base de données des développeurs à `https://db.debian.org/` est un annuaire LDAP de gestion des informations des développeurs Debian. Vous pouvez utiliser cette ressource

pour rechercher la liste des développeurs Debian. Une partie de ces informations est également disponible à travers le service `finger` sur les serveurs Debian, essayez `finger yourlogin@debian.org` pour voir ce qu'il indique.

Les développeurs peuvent se connecter à la base de données (https://db.debian.org/login.html) 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 (http://www.debian.org/devel/developers.loc), 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 accessibles publiquement. Pour plus d'informations, veuillez lire la documentation en ligne que vous pouvez trouver à http://db.debian.org/doc-general.html.

Les développeurs peuvent également envoyer leurs clés SSH pour qu'elles soient utilisées pour autorisation sur les machines Debian officielles et même ajouter de nouvelles entrées DNS du type *.debian.net. Ces fonctionnalités sont documentées à http://db.debian.org/doc-mail.html.

4.6 L'archive Debian

La distribution Debian GNU/Linux est composée d'un grand nombre de paquets (fichiers `.deb` : actuellement, à peu près 9000) et de quelques autres fichiers (comme la documentation et des images des disquettes d'installation).

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

```
dists/stable/main/
dists/stable/main/binary-i386/
dists/stable/main/binary-m68k/
dists/stable/main/binary-alpha/
      ...
dists/stable/main/source/
      ...
dists/stable/main/disks-i386/
dists/stable/main/disks-m68k/
dists/stable/main/disks-alpha/
      ...

dists/stable/contrib/
dists/stable/contrib/binary-i386/
```

```
dists/stable/contrib/binary-m68k/
dists/stable/contrib/binary-alpha/
     ...
dists/stable/contrib/source/

dists/stable/non-free/
dists/stable/non-free/binary-i386/
dists/stable/non-free/binary-m68k/
dists/stable/non-free/binary-alpha/
     ...
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/n/
pool/non-free/n/netscape/
     ...
```

Comme vous pouvez le voir, le répertoire racine contient deux répertoires, `dists/` et `pool/`.

Le second est un ensemble de répertoires où sont stockés les paquets. Ceux-ci sont manipulés grâce à la base de données de l'archive et aux logiciels qui l'accompagnent. Le premier répertoire contient les distributions *stable*, *testing* et *unstable*. Les fichiers `Packages` et `Sources` qui se trouvent dans les répertoires de distribution peuvent faire référence à des fichiers du répertoire `pool/`. Le découpage en sous-répertoires est identique d'un répertoire de distribution à l'autre . Ce que nous exposerons ci-dessous pour la distribution *stable* est également applicable aux 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 sources (`source/`) et un répertoire pour chaque architecture acceptée (`binary-i386/`, `binary-m68k/`, etc.).

La section *main* contient d'autres répertoires destinés aux images de disquettes et à plusieurs documents essentiels pour installer la distribution Debian sur une architecture particulière (`disk-i386/`, `disk-m68k/`, etc.).

4.6.1 Les sections

La section *main* constitue la **distribution Debian GNU/Linux officielle**. Elle 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 (`http://www.debian.org/social_contract#guidelines`)[2] et à toutes les autres recommandations décrites dans la charte Debian (`http://www.debian.org/doc/debian-policy/`)[3]. Les DFSG[4] constituent notre définition de « logiciel libre ». 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 rangés dans la section *non-free*. Bien que nous supportions l'usage de ces paquets et qu'ils bénéficient de nos infrastructures (système de suivi des bogues, listes de diffusion, etc.), ces paquets *non-free* ne font pas partie de la distribution Debian.

La *charte Debian* donne des définitions plus précises pour 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 serveur FTP ou sur cédérom. 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.

[2] Debian Free Software Guidelines
[3] Debian Policy Manual
[4] *Debian Free Software Guidelines*

D'un autre côté, un distributeur de cédérom pourra facilement vérifier la licence de chacun des paquets de la section *non-free* et inclure tous les paquets qu'il lui sera autorisé (dans la mesure où cela varie énormément d'un distributeur à l'autre, ce travail ne peut être fait par les développeurs Debian).

Notez que le terme « section » est également utilisé pour faire référence aux catégories qui simplifient l'organisation des paquets disponibles et leur recherche, e.g. *admin*, *net*, *utils* etc. Il fut un temps où ces sections (sous-sections, plutôt) existaient sous la forme de sous-répertoires dans l'archive Debian. Actuellement, elles n'existent plus que dans le champ en-tête « Section » des paquets.

4.6.2 Les architectures

À ses débuts, le noyau Linux existait uniquement pour les architectures Intel x86 ; il en était de même pour Debian. Linux devenant de plus en plus populaire, il a été porté vers d'autres architectures.

Le noyau 2.0 existe pour les architectures Intel x86, DEC Alpha, SPARC, Motorola, 680x0 (Atari, Amiga, et Macintosh), MIPS et PowerPC. Le noyau 2.2 reconnaît de nouvelles architectures, comme ARM et UltraSPARC. Puisque Linux reconnaît ces architectures, le projet Debian a décidé qu'il devait également les accepter. C'est pourquoi plusieurs portages sont en cours ; en fait, il y a aussi des portages vers d'autres noyaux non-Linux. À côté d'*i386* (notre nom pour Intel x86), nous avons, au moment où j'écris, *m68k*, *alpha*, *powerpc*, *sparc*, *hurd-i386*, *arm*, *ia64*, *hppa*, *s390*, *mips*, *mipsel* et *sh*.

Debian GNU/Linux 1.3 est disponible uniquement pour *i386*. Debian 2.0 reconnaît les architectures *i386* et *m68k*. Debian 2.1 reconnaît les architectures *i386*, *m68k*, *alpha* et *sparc*. Debian 2.2 accepte en plus les architectures *powerpc* et *arm*. Debian 3.0 a ajouté cinq nouvelles architectures : *ia64*, *hppa*, *s390*, *mips* et *mipsel*.

Pour chaque portage, vous trouverez des informations destinées aux développeurs et utilisateurs sur la page Portage Debian (`http://www.debian.org/ports/`).

4.6.3 Les paquets

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

Les paquets sources sont constitués de deux ou trois fichiers : un fichier `.dsc` et soit un fichier `.tar.gz`, soit un fichier `.orig.tar.gz` et un fichier `.diff.gz`.

Si un paquet est développé spécifiquement pour le projet Debian et n'est pas distribué en dehors de Debian, il n'y a qu'un fichier `.tar.gz` qui contient les sources du programme. Si un paquet est distribué ailleurs, le fichier `.orig.tar.gz` contient ce que l'on appelle *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` 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 Les distributions

L'organisation des répertoires présentée précédemment est elle-même englobée par les *répertoires des distributions*. Chaque distribution est 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.us.debian.org`, l'archive Debian se trouve dans `/debian` ce qui est un emplacement courant. Un autre emplacement courant est `/pub/debian`.

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

Stable, *testing* et *unstable*

Il y a 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*[5] (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 garantissent que les paquets de *testing* sont de bonne qualité. La mise à jour de *testing* est lancée chaque jour après que les nouveaux paquets ont été installés. Voir 'La distribution *testing*' page 65.

Après une période de développement, quand le responsable de distribution[6] 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. Après la résolution des problèmes restants à la satisfaction de l'équipe de publication, la distribution est publiée. La publication veut dire 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`.

[5] *unstable* signifie « instable »
[6] *Release manager*

Ce cycle de développement est basé sur l'idée que la distribution *unstable* devient *stable* après une période de test (*testing*). Une distribution contient inévitablement des bogues, même si elle est classée stable. C'est pourquoi les distributions stables sont mises 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 des paquets proposés pour les mises à jour de *stable* dans le répertoire `proposed-updates`. De temps en temps, les paquets de ce répertoire qui ne présentent pas de problème sont installés dans la distribution *stable* et le numéro de révision de cette distribution est incrémenté (« 3.0 » devient « 3.0r1 », « 2.2r4 » devient « 2.2r5 » et ainsi de suite). Veuillez vous référer aux envois dans la distribution *stable* pour plus de détails.

Notez que, pendant la période de gel, les développements continuent sur la distribution *unstable* car cette distribution reste en place.

Informations complémentaires sur la distribution « testing »

Les paquets sont habituellement installés dans la distribution *testing* après avoir atteint un certain degré de test dans *unstable*.

Pour plus de détails, veuillez consulter les informations à propos de la distribution *testing*.

Experimental

La distribution *experimental* est une distribution particulière. Ce n'est pas une distribution à part entière comme le sont *stable* et *unstable*. Elle est prévue pour servir de plate-forme de développement pour les projets expérimentaux qui risquent vraiment de détruire le système ou bien pour des logiciels qui sont 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 depuis *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 compacté expérimental, par exemple, devrait probablement aller dans *experimental*.

Une nouvelle version amont qui ajoute de nouvelles fonctions tout en supprimant de nombreuses autres ne devra pas être téléchargée dans l'archive Debian, elle pourra cependant être téléchargée dans *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 jour 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 ftpmasters, les paquets expérimentaux sont automatiquement 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`.

Veuillez considérer l'utilisation de l'option `-v` de `dpkg-buildpackage` si l'envoi d'un paquet dans *unstable* ferme finalement des bogues qui ont d'abord été corrigés dans *experimental*. Lors de l'envoi vers *unstable* d'un paquet contenant des bogues corrigés dans *experimental*, veuillez considérer l'utilisation de l'option `-v` de `dpkg-buildpackage` pour qu'ils soient finalement fermés.

4.6.5 Les noms de distribution

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 ». Il y a aussi une pseudo-distribution nommée « Sid », il s'agit de la distribution *unstable* ; comme les paquets vont d'*unstable* à *testing* quand ils approchent de la stabilité, la distribution « Sid » n'est jamais diffusée. 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 (i.e. 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 prochaine distribution à diffuser « testing », il aurait fallu changer son nom en « stable » au moment de la publication, ce qui aurait forcé les miroirs FTP à télécharger à 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 distribution Debian *x.y* est disponible. (Cela s'est produit par le passé, un distributeur avait gravé des cédéroms 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 des distributions dans l'archive sont déterminés par leur nom de code et non par leur statut (exemple : slink). 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 Les miroirs Debian

Les différentes archives de téléchargement et le site web disposent de plusieurs miroirs pour soulager les serveurs principaux d'une lourde charge. En fait, certains de ces serveurs 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 plusieurs réseaux différents et évite aux utilisateurs de surcharger l'emplacement primaire. Notez que, 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 à `http://www.debian.org/mirror/`, y compris une liste des miroirs publics disponibles FTP/HTTP. Cette page utile inclut également des informations et des outils pour créer son propre miroir, soit en interne soit pour un accès public.

Les miroirs sont en général 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 Le système Incoming

Le système Incoming est responsable de la collecte des paquets mis à jour et de leur installation dans l'archive Debian. Il est constitué d'un ensemble de répertoires et de scripts qui sont installés 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 de la plupart des développeurs car ftp-master est à accès restreint ; il est parcouru toutes les 15 minutes par le script `katie` qui vérifie l'intégrité des paquets envoyés et leurs signatures de chiffrage. Si le paquet est considéré comme prêt à être installé, il est déplacé dans le répertoire `accepted`. S'il s'agit du premier envoi du paquet (ou s'il a de nouveaux paquets binaire), il est déplacé dans le répertoire `new` où il attend l'approbation des ftpmasters. Si le paquet contient des fichiers devant être installés *à la main*, il est déplacé dans le répertoire `byhand` où il attend une installation manuelle par les ftpmasters. Sinon, si une erreur a été détectée, le paquet est refusé et il est déplacé dans le répertoire `reject`.

Une fois que le paquet est accepté, le système envoie une confirmation par courrier au responsable et ferme les bogues corrigés par l'envoi, puis les compilateurs automatiques peuvent commencer la recompilation. Le paquet est maintenant accessible publiquement à `http://incoming.debian.org/` jusqu'à ce qu'il soit vraiment installé dans l'archive Debian. Cela se produit seulement une fois par jour (et c'est également appelé une « exécution dinstall » pour des raisons historiques) ; le paquet est alors supprimé de `Incoming` et installé dans le pool avec les autres paquets. Une fois que toutes les autres mises à jour (fabrication des

nouveaux fichiers d'index `Packages` et `Sources` par exemple) ont été effectuées, un script spécial est appelé pour demander aux miroirs primaires de se mettre à jour.

Le logiciel de maintenance de l'archive enverra également le fichier `.changes` signé avec OpenPGP/GnuPG que vous avez envoyé, à la liste de diffusion appropriée. Si un paquet est diffusé avec le champ `Distribution:` positionné à « stable », l'annonce sera envoyée à `<debian-changes@lists.debian.org>`. Si un paquet est diffusé avec le champ `Distribution:` positionné à « unstable » ou « experimental », l'annonce sera à la place 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 `merkel.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-paquet`* affiche chaque version du paquet disponible dans les différentes distributions. Les informations détaillées par version incluent la description du paquet, les dépendances et des liens pour télécharger le paquet.

Le système de suivi des bogues trie les bogues par paquet. Vous pouvez regarder les bogues de chaque paquet à `http://bugs.debian.org/`*`nom-paquet`*.

4.9.2 L'outil `madison`

`madison` est un outil en ligne de commande qui est disponible sur `ftp-master.debian.org` et sur le miroir `merkel.debian.org`. Il utilise un seul paramètre qui correspond au nom du paquet. Il affiche comme résultat quelle version du paquet est disponible pour chaque combinaison d'architecture et de distribution. Un exemple l'expliquera mieux.

```
$ madison libdbd-mysql-perl
libdbd-mysql-perl |   1.2202-4  |   stable   | source, alpha, arm, i386, m68k
libdbd-mysql-perl |   1.2216-2  |   testing  | source, arm, hppa, i386, ia64,
libdbd-mysql-perl | 1.2216-2.0.1|   testing  | alpha
libdbd-mysql-perl |   1.2219-1  |   unstable | source, alpha, arm, hppa, i386
```

Dans cet exemple, vous pouvez voir que la version dans *unstable* diffère de celle de *testing* et qu'il y a eu une NMU binaire seulement pour l'architecture alpha. Chaque version du paquet a été recompilée sur la plupart des architectures.

4.10 Système de suivi des paquets

Le système de suivi des paquets (PTS)[7] est un outil pour suivre par courrier l'activité concernant un paquet source. Cela veut vraiment dire que vous pourrez recevoir les mêmes courriers que le responsable, simplement en vous inscrivant au paquet dans le PTS.

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

Par défaut, vous recevrez :

bts Tous les rapports de bogue et les discussions qui suivent.

bts-control Les courriers de notification de `<control@bugs.debian.org>` sur les changement d'état de l'un des rapports de bogue.

upload-source Le courrier de notification de `katie` quand un paquet source envoyé a été accepté.

katie-other Les autres courriers d'avertissement et d'erreur de `katie` (comme une incohérence de passage en force pour les champs section ou priorité).

default Tout courrier non automatique envoyé au PTS par les personnes qui veulent 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.

summary Des courriers de résumé réguliers sur l'état du paquet. Actuellement, seule la progression du paquet dans *testing* est envoyée.

Vous pouvez également décider de recevoir des informations supplémentaires :

upload-binary Le courrier d'information de `katie` quand un paquet binaire envoyé est accepté. En d'autres termes, à chaque fois qu'un démon de compilation ou un porteur envoie votre paquet pour une autre architecture, vous recevez un courrier pour suivre comment votre paquet est recompilé pour toutes les architectures.

cvs Les notifications de modifications CVS[8] si le responsable a mis en place un système pour faire suivre les notifications de modifications vers le PTS.

ddtp Les traductions des descriptions ou des questionnaires debconf soumis au Projet de traduction des descriptions de paquets Debian[9].

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

[7] « Package Tracking System »
[8] *CVS commits*
[9] *Debian Description Translation Project, DDTP*

4.10.1 L'interface de courrier du PTS

Vous pouvez contrôler votre (vos) inscription(s) au PTS en envoyant différentes commandes à `<pts@qa.debian.org>`.

`subscribe <paquet source> [<adresse>]` Inscrit *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 valide, vous obtiendrez un avertissement. Cependant, s'il s'agit d'un paquet binaire valide, le PTS vous inscrira pour le paquet source correspondant.

`unsubscribe <paquet source> [<adresse>]` Supprime une inscription précédente 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 précédentes 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 que vous acceptez. Pour une explication des 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 bugs.debian.org`<control@bugs.debian.org>`
- `summary` : courriers de résumé automatique sur l'état d'un paquet
- `cvs` : notifications de modifications CVS
- `ddtp` : traductions des descriptions et questionnaires debconf
- `derivatives` : changements effectués sur le paquet dans des distributions dérivées
- `upload-source` : annonce d'un nouvel envoi de paquet source qui a été accepté
- `upload-binary` : annonce d'un nouvel envoi de binaire seulement (portage)
- `katie-other` : autres courriers des ftpmasters (incohérence de passage en force, etc.)
- `default` : tous les autres courriers (ceux qui ne sont pas automatiques)

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

`keyword [<adresse>] {+|-|=} <liste de mots-clés>` Accepte (+) ou refuse (-) les courriers classés sous le(s) mot(s)-clé(s). 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 sous le(s) mot(s)-clé(s). Définit la liste (=) des mots-clés acceptés. Ceci change les mots-clés de toutes les inscriptions actuellement en cours d'un utilisateur.

`keyword <paquet source> [<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 que vous vous êtes 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 (par exemple, avec `procmail`). 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
X-Unsubscribe: echo 'unsubscribe dpkg' | mail pts@qa.debian.org
```

4.10.3 Faire suivre les modifications de CVS vers le PTS

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

Une fois que votre référentiel génère des notifications de modifications, vous devez simplement vous assurer qu'il envoie une copie de tous ces courriers à *paquet source*`_cvs@packages.qa.debian.org`. Seules les personnes qui ont accepté le mot-clé *cvs* recevront les notifications.

4.10.4 L'interface web du PTS

Le PTS possède une interface web à `http://packages.qa.debian.org/` qui réunit beaucoup d'informations au même endroit à propos de chaque paquet source. Il propose plusieurs liens utiles (BTS, statistiques QA, informations de contact, état de traduction DDTP, journaux de compilation automatique) et il regroupe beaucoup d'autres informations provenant de différents endroits (les 30 dernières entrées de changelog, l'état dans *testing*, etc.). Il s'agit d'un outil très pratique si vous désirez connaître ce qu'il en est d'un paquet source spécifique. De plus, il y a un formulaire qui permet de facilement s'inscrire au PTS par courrier.

Vous pouvez 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 développements des paquets : vous pouvez ajouter du contenu personnalisé aux pages de vos paquets. Vous pouvez ajouter des

« informations statiques »[10] (des annonces qui sont destinées à rester disponibles indéfiniment) et des nouvelles dans la section « dernières nouvelles »[11].

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 qui peut être utile à la maintenance du paquet.

Les nouvelles usuelles peuvent être utilisées pour annoncer que :

– des paquets bêta sont disponibles pour test,
– 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 vacance (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 vous suffit d'envoyer un courrier soit à `<pts-static-news@qa.debian.org>` (pour les annonces statiques), soit à `<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 dans 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 valides utilisés pour générer des nouvelles dans le PTS. Le premier ajoute un lien vers l'interface cvsweb de debian-cd dans la section « Informations statiques » :

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

Package: debian-cd
Url: http://cvs.debian.org/debian-cd/
```

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
```

[10] *Static information*
[11] *Latest news*

```
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 l'éditer. La seule chose que vous puissiez faire est d'envoyer une deuxième nouvelle qui va déprécier l'information contenue dans la précédente.

4.11 Vue d'ensemble des paquets d'un développeur

Un portail web pour l'Assurance Qualité (QA) est disponible à http://qa.debian.org/developer.php qui affiche un tableau de tous les paquets d'un développeur (y compris ceux pour lequel 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 pour ne pas oublier quels paquets sont sous votre responsabilité.

4.12 Debian *Forge : Alioth

Alioth est un service de Debian plutôt récent, basé sur une version légèrement modifiée du logiciel GForge (qui a évolué à partir de SourceForge). 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 dépôts CVS, etc. Tous ces outils sont gérés par une interface web.

Alioth est destiné à fournir des facilités pour des projets de logiciels soutenus ou dirigés par Debian, à faciliter les contributions de développeurs externes aux projets initiés par Debian et à aider des projets dont les buts sont de promouvoir Debian ou ses dérivés.

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.

Pour plus d'informations, veuillez visiter http://alioth.debian.org/.

4.13 « Goodies » pour les développeurs

4.13.1 Abonnements LWN

Depuis octobre 2002, HP parraine l'abonnement à LWN pour tous les développeurs Debian intéressés. Les détails sur les moyens d'accéder à cet avantage sont expliqués dans http://lists.debian.org/debian-devel-announce/2002/10/msg00018.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 (`http://www.debian.org/devel/wnpp/`). Vous pourrez ainsi vérifier que personne ne travaille déjà sur ce paquet et éviter un double travail. 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 'Rapporter des bogues' page 95) concernant le pseudo-paquet `wnpp`. Ce courrier devra décrire le paquet que vous projetez de créer, la licence de ce paquet et l'URL à laquelle le source peut être téléchargé. Cette liste n'est pas limitative.

Le sujet de votre rapport de bogue devra être <ITP[1] : *NomDuPaquet* — *courte description*>, 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:` car de cette manière le sujet du message ne contiendrait pas le numéro du bogue.

Il faudra aussi ajouter une entrée `Closes: bug#`*nnnnn* dans le fichier `changelog` du nouveau paquet. Cette indication fermera automatiquement le rapport de bogue à l'installation du nouveau paquet sur les serveurs d'archivage (voir 'Quand les rapports de bogue sont-ils fermés par une mise à jour ?' page 45).

Lors de la fermeture de bogues de sécurité, incluez les numéros CVS ainsi que « Closes : #nnnnn ». Ceci est utile l'équipe de sécurité pour suivre les failles de sécurité. Si un envoi est effectué pour corriger le bogue avant que l'identifiant de l'alerte soit connu, il est conseillé de modifier l'entrée de changelog historique lors du prochain envoi. Même dans ce cas, veuillez

[1] *Intent To Package* : intention d'empaquetage

inclure tous les pointeurs disponibles vers les informations de contexte dans l'entrée de changelog d'origine.

Plusieurs raisons nous poussent à demander aux responsables d'annoncer leur intention :

- Les responsables ont ainsi la possibilité de puiser dans l'expérience des autres responsables et cela leur permet de savoir si une autre personne travaille déjà dessus.
- D'autres personnes qui envisagent de travailler sur le même paquet apprendront ainsi qu'il existe déjà un volontaire, l'effort peut alors être partagé.
- Cela permet aux autres responsables de connaître le nouveau paquet mieux que ne le permettent la description d'une ligne et l'habituelle entrée de type changelog *Initial release* postée sur `debian-devel-changes`.
- C'est une information utile pour les gens qui utilisent la distribution *unstable* et qui sont nos premiers testeurs. Nous devons leur faciliter la tâche.
- Avec ces annonces, les développeurs Debian et toutes les autres personnes intéressées peuvent se faire 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 Enregistrer les modifications dans le paquet

Les modifications que vous apportez au paquet doivent être notifiées 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 si des 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 dans 'Choisir une distribution' page 38. Vous trouverez plus d'informations sur la structure de ce fichier dans la section « `debian/changelog` » de la *charte Debian*.

Les entrées du fichier `changelog` peuvent être utilisées pour fermer des rapports de bogue au moment où le paquet est installé dans l'archive. Voir la section 'Quand les rapports de bogue sont-ils fermés par une mise à jour ?' page 45.

Par convention, l'entrée changelog d'un paquet contenant une nouvelle version amont ressemble à :

```
* new upstream version
```

Quelques outils peuvent vous aider à créer des entrées et à finaliser le fichier `changelog` pour une livraison — voir les sections '`devscripts`' page 112 et '`dpkg-dev-el`' page 113.

Voir aussi 'Les meilleures pratiques pour le fichier `debian/changelog`' page 76.

5.3 Tester le paquet

Avant d'envoyer votre paquet, vous devriez faire quelques tests de base. Vous devriez au moins faire les tests suivants (il vous faut une ancienne version du paquet) :

– Installez le paquet et vérifiez que le logiciel fonctionne. Si le paquet existait déjà dans une version plus ancienne, faites une mise à jour.
– Exécutez `lintian` sur votre paquet. Vous pouvez exécuter `lintian` comme suit : `lintian -v` *`version-paquet`*`.changes`. Ce programme fera une vérification sur les paquets source et binaire. Si vous ne comprenez pas les messages retournés par `lintian`, essayez l'option `-i`. Cette option rendra `lintian` beaucoup plus bavard dans sa description du problème.
  En principe, un paquet pour lequel `lintian` renvoie des erreurs (elles commencent par `E`) *ne doit pas* être installé dans l'archive.
  Pour en savoir plus sur `lintian`, reportez-vous à la section '`lintian`' page 108.
– Vous pouvez facultativement exécuter '`debdiff`' page 109 pour analyser les modifications depuis une ancienne version si celle-ci existe.
– Faites régresser le paquet vers sa version précédente si elle existe — cela permet de tester les scripts `postrm` et `prerm`.
– Désinstallez le paquet et réinstallez-le.
– Copiez le paquet source dans une répertoire différent et tentez de le décompacter et de le reconstruire. Ceci teste si le paquet repose sur des fichiers existants en dehors de ceux du paquet ou s'il repose sur des permissions préservées des fichiers livrés dans le fichier .diff.gz.

5.4 Disposition du paquet source

Il existe deux types de paquets source Debian :

– les paquets appelés *natifs* pour lesquels il n'y a pas de distinction entre les sources d'origine et les correctifs appliqués pour Debian
– les paquets (plus courants) où il y a un fichier archive source d'origine accompagné par un autre fichier contenant les correctifs pour Debian.

Pour les paquets natifs, le paquet source inclut un fichier de contrôle source Debian (`.dsc`) et l'archive source (`.tar.gz`). Un paquet source d'un paquet non natif inclut un fichier de contrôle source Debian, l'archive source d'origine (`.orig.tar.gz`) et les correctifs Debian (`.diff.gz`).

Le caractère natif d'un paquet est déterminé lorsqu'il est construit par `dpkg-buildpackage(1)`. Le reste de cette section ne se rapporte qu'aux 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 téléchargé et mentionné dans le fichier `.changes`. Par la suite, ce fichier `tar` sera utilisé pour générer les fichiers `diff` et `.dsc` et il ne sera pas nécessaire de le télécharger à nouveau.

Par défaut, `dpkg-genchanges` et `dpkg-buildpackage` incluront le fichier `tar` amont si et seulement si le numéro de révision du paquet source est 0 ou 1, ce qui indique une nouvelle version amont. 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 originaux, `dpkg-source` *doit*, pour construire les fichiers `.dsc` et `diff` de la mise à jour, utiliser un fichier `tar` identique à l'octet près à celui déjà présent dans l'archive.

Veuillez noter que, dans des paquets non natifs, les permissions sur des fichiers qui ne sont pas présents dans l'archive .orig.tar.gz ne seront pas préservées car diff ne stocke pas les permissions de fichier dans le correctif.

5.5 Choisir une distribution

Chaque envoi doit spécifier 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 fait, il y a deux autres possibilités : « stable-security » et « testing-security », mais veuillez lire 'Gérer les bogues de sécurité' page 46 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 spécial : mise à jour d'un paquet de la distribution *stable*

Livrer un paquet pour la distribution *stable* signifie que le paquet sera dirigé vers le répertoire `stable-proposed-updates` des archives Debian pour y être testé avant d'être effectivement inclus dans *stable*.

Une livraison 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, des envois vers *stable* étaient également utilisés pour corriger les problèmes de sécurité. Cependant, cette pratique est dépréciée car les envois utilisés pour les avis de sécurité Debian[2] sont automatiquement copiés dans l'archive `proposed-updates` appropriée quand l'avis est publié. Reportez-vous à 'Gérer les bogues de sécurité' page 46 pour des informations plus détaillées sur la gestion des problèmes de sécurité.

[2] Debian security advisory

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

Les paquets livrés pour *stable* doivent être compilés avec la distribution *stable* pour que leurs dépendances se limitent aux bibliothèques (et autres paquets) disponibles dans *stable* ; un paquet livré pour la distribution *stable* qui dépend d'une bibliothèque qui n'est disponible que dans *unstable* sera rejeté. Modifier les dépendances d'autres paquets (en manipulant le champ `Provides` ou les fichiers shlibs) et, peut-être, rendre ces paquets non installables, est fortement déconseillé.

L'équipe responsable de la distribution[3] (joignable à l'adresse `<debian-release@lists.debian.org>`) évaluera régulièrement le contenu de *stable-proposed-updates* et décidera si votre paquet peut être inclus dans la distribution *stable*. Soyez précis (et, si nécessaire, prolixe) quand vous décrivez, dans le fichier changelog, vos changements pour une livraison vers *stable*, sinon le paquet ne sera pas pris en considération.

Il est fortement conseillé de discuter avec le responsable de la version stable *avant* de réaliser un envoi dans *stable*/*stable-proposed-updates*, pour que le paquet envoyé corresponde aux besoins de la prochaine révision de *stable*.

5.5.2 Cas spécial : mise à jour d'un paquet de la distribution *testing*/*testing-proposed-updates*

Veuillez consulter les informations dans la section sur *testing* pour des détails.

5.6 Mettre à jour un paquet

5.6.1 Installer un paquet sur `ftp-master`

Pour installer un paquet, vous devez envoyer les fichiers (y compris les fichiers changes et dsc signés) par ftp anonyme sur `ftp-master.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*) Debian.

Attention, il est préférable de transférer le fichier `changes` en dernier. Dans le cas contraire, votre livraison 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`' page 112 ou '`dput`' page 112 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 README dans ce répertoire FTP et le paquet Debian '`dcut`' page 112.

[3] *the Release team*

5.6.2 Installer un paquet sur non-US

Note : non-us a été interrompu avec la publication de *Sarge*.

5.6.3 Envois différés

Les envois différés sont pour le moment réalisés *via* la file différée sur gluck. Le répertoire d'envoi est `gluck:~tfheen/DELAYED/[012345678]-day`. 0-day est envoyé approximativement plusieurs fois par jour vers ftp-master.

Avec une version assez récente de dput, cette section

```
[tfheen_delayed]
method = scp
fqdn = gluck.debian.org
incoming = ~tfheen
```

dans votre fichier ~/.dput.cf devrait fonctionner correctement pour réaliser des envois dans la file DELAYED.

Note : Comme la file d'envoi va sur `ftp-master`, la prescription que l'on trouve dans 'Installer un paquet sur `ftp-master`' page précédente s'applique également ici.

5.6.4 Envois de sécurité

N'envoyez **PAS** 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 délais dans la gestion de cet envoi non désiré. Pour plus de détails, consultez la section 'Gérer les bogues de sécurité' page 46.

5.6.5 Les autres files d'envoi

Les files scp sur ftp-master et security sont pratiquement inutilisables à cause des restrictions de connexion sur ces hôtes.

Les files anonymes sur ftp.uni-erlangen.de et ftp.uk.debian.org sont actuellement arrêtées. Un travail est en cours pour les restaurer.

Les files sur master.debian.org, samosa.debian.org, master.debian.or.jp et ftp.chiark.greenend.org.uk sont arrêtées de façon permanente et ne seront pas restaurées. La file du Japon sera remplacée par une nouvelle file sur hp.debian.or.jp un jour.

À l'heure actuelle, la file en ftp anonyme sur auric.debian.org (le précédent ftp-master) fonctionne, mais elle est déconseillée et sera supprimée à un moment donné.

5.6.6 Notification de l'installation d'un 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 `katie`. Les mises à jour de paquets sur la distribution *unstable* sont installées ainsi. Dans les autres cas et notamment dans le cas d'un nouveau paquet, celui-ci sera installé manuellement. Il peut s'écouler jusqu'à un mois entre le téléchargement d'un paquet vers un serveur et son installation effective. Soyez 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 bogues ont été clos. Lisez attentivement ce courrier et vérifiez 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, le logiciel de démon de file vous enverra également une notification par courriel.

5.7 Spécifier la section, la sous-section et la priorité d'un paquet

Les champs `Section` et `Priority` du fichier `debian/control` n'indiquent ni où le paquet sera installé dans l'archive Debian, ni sa priorité. Afin de conserver la cohérence de l'archive, ce sont les administrateurs qui contrôlent ces champs. Les valeurs du fichier `debian/control` sont juste des indications.

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` de votre paquet diffèrent, vous en serez informé par courrier électronique quand votre paquet sera installé dans l'archive. Vous pourrez corriger votre fichier *debian/control* avant votre prochain téléchargement 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 fichier `debian/control` est correct. Ensuite, envoyez un courrier à `<override-change@debian.org>` ou un rapport de bogue sur le pseudo-paquet `ftp.debian.org` demandant la modification de la section ou de la priorité de votre paquet. Exposez bien les raisons qui vous amènent à demander ces changements.

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 dans 'Les sections' page 21. Si la section est « main », elle devrait être omise. La liste des sous-sections autorisées peut être trouvée dans `http://www.debian.org/doc/debian-policy/ch-archive.html#s-subsections`.

5.8 Gérer les bogues

Chaque développeur doit être capable de travailler avec le système de suivi des bogues (http://www.debian.org/Bugs/) Debian. Il faut savoir comment remplir des rapports de bogue correctement (voir 'Rapporter des bogues' page 95), comment les mettre à jour et les réordonner et comment les traiter et les fermer.

Les fonctionnalités du système de suivi des bogues sont décrites dans la documentation du BTS pour les développeurs (http://www.debian.org/Bugs/Developer). Ceci inclut la fermeture des bogues, l'envoi des messages de suivi, l'assignation des niveaux de gravité et des marques, l'indication que les bogues ont été envoyés au développeur amont et autres problèmes.

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 courrier de contrôle. Toutes les commandes disponibles pour ce serveur sont décrites dans la documentation du serveur de contrôle du BTS (http://www.debian.org/Bugs/server-control).

5.8.1 Superviser les rapports de bogue

Si vous voulez être un bon responsable, consultez régulièrement la page du système de suivi des bogues (http://www.debian.org/Bugs/). Cette page contient tous les rapports de bogue qui concernent vos paquets. Vous pouvez les vérifier avec cette page : `http://bugs.debian.org/`*`votre_compte`*`@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 bogues 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 Répondre à des rapports de bogue

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 veut dire que vous n'avez pas besoin d'envoyer une copie du courrier à `<123@bugs.debian.org>`).

Si vous recevez un bogue mentionnant « FTBFS », cela veut dire « Fails To Build From Source » (Erreur de construction à partir du source). Les porteurs emploient fréquemment cet acronyme.

Une fois que vous avez traité un rapport de bogue (i.e. que vous l'avez corrigé), marquez-le comme *done* (fermez-le) 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 dans 'Quand les rapports de bogue sont-ils fermés par une mise à jour ?' page 45.

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 générale des bogues

En tant que responsable de paquet, vous trouverez fréquemment des bogues dans d'autres paquets ou vous aurez des bogues sur vos paquets qui sont en fait des bogues sur d'autres paquets. Les fonctions intéressantes du système de suivi des bogues sont décrites dans la documentation du BTS pour les développeurs Debian (`http://www.debian.org/Bugs/Developer`). Les instructions du serveur de contrôle du BTS (`http://www.debian.org/Bugs/server-control`) 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 paquet est l'une des « obligations civiques » du responsable, reportez-vous à 'Rapporter des bogues' page 95 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 gérer un rapport de bogue :

1 Décider si le rapport correspond à un bogue réel ou non. Parfois, les utilisateurs exécutent simplement un programme de la 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 ré-ouvrir 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 avec `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 réattribuant 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 (`http://www.debian.org/devel/tech-ctte`).

2 Si le bogue est réel, mais est causé par un autre paquet, réattribuez simplement le bogue à l'autre paquet. Si vous ne savez pas à quel paquet il doit être réattribué, vous pouvez demander de l'aide sur IRC ou sur `<debian-devel@lists.debian.org>`. Veuillez vous assurer que le (ou les) responsable(s) du paquet sur lequel est réattribué le paquet sait pourquoi vous l'avez ainsi réattribué.

   Parfois, vous devez également ajuster la gravité du bogue pour qu'elle corresponde à notre 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 d'ordre cosmétique.

3 Si le bogue est réel, mais que le problème a déjà été rapporté par quelqu'un d'autre, les deux rapports de bogues pertinents devraient être fusionnés en un seul en utilisant la commande `merge` (fusion) du BTS. Ainsi, quand le bogue sera corrigé, tous les créateurs de rapport de bogue en seront informés (notez, cependant, que les courriels envoyés à l'un des créateurs de rapport de bogue ne seront pas automatiquement envoyés aux autres créateurs de rapport de bogue). Pour plus de détails sur les technicités de la commande de fusion et sa voisine, la commande `unmerge` (séparation), veuillez consulter la documentation du serveur de contrôle du BTS.

4 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.

5 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, vous le fermez simplement, sinon vous devez le rappeler à l'auteur. Si vous avez les compétences nécessaires, vous pouvez préparer un correctif pour corriger 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).

6 Si vous avez corrigé un bogue sur votre copie locale ou si un correctif a été inclus dans le référentiel CVS, vous pouvez marquer le bogue avec `pending` (en attente) pour informer que le bogue est corrigé et qu'il sera fermé lors du prochain envoi (ajoutez le `closes:`

dans le `changelog`). C'est particulièrement utile si plusieurs développeurs travaillent sur le même paquet.

7 Une fois que le paquet corrigé est disponible dans la distribution *unstable*, vous pouvez fermer le bogue. Ceci peut être fait automatiquement, pour cela, reportez-vous à 'Quand les rapports de bogue sont-ils fermés par une mise à jour ?' de la présente page.

5.8.4 Quand les rapports de bogue sont-ils fermés par une mise 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 vous devez 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 d'éviter d'avoir à fermer manuellement les bogues après l'envoi — indiquez simplement les bogues corrigés dans le fichier `changelog` en suivant une syntaxe précise. 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.
```

Techniquement parlant, l'expression rationnelle Perl suivante décrit comment les lignes de *changelogs* de fermeture de bogues sont identifiées :

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

Nous préférons la syntaxe `closes: #`*XXX*, car c'est l'entrée la plus concise et la plus facile à intégrer dans le texte du fichier `changelog`.

À moins que cela soit spécifié différemment par l'option *-v* de `dpkg-buildpackage`, seuls les bogues fermés dans l'entrée de changelog la plus récente sont fermés (fondamentalement, seuls les bogues mentionnés dans la partie de changelog 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 cassé 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 dans les 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 pas** pas fermer des bogues dans une entrée de changelog d'une version 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 de changelog, veuillez vous reporter aux 'Les meilleures pratiques pour le fichier `debian/changelog`' page 76.

5.8.5 Gérer les 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 le site 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 à `<team@security.debian.org>` dès que possible. **N'ENVOYEZ PAS** de paquet pour *stable* ; l'équipe de sécurité le fera. Les informations utiles incluent, par exemple :

- les versions du paquet connues pour être affectées par le bogue. Vérifiez chaque version présente dans les distributions maintenues par Debian ainsi que dans *testing* et dans *unstable*,
- la nature d'une solution si elle est disponible (les correctifs sont particulièrement utiles),
- tout paquet corrigé que vous avez vous-même préparé (envoyez seulement les fichiers `.diff.gz` et `.dsc` et lisez d'abord 'Préparer les paquets pour corriger des problèmes de sécurité' page 48),
- toute assistance que vous pouvez fournir pour aider les tests (exploits, tests de régression, etc.),
- toute information nécessaire pour l'annonce de sécurité (voir 'Annonces de sécurité' page suivante).

Confidentialité

À la différence de la plupart des autres activités au sein de Debian, les informations sur les problèmes de sécurité doivent parfois être gardées en privé pour un certain temps. Ceci permet

aux distributeurs de logiciels de coordonner leur dévoilement afin de minimiser l'exposition de leurs utilisateurs. Cette décision dépend de la nature du problème et de l'existence d'une solution correspondante et également s'il s'agit d'un fait connu publiquement.

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 remplit un rapport de bogue,
- quelqu'un l'informe par courrier privé.

Dans les deux premiers cas, l'information est publique et il est important d'avoir une solution le plus vite possible. Dans le dernier cas, cependant, ce n'est peut-être pas une information publique. Dans ce cas, il existe quelques options possibles pour traiter le problème :

- si l'exposition de sécurité est mineure, il n'y a parfois pas besoin de garder le secret sur le problème et une correction devrait être effectuée et diffusée,
- si le problème est grave, il est préférable de partager cette information avec d'autres vendeurs et de coordonner une publication. L'équipe de sécurité reste en contact avec les différentes organisations et individus et peut prendre soin des actions à mener.

Dans tous les cas, si la personne qui indique 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 qu'une correction puisse être effectuée pour la version stable de Debian. Quand vous envoyez des informations confidentielles à l'équipe de sécurité, assurez-vous de bien mentionner ce fait.

Veuillez noter que si le secret est nécessaire, vous ne pourrez pas envoyer un correctif vers *unstable* (ou ailleurs) puisque les informations de changelog et de diff sont publiques pour *unstable*.

Il existe deux raisons pour 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.

Annonces de sécurité

Les annonces de sécurité ne sont émises que pour la distribution actuelle diffusée *stable*, *PAS* pour *testing* ou *unstable*. Quand elle est diffusée, l'annonce est envoyée à la liste de diffusion `<debian-security-announce@lists.debian.org>` et elle est postée sur la page de sécurité (`http://www.debian.org/security/`). Les annonces de sécurité sont écrites et postées par les membres de l'équipe de sécurité. Cependant, ils ne verront aucun inconvénient à ce qu'un responsable leur apporte des informations ou écrive une partie du texte. Les informations qui devraient être présentes dans une annonce incluent :
- une description du problème et de sa portée, y compris :
  - le type du problème (usurpation de privilège, déni de service, etc.),
  - quels sont les privilèges obtenus et par quels utilisateurs (si c'est le cas)
  - comment il peut être exploité,
  - si le problème peut être exploité à distance ou localement,
  - comment le problème a été corrigé,

Ces informations permettent aux utilisateurs d'estimer la menace pesant sur leurs systèmes.
- les numéros de version des paquets affectés,
- les numéros de version des paquets corrigés,
- une information sur la façon de récupérer les paquets mis à jour (habituellement l'archive de sécurité Debian),
- des références à des annonces amont, des identifiants CVE (`http://cve.mitre.org`) et toute autre information utile pour recouper les références de la vulnérabilité.

Préparer les paquets pour corriger des problèmes de sécurité

Une façon d'aider l'équipe de sécurité dans ses travaux est de lui fournir des paquets corrigés convenables pour une annonce de sécurité pour 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 un comportement identique dans une version lorsque celle-ci est diffusée, donc tout changement qui est fait 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 veut dire 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 récrites. 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 au préalable.

Il existe une autre règle de conduite liée à cela : 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 parfois un correctif de sécurité peut casser de manière subtile des fonctionnalités qui ne semblent pas liées.

N'incluez **PAS** de changements dans votre paquet qui ne soient pas liés directement à la correction de la vulnérabilité. Ceux-ci devraient être ensuite enlevés et cela perd du 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 seraient sinon rejetés de la distribution stable, n'essayez donc pas de faire cela, s'il vous plaît.

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 utiles pour cela, voir '`debdiff`' page 109).

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

- Ciblez la bonne distribution dans votre fichier `debian/changelog`. Pour *stable*, il s'agit de `stable-security` et pour *testing*, c'est `testing-security` et pour l'ancienne distribution stable, c'est `oldstable-security`. Ne ciblez ni *distribution*-proposed-updates, ni `stable` !
- L'envoi devra avoir « 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. Incluez toujours une référence externe, de préférence un identifiant CVE, pour qu'elle puisse être recoupée. Incluez la même information dans le changelog pour *unstable* pour qu'il soit clair que le même bogue a été corrigé car cela est très utile pour vérifier que le bogue a été corrigé pour la prochaine version stable. Si aucun identifiant CVE n'a encore été assigné, l'équipe de sécurité en demandera un pour qu'il puisse être inclus dans le paquet et dans le changelog.
- Assurez-vous que le numéro de version est correct. Il doit être plus élevé que celui du paquet actuel, mais moins que ceux des paquets des versions des distributions suivantes. En cas de doute, testez-le avec `dpkg --compare-versions`. Soyez attentif à ne pas ré-utiliser un numéro de version que vous auriez déjà utilisé pour un précédent envoi. Pour *testing*, il doit y avoir un numéro de version supérieur dans *unstable*. S'il n'y en a pas encore (par exemple, si *testing* et *unstable* ont la même version), vous devez envoyer une nouvelle version vers *unstable* en premier.
- Ne faites pas d'envoi de source seul si votre paquet possède un ou plusieurs paquets binary-all (n'utilisez pas l'option `-S` de `dpkg-buildpackage`). L'infrastructure `buildd` ne construira pas ceux-ci. Ce point s'applique aux envois de paquets normaux également.
- Sauf si la source amont a été envoyée auparavant à security.debian.org (par une précédente mise à jour de sécurité), construisez le paquet avec la source amont complète (`dpkg-buildpackage -sa`). S'il y a déjà eu un précédent envoi à security.debian.org, vous pouvez faire un envoi sans le paquet source (`dpkg-buildpackage -sd`).
- Assurez-vous d'utiliser exactement le même nom `*.orig.tar.gz` 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, dont tous les paquets appartiennent à la distribution pour laquelle vous construisez le paquet. Si vous n'avez pas un tel système, vous pouvez utiliser l'une des machines de debian.org (voir 'Les serveurs Debian' page 16) ou mettez en place un chroot (voir '`pbuilder`' page 111 et '`debootstrap`' page 111).

Mettre à jour le paquet corrigé

Vous *NE* devez *PAS* 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 délais dans la gestion de l'envoi indésirable.

Vous *NE* devez *PAS* envoyer votre correction dans *proposed-updates* sans vous coordonner avec l'équipe de sécurité. Les paquets seront copiés de security.debian.org dans le 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 à la place sans la mise à jour de sécurité de ce paquet.

Une fois que vous avez créé et testé le nouveau paquet 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 que l'envoi vers la file d'attente de sécurité a été 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 nécessaire 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 qui convient sur ftp-master.

5.9 Déplacer, effacer, changer le nom, adopter et abandonner des paquets

Certaines manipulations de l'archive ne sont pas possibles avec le processus de mise à jour automatisé. Elles sont appliquées manuellement par les développeurs. Ce chapitre décrit ce qu'il faut faire dans ces situations.

5.9.1 Déplacer des paquets

Il se peut qu'un paquet puisse changer de section. Une nouvelle version d'un paquet de la section `non-free` pourrait, par exemple, être distribuée sous licence GNU GPL ; dans ce cas, le paquet doit être déplacé dans la section `main` ou `contrib`[4].

Si vous avez besoin de modifier la section de l'un de vos paquets, modifiez les informations de contrôle du paquet pour le placer dans la section désirée et téléchargez à nouveau votre paquet dans l'archive. Reportez-vous à la charte Debian (`http://www.debian.org/doc/debian-policy/`) pour en savoir plus. Vous devez vous assurer d'inclure le fichier `.orig.tar.gz` dans votre envoi (même si vous n'envoyez pas de nouvelle version amon) ou il n'apparaîtra pas dans la nouvelle section avec le reste du paquet. Si votre nouvelle section est valide, il sera déplacé automatiquement. Si ce n'est pas le cas, contactez les responsables ftp pour comprendre ce qui s'est passé.

[4] Reportez-vous à la charte Debian (`http://www.debian.org/doc/debian-policy/`) pour savoir dans quelle section un paquet doit être classé.

Si vous avez besoin de modifier la sous-section de l'un de vos paquets (`devel` ou `admin` par exemple), la procédure est légèrement différente. Modifiez la sous-section dans le fichier de contrôle de votre paquet et téléchargez-le. Il vous faudra ensuite demander la modification du fichier *override* comme décrit dans la section 'Spécifier la section, la sous-section et la priorité d'un paquet' page 41.

5.9.2 Supprimer des paquets

Si, pour une raison ou une autre, vous avez besoin de supprimer complètement un paquet de l'archive (disons qu'il s'agit d'une vieille bibliothèque devenue inutile que l'on conservait pour des raisons de compatibilité), il vous faudra envoyer un rapport de bogue concernant le pseudo-paquet `ftp.debian.org` et demander sa suppression ; comme pour tous les bogues, ce bogue devrait être de gravité normale. N'oubliez pas de préciser de quelle distribution le paquet doit être supprimé. Normalement, vous ne devriez avoir à supprimer que des paquets d'*unstable* ou d'*experimental*. Les paquets de *testing* ne sont pas supprimés directement. Ils sont plutôt enlevés automatiquement après que le paquet a été supprimé d'*unstable* et si aucun paquet de *testing* n'en dépend.

Il existe une exception pour laquelle il n'est pas nécessaire de faire une demande explicite de suppression : si un paquet (source ou binaire) est orphelin, il sera supprimé de façon semi-automatique. Pour un paquet binaire, cela veut dire s'il n'y a plus de paquet source produisant le 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.

Vous devez détailler dans votre demande de suppressions les raisons justifiant cette demande. Ceci a pour but d'éviter les suppressions non désirées 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.

Vous ne pouvez 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.

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 d'autres programmes utiles, citons `apt-cache rdepends`, `apt-rdepends` et `grep-dctrl`. Le retrait de paquets orphelins est discuté sur `<debian-qa@lists.debian.org>`.

Une fois que le paquet a été supprimé, les bogues du paquet doivent être gérés. Soit ils sont réattribués à un autre paquet dans le cas où le code actuel a évolué en un autre paquet (par exemple, `libfoo12` a été supprimé parce que `libfoo13` le remplace) ou ils sont fermés si le logiciel ne fait simplement plus partie de Debian.

Supprimer des paquets dans `Incoming`

Par le passé, il était possible de supprimer un paquet de `Incoming`. Cependant, ce n'est plus possible depuis la mise en place du nouveau système de file d'attente. Il vous faudra télécharger une nouvelle version de votre paquet avec un numéro de version plus élevé que celui que vous voulez remplacer. Les deux versions seront installées dans l'archive mais seule la plus récente sera accessible dans *unstable* car la précédente sera immédiatement remplacée par la nouvelle. Toutefois, si vous testez correctement vos paquets, le besoin d'en remplacer un ne devrait pas être trop fréquent.

5.9.3 Remplacer un paquet ou changer son nom

Si vous vous trompez en nommant un paquet, vous devrez intervenir en deux étapes pour changer son nom. D'abord, modifiez votre fichier `debian/control` pour que votre nouveau paquet remplace et entre en conflit avec l'ancien paquet que vous voulez remplacer (reportez-vous à la charte Debian (`http://www.debian.org/doc/debian-policy/`) pour les détails). Une fois que votre paquet est installé dans l'archive, faites un rapport de bogue concernant le pseudo-paquet `ftp.debian.org` et demandez la suppression du paquet mal nommé. N'oubliez pas de réattribuer correctement les bogues du paquet en même temps.

D'autres fois, vous pouvez commettre une erreur en construisant le paquet et vous désirez 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 : si vous désirez remplacer l'archive source amont de votre paquet, vous devez l'envoyer 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`. Cette restriction donne à chaque fichier du site ftp un nom unique, ce qui aide à garantir la consistance dans le réseau des miroirs.

5.9.4 Abandonner un paquet

Si vous ne pouvez plus maintenir un paquet, vous devez en informer les autres et faire le nécessaire pour qu'il soit marqué *abandonné* (i.e. *orphaned*). Vous devriez aussi 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 pseudo-paquet `wnpp`. Le titre de votre rapport de bogue devrait être « O[5] : *`paquet -- courte description`* » pour indiquer que le paquet est abandonné. La gravité du bogue sera *normale* ; si le paquet a une priorité standard ou supérieure, elle devrait être *importante*. 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 : car de cette manière 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 à la place soumettre un rapport de bogue sur `wnpp`

[5] *Orphaned* : abandonné.

et l'intituler `RFA: paquet -- description courte`. `RFA` veut dire *Request For Adoption* (demande d'adoption).

Vous pouvez trouver plus d'informations sur les pages web WNPP (`http://www.debian.org/devel/wnpp/`)[6].

5.9.5 Adopter un paquet

Une liste des paquets en attente de nouveau responsable est disponible à la page paquets en souffrance et paquets souhaités (`http://www.debian.org/devel/wnpp/`). Si vous voulez prendre en charge un paquet de cette liste, reportez-vous à la page mentionnée ci-dessus pour connaître la procédure à suivre.

Prendre un paquet parce qu'il vous semble que celui-ci est 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 depuis un moment, veuillez vous reporter à 'Gérer les responsables non joignables' page 98).

Généralement, vous ne pouvez pas adopter un paquet sans l'accord du responsable actuel. 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 (`http://www.debian.org/devel/tech-ctte`) pour plus d'information).

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 que vous aurez installé une nouvelle version du paquet dans l'archive avec le champ `Maintainer` à jour. Cela peut prendre quelques heures après le téléchargement. Si vous pensez ne pas avoir de mise à jour à faire pour un moment, vous pouvez utiliser le 'Système de suivi des paquets' page 28 pour recevoir les rapports de bogue. Cependant, assurez-vous que cela ne pose aucun problème à l'ancien responsable de continuer à recevoir les bogues durant ce temps.

5.10 Le portage

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

Porter un paquet consiste à faire un paquet binaire pour des architectures différentes de celle du paquet binaire fait par le 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.

[6] *Work-needing and prospective packages* : paquets en souffrance et paquets souhaités.

Pour un paquet binaire *i386*, par exemple, il faut compter une recompilation pour chaque autre architecture, soit un total de 12 recompilations.

5.10.1 Être courtois 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 commises régulièrement par les responsables Debian — problèmes courants qui bloquent souvent les porteurs et compliquent inutilement leur travail.

Ici, la première et la plus importante chose est de répondre rapidement aux rapports de bogues et remarques 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 de mise en paquet dans le paquet source. Voici un pense-bête pour les choses auxquelles vous devez être attentif :

1. Vérifiez que les champs `Build-Depends` et `Build-Depends-Indep` du fichier `debian/control` sont corrects. Le meilleur moyen de le vérifier est d'utiliser le paquet `debootstrap` pour créer un environnement *unstable chrooté* (voir '`debootstrap`' page 111). Dans cet environnement *chrooté*, il faudra installer le paquet `build-essential` et tous les paquets mentionnés dans les champs `Build-Depends` et `Build-Depends-Indep`. Ensuite, vous essayerez de fabriquer votre paquet dans cet environnement. Ces étapes peuvent être automatisées en utilisant le programme `pbuilder` qui est fourni par le paquet de même nom (voir '`pbuilder`' page 111).
   Si vous n'arrivez pas à installer un environnement *chrooté*, `dpkg-depcheck` pourra peut-être vous aider (voir '`dpkg-depcheck`' page 113).
   Consultez la charte Debian (`http://www.debian.org/doc/debian-policy/`) 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 pour le faire. Trop souvent, les développeurs ne respectent pas les instructions de la charte Debian (`http://www.debian.org/doc/debian-policy/`). Choisir la valeur « i386 » est la plupart du temps incorrect.
3. Vérifiez que votre paquet source est bon. 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` et `debian/substvars` ne sont pas dans votre paquet source. Ils doivent être effacés par la cible *clean* de `debian/rules`.
5. Assurez-vous que vous ne vous appuyez pas sur des éléments de configuration ou des 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 vous appuyer sur des 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 de votre paquet (un sous-cas de la remarque précédente).

7 Si possible, ne vous appuyez pas sur une particularité présente dans un compilateur précis ou dans une certaine version d'un compilateur. Si vous ne pouvez pas faire autrement, assurez-vous que les dépendances de fabrication reflètent bien cette restriction. Dans ce cas, vous cherchez sûrement les problèmes car quelques architectures pourraient choisir un compilateur différent.

8 Vérifiez que votre 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 Instructions pour les mises à jour des porteurs

Si le paquet se construit tel quel sur l'architecture que vous visez, vous avez de la chance et votre travail est facile. Cette section s'applique dans ce cas ; elle décrit comment construire et installer correctement votre paquet binaire dans l'archive Debian. Si vous devez modifier le paquet pour le rendre compilable sur votre architecture cible vous devez faire une mise à jour des sources, consultez la section 'Comment faire une mise à jour indépendante ?' page 59.

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 (cela inclut 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 votre envoi localement pour son acceptation dans l'archive, vous pouvez exécuter `debsign` sur votre fichier `.changes` pour qu'il soit signé de manière commode ou utilisez le mode de signature à distance de `dpkg-sig`.

Mises à jour indépendantes binaires ou recompilations

Parfois, l'envoi du portage initial pose problème car l'environnement dans lequel le paquet a été construit n'était pas bon (bibliothèques plus à jour 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 (`katie` refuse d'installer de nouveaux paquets s'ils n'ont pas un numéro de version supérieur à celui actuellement disponible).

Vous devez vous assurer que votre mise à jour indépendante binaire 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 les modifications nécessaires du changelog, ce type de mise à jour reste une mise à jour indépendante binaire — 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 mise à jour indépendante par recompilation uniquement 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 que vous avez recompilée était la version 2.9.3, votre mise à jour indépendante aura le numéro de version 2.9-3+b1. Si la dernière version était 3.4+b1 (i.e. un paquet natif avec une précédente mise à jour indépendante par recompilation), votre mise à jour indépendant aura le numéro de version 3.4+b2. [7]

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.

Quand faire une mise à jour indépendante source pour un portage ?

Les porteurs qui font des mises à jour indépendantes sources suivent généralement les instructions de la section 'Mise à jour indépendante' page 59 tout comme les non-porteurs. Les délais d'attente sont cependant plus courts car les porteurs doivent manipuler un grand nombre de paquets. À nouveau, la situation diffère selon la distribution visée. Elle varie également selon que l'architecture est candidate pour inclusion dans la prochaine version stable ; les responsables de publication décident et annoncent quelles architectures sont candidates.

Si vous êtes porteur et faites une mise à jour 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 système de suivi des bogues et le moment où vous pouvez faire une mise à jour indépendante *(NMU)* — est de sept jours. Ce délai peut être raccourci 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 tout d'abord vous coordonner avec l'équipe de publication appropriée.

Deuxième différence, les porteurs qui font des mises à jour indépendantes sources doivent choisir une gravité *sérieuse* (i.e. *serious*) ou supérieure quand ils envoient leur rapport au sys-

[7] Par le passé, de telles mises à jour indépendantes utilisaient le numéro de troisième niveau de la partie Debian de la révision pour dénoter l'état de recompilation uniquement ; cependant, cette syntaxe était ambigüe pour les paquets natifs et ne permettait pas un ordre correct des mises à jour indépendantes par recompilation uniquement, des mises à jour indépendantes de source et des mises à jour indépendantes de sécurité sur le même paquet et elle a donc été abandonnée en faveur de cette nouvelle syntaxe.

tème de suivi des bogues. Ceci assure qu'un paquet source unique permet de produire un paquet binaire pour chaque architecture supportée 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. Quelques fois, 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 que l'on sache le retirer une fois que le problème aura disparu.

Les porteurs peuvent aussi avoir une adresse où ils publient 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 adresses n'ont rien d'officiel, alors soyez sur vos gardes si vous les utilisez.

5.10.3 Infrastructure de portage et automatisation

Il existe une infrastructure et plusieurs outils pour faciliter l'automatisation du portage des paquets. Cette section contient un bref aperçu de cette automatisation et du portage de ces outils ; veuillez vous reporter à la documentation des paquets ou les références pour une information complète.

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.

Outils pour les porteurs

Les descriptions de plusieurs outils de portage peuvent être trouvées dans les 'Outils de portage' page 114.

`buildd`

Le système `buildd` est un système distribué pour la compilation d'une distribution. Il est habituellement utilisé en conjonction avec des automates de compilation ; ce sont des machines « esclaves » qui récupèrent des paquets sources et tentent de les compiler. Il est aussi possible d'interagir par courrier électronique avec ce système. Cette interface est utilisée par les

porteurs pour récupérer un paquet source (en général, un paquet qui ne peut être compilé automatiquement) et travailler dessus.

`buildd` n'est pas disponible sous forme de paquet ; pourtant, la plupart des équipes de porteurs l'utilisent aujourd'hui ou ont prévu de l'utiliser bientôt. L'outil de construction automatisé réel est dans le paquet `sbuild`, voir la description dans '`sbuild`' page 111. Le système `buildd` regroupe également un ensemble de composants très utiles, continuellement utilisés, mais non encore mis en paquet, tels que `andrea` et `wanna-build`.

Une partie des informations produites par `buildd` — utiles pour les porteurs — est disponible sur la toile à l'adresse `http://buildd.debian.org/`. Ces informations incluent les résultats produits toutes les nuits par `andrea` (dépendances des sources) et `quinn-diff` (paquets à recompiler).

Nous sommes très fiers de ce système car il a de nombreux usages potentiels. Des groupes de développeurs indépendants peuvent utiliser ce système pour créer différentes saveurs de Debian — qui peuvent être ou ne pas être intéressantes pour tous (par exemple, une version de Debian compilée avec des vérifications relatives à `gcc`). Ce système nous permettra aussi de recompiler rapidement toute une distribution.

Les administrateurs des buildds pour chaque architecture peuvent être contactés à l'adresse électronique $arch@buildd.debian.org.

5.10.4 Quand votre paquet *n'est pas* portable

Certains paquets ont encore des problèmes pour être construits et/ou pour fonctionner sur certaines des architectures prises en charge par Debian et ne peuvent pas du tout être portés, ou pas dans un laps de temps raisonnable. Un exemple est un paquet qui est spécifique SGVA (i386 seulement) ou qui utilise des fonctionnalités spécifiques au matériel qui ne sont pas gérées sur toutes les architectures.

Pour éviter que des paquets cassés soient envoyés dans l'archive et qu'ils fassent perdre du temps des 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 moyens de faire cela. Le moyen préféré est d'avoir une petite suite de tests pendant la construction qui testera la fonctionnalité et qui échouera si cela ne fonctionne pas. C'est de toute façon une bonne idée et empêchera des (certains) envois cassés pour toutes les architectures, et cela permettra également au paquet d'être construit dès que la fonctionnalité nécessaire est disponible.
  De plus, si vous croyez que la liste des architectures gérées est plutôt constante, vous devriez changer « any » en une liste des architectures gérées dans le fichier `debian/control`. Ainsi, la construction échouera également et l'indiquera à un lecteur humain sans vraiment essayer.
– Pour empêcher les compilateurs automatiques de tenter sans raison de construire votre paquet, il doit être inclus dans `packages-arch-specific`, une liste utilisée par le script `wanna-build`. La version actuelle est disponible à `http://cvs.debian.org/srcdep/Packages-arch-specific?cvsroot=dak` ; veuillez consulter le début du fichier pour savoir qui contacter pour le modifier.

Veuillez noter qu'il est insuffisant de simplement ajouter votre paquet à Packages-arch-specific sans le faire également 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 dans le passé, certains paquets binaires ont été envoyés pour des architectures non gérées, demandez leur suppression en remplissant un bogue sur `ftp.debian.org`.

5.11 Mise à jour indépendante

Dans certaines circonstances, il est nécessaire qu'une personne autre que le responsable d'un paquet fasse une mise à jour de ce paquet. Ce type de mise à jour est désigné en anglais par l'expression *non-maintainer upload (NMU)*. Dans le présent document, nous traduisons librement cette expression par « mise à jour indépendante ».

Cette section ne traite que des mises à jour indépendantes de source, c.-à-d., les mises à jour qui envoient une nouvelle version d'un paquet. Pour les mises à jour indépendantes par des porteurs ou des membres de la QA, veuillez consulter 'Mises à jour indépendantes binaires ou recompilations' page 55. Si un buildd construit et envoie un paquet, cela est également à strictement parler une NMU binaire. Veuillez consulter '`buildd`' page 57 pour plus d'informations.

La raison principale pour laquelle une mise à jour indépendante est réalisée est quand un développeur a besoin de corriger le paquet d'un autre développeur pour résoudre des problèmes sérieux ou des bogues paralysants ou quand le responsable d'un paquet ne peut pas fournir une correction dans un délai raisonnable.

Tout d'abord, il est capital que ces mises à jour indépendantes soient aussi peu intrusives que possible. Ne faites pas de ménage, ne modifiez pas le nom des modules ou des fichiers, ne déplacez pas les répertoires ; plus généralement, ne corrigez pas ce qui n'est pas cassé. Faites un correctif aussi petit que possible. Si certaines choses froissent votre sens de l'esthétique, parlez-en au responsable du paquet, au responsable amont ou soumettez un rapport de bogue. Quoiqu'il en soit, les changements esthétiques *ne doivent pas* être effectués lors d'une mise à jour indépendante.

Et souvenez-vous du Serment d'Hippocrate « Par dessus tout, ne pas faire de 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.

5.11.1 Comment faire une mise à jour indépendante ?

Les mises à jour indépendantes qui corrigent des bogues de gravité importante, sérieuse et plus élevée sont encouragées et acceptées. Vous devriez vous efforcer de contacter le responsable actuel du paquet : il est peut-être sur le point d'envoyer un correctif pour le problème ou il a peut-être une meilleure solution.

Les mises à jour indépendantes doivent être réalisées pour assister un responsable de paquet à résoudre des bogues. Les responsables devraient être reconnaissants pour cette aide et les

personnes faisant la mise à jour indépendante devraient respecter les décisions du responsable et tenter d'aider personnellement le responsable dans son travail.

Une mise à jour indépendante devrait suivre toutes les conventions décrites dans cette section. Pour un envoi vers *testing* ou *unstable*, l'ordre suivant des étapes est recommandé :

- Vérifiez que les bogues du paquet qui devraient être corrigés par la mise à jour indépendante sont bien référencés dans le système de suivi des bogues. S'ils n'y sont pas, faites des rapports de bogue immédiatement.
- Attendez la réponse du responsable quelques jours. Si vous n'obtenez aucune réponse, vous pouvez l'aider en lui envoyant le correctif qui corrige le bogue. N'oubliez pas de marquer le bogue avec le mot-clé « patch ».
- Patientez quelques jours. Si vous n'avez toujours aucune réponse du responsable, envoyez-lui un courrier annonçant votre intention d'effectuer une mise à jour indépendante du paquet. Préparez la NMU comme décrit dans cette section et testez-la soigneusement sur votre machine (cf. 'Tester le paquet' page 37). Re-vérifiez que votre correctif n'a aucun effet de bord inattendu. Assurez-vous que votre correctif est aussi minimaliste et non intrusif que possible.
- Envoyez votre paquet à incoming dans `DELAYED/7-day` (cf. 'Envois différés' page 40), envoyez le correctif final au responsable par le BTS et expliquez-lui qu'il a 7 jours pour réagir s'il veut annuler la NMU.
- Suivez ce qui se passe, vous êtes responsable pour tout bogue que vous auriez introduit avec votre NMU. Vous devriez probablement utiliser le 'Système de suivi des paquets' page 28 (PTS) pour vous tenir informé de l'état du paquet après votre NMU.

Parfois, le responsable de version ou un groupe organisé de développeurs peut annoncer une certaine période de temps au cours de laquelle les règles de mise à jour indépendante seront plus souples. Ceci implique habituellement une période plus courte d'attente avant d'envoyer des correctifs et une période de délai plus courte. Il est important de noter que, même au cours de ces « chasses aux bogues », la personne désirant faire la mise à jour indépendante doit remplir des bogues et contacter en premier le développeur, et ensuite seulement passer à l'action. Veuillez vous reporter à 'Les chasses aux bogues' page 97 pour des détails.

Pour la distribution *testing*, les règles peuvent être changées par les responsables de publication. Veuillez porter une attention spéciale au fait que le moyen habituel pour un paquet d'entrer dans *testing* est de passer par *unstable*.

Pour la distribution *stable*, veuillez y apporter une attention supplémentaire. Bien sûr, les responsables de publication peuvent également modifier les règles ici. Veuillez vérifier avant votre envoi que tous vos changements sont acceptables pour inclusion dans la prochaine version stable par le responsable de publication.

Quand un bogue de sécurité est détecté, l'équipe de sécurité peut effectuer une mise à jour indépendante en utilisant ses propres règles. Veuillez vous référer à 'Gérer les bogues de sécurité' page 46 pour plus d'informations.

Pour les différences concernant les mises à jour indépendantes par les porteurs, veuillez voir 'Quand faire une mise à jour indépendante source pour un portage ?' page 56.

Bien sûr, il est toujours possible de s'accorder avec un responsable pour des règles spéciales

(comme quand le responsable demande « merci d'envoyer le correctif directement pour moi et pas de diff nécessaire »).

5.11.2 Numéro de version pour les mises à jour indépendantes

Chaque fois que vous modifiez un paquet, le numéro de version de ce paquet doit changer, même pour la plus triviale des modifications. Notre système de gestion de paquets s'appuie sur ces numéros de version.

Si vous faites une mise à jour indépendante *(NMU)*, vous devez ajouter un numéro de version mineur à la partie *révision-debian* du numéro de version (la partie qui suit le dernier trait d'union). Ce numéro supplémentaire débutera à « 1 ». Prenons pour exemple le paquet « foo » qui porte le numéro de version 1.1-3. Dans l'archive, le fichier de contrôle du paquet source serait `foo_1.1-3.dsc`. La version amont est « 1.1 » et la révision Debian est « 3 ». La mise à jour indépendante suivante ajouterait le numéro de version mineur « .1 » au numéro de révision Debian ; le nouveau fichier de contrôle du paquet source serait alors `foo_1.1-3.1.dsc`.

Le numéro de révision mineur est nécessaire pour éviter de prendre un numéro de version au responsable officiel du paquet, ce qui pourrait perturber son travail. Cela a aussi l'avantage de montrer clairement que le paquet n'a pas été livré par le responsable officiel.

S'il n'y a pas de partie *révision-debian* dans le numéro de version du paquet, il faut en créer une en démarrant à « 0.1 ». S'il est absolument nécessaire qu'une personne qui n'est pas responsable d'un paquet fasse une livraison basée sur une nouvelle version amont, cette personne doit choisir « 0.1 » comme numéro de révision Debian. Le responsable du paquet doit, lui, démarrer sa numérotation à « 1 ».

Si vous envoyez un paquet vers *testing* ou *stable*, vous devrez parfois créer une branche (« fork ») dans l'arbre de numéro des version. Pour cela, vous pouvez utiliser des numéros de version comme 1.1-3sarge0.1.

5.11.3 Les mises à jour indépendantes sources doivent être mentionnées dans le fichier changelog

Une personne qui fait une mise à jour indépendante source doit ajouter une entrée dans le fichier `changelog` qui indique les bogues corrigés et qui précise pourquoi cette mise à jour était nécessaire. Cette entrée comportera l'adresse de la personne ayant fait l'envoi ainsi que la version livrée.

Par convention, dans le cas d'une mise à jour indépendante source *(NMU)*, l'entrée du fichier changelog débute par la ligne :

```
* Non-maintainer upload
```

5.11.4 Mise à jour indépendante source et système de suivi des bogues

Un développeur qui n'est pas responsable d'un paquet doit faire aussi peu de modifications que possible et doit toujours envoyer ses modifications au système de suivi des bogues au format diff unifié (`diff -u`).

Et si vous recompilez simplement le paquet ? Si vous avez simplement besoin de recompiler le paquet pour une seule architecture, vous pouvez faire une NMU binaire seulement comme décrit dans 'Mises à jour indépendantes binaires ou recompilations' page 55 qui ne nécessite pas qu'un correctif soit envoyé. Si vous désirez que le paquet soit recompilé pour toutes les architectures, vous devez alors faire une NMU source et vous devrez envoyer un correctif.

Si la mise à jour indépendante source (*source NMU*) corrige des bogues, ceux-ci doivent être marqués *fixed* (corrigé) dans le système de suivi des bogues plutôt que clos. Par convention, seul le responsable du paquet et la personne qui a ouvert le rapport de bogue ferment les rapports. Heureusement, le système d'archivage Debian reconnaît les mises à jours indépendantes et positionne correctement le statut des bogues à *fixed* si la personne qui fait la mise à jour a listé tous les bogues dans le fichier changelog en utilisant la syntaxe `Closes: bug#nnnnn` (voir 'Quand les rapports de bogue sont-ils fermés par une mise à jour ?' page 45 pour en savoir plus sur la fermeture de bogue par le fichier `changelog`). Ce passage au statut *fixed* assure que chacun sait que le bogue est corrigé par une mise à jour indépendante tout en laissant le rapport de bogue ouvert jusqu'à ce que le responsable du paquet incorpore les modifications de cette mise à jour dans la version officielle du paquet.

Après avoir fait une mise à jour indépendante, il vous faudra aussi envoyer l'information aux bogues existants que vous avez corrigés par votre NMU en incluant le diff unifié. Historiquement, c'était une habitude de créer un nouveau rapport de bogue et inclure un correctif comprenant toutes les modifications que vous avez réalisées. Le responsable officiel pourra choisir d'appliquer le correctif, il pourra aussi employer une autre méthode pour régler le problème. Certains bogues sont corrigés dans la version amont, ce qui est une bonne raison pour annuler les modifications d'une mise à jour indépendante. Si le responsable choisit de mettre à jour le paquet plutôt que d'utiliser les correctifs de la mise à jour indépendante, il devra s'assurer que cette nouvelle version corrige effectivement chacun des bogues corrigés dans la mise à jour indépendante.

De plus, le responsable officiel devrait *toujours* conserver les entrées documentant une mise à jour indépendante dans le fichier `changelog` — et bien sûr, également conserver les modifications. Si vous annulez certaines des modifications, veuillez réouvrir les rapports de bogue correspondants.

5.11.5 Fabriquer une mise à jour indépendante source

Les paquets faisant l'objet d'une mise à jour indépendante source sont construits comme les autres. Sélectionnez une distribution en utilisant les règles décrites dans la section 'Choisir une distribution' page 38 en suivant toutes les instructions de la section 'Mettre à jour un paquet' page 39.

Vérifiez que vous n'avez pas modifié la valeur du champ `maintainer` dans le fichier `debian/control`. Votre nom, mentionné dans l'entrée du fichier `debian/changelog` concernant la mise à jour, sera utilisé pour signer le fichier `.changes`.

5.11.6 Valider une mise à jour indépendante

Si l'un de vos paquets a subi une mise à jour indépendante, vous devez récupérer les changements dans votre copie des sources. Ceci est aisé, vous avez simplement à appliquer le correctif qui vous a été envoyé. Une fois ceci fait, vous devez fermer les bogues qui ont été marqués comme fixés par la mise à jour. Le moyen le plus simple est d'utiliser l'option `-v` de `dpkg-buildpackage` car cela vous permet d'inclure tous les changements depuis votre dernier envoi de responsable. Sinon, vous pouvez soit les fermer manuellement en envoyant les courriers nécessaires au BTS soit ajouter les `closes: #nnnn` nécessaires dans l'entrée du changelog de votre prochain envoi.

Dans tous les cas, vous ne devriez pas être perturbé par la NMU. Une NMU n'est pas une attaque personnelle contre le responsable. C'est une preuve que le paquet est important pour quelqu'un et qu'il est désireux de vous aider dans votre travail, vous devriez donc lui être reconnaissant. Vous pouvez également lui demander s'il serait intéressé pour vous aider sur une base plus régulière comme co-responsable ou responsable de secours (cf. 'Maintenance collective' page suivante).

5.11.7 Mise à jour indépendante ou envoi de QA ?

Sauf si vous savez que le responsable est toujours actif, il est sage de vérifier le paquet pour voir s'il n'a pas été abandonné. La liste actuelle des paquets orphelins pour lesquels le champ responsable n'a pas été positionné correctement est disponible à `http://qa.debian.org/orphaned.html`. Si vous effectuez une mise à jour indépendante sur un paquet incorrectement orphelin, veuillez positionner le responsable à « Debian QA Group <packages@qa.debian.org> ».

5.11.8 Qui peut faire une mise à jour indépendante ?

Seuls les responsables Debian officiels peuvent faire des mises à jour indépendantes binaire ou source. Un responsable officiel est une personne dont la clé est dans le porte-clés Debian. Tout autre personne est toutefois invitée à télécharger les paquets sources pour corriger des bogues ; au lieu de faire des mises à jour indépendantes, ils pourront soumettre les correctifs qui le méritent au système de suivi des bogues. Les responsables apprécient presque toujours les correctifs et les rapports de bogue soignés.

5.11.9 Terminologie

Deux nouvelles expressions sont introduites dans cette section : « mise à jour indépendante source » et « mise à jour indépendante binaire ». Ces deux expressions ont une signification

technique précise dans ce document. Elles correspondent toutes deux au même type d'activité ; elles impliquent toutes deux qu'une personne fait une mise à jour d'un paquet alors qu'elle n'est pas officiellement responsable de ce paquet. C'est pourquoi nous qualifions ces mises à jours d'*indépendantes*[8].

Une mise à jour indépendante source est une livraison de paquet faite par une personne qui n'est pas le responsable officiel de ce paquet avec pour objectif de corriger un bogue dans le paquet. Une mise à jour indépendante source implique toujours une modification des sources du paquet, même s'il ne s'agit que d'un changement dans le fichier `debian/changelog`. Ce changement peut tout aussi bien concerner la partie amont du source que la partie spécifique à Debian. Une mise à jour indépendante source peut aussi inclure des paquets spécifiques à une architecture tout comme un fichier *diff* modifié.

Une mise à jour indépendante binaire est constituée par la recompilation et l'archivage d'un paquet pour une architecture donnée. Il s'agit souvent du résultat d'un effort de portage. Une mise à jour indépendante binaire est la livraison d'un paquet compilé (souvent pour une autre architecture) à condition que cette compilation n'ait pas nécessité de modifications des sources. Dans de nombreux cas, les porteurs sont obligés de modifier les sources pour les rendre compilables sur leur architecture cible ; il s'agira alors d'une mise à jour indépendante source et non d'une mise à jour indépendante binaire. Comme vous pouvez le remarquer, nous ne faisons pas de distinction entre les mises à jour indépendantes faites par des porteurs et les autres mises à jour indépendantes.

Les mises à jour indépendantes sources et binaires sont toutes deux couvertes par l'expression « mise à jour indépendante » (NMU[9]). Pourtant, cela conduit souvent à des confusions car beaucoup associent « mise à jour indépendante » et « mise à jour indépendante source ». Il faut donc rester vigilant : utilisez toujours « mise à jour indépendante binaire » ou «NMU binaire » pour les mises à jour indépendantes de binaires seuls.

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 petit. 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.

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

[8] Contrairement à ce que pourrait laisser entendre cette traduction de *non-maintainer upload*, il n'est pas question d'agir sans prévenir le responsable au préalable (voir 'Comment faire une mise à jour indépendante ?' page 59).

[9] Non-maintainer upload

– Donner au co-responsable un accès aux sources à partir desquelles vous construisez le paquet. Habituellement, cela implique que vous utilisiez un système de contrôle de version comme `CVS` ou `Subversion`.
– Ajouter les nom et adresse correctes du co-responsable au champ `Uploaders` dans la partie globale du fichier `debian/control`.

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

– En utilisant le PTS (‘Système de suivi des paquets’ page 28), les co-responsables devraient s’inscrire eux-mêmes aux paquets sources appropriés.

La maintenance collective peut souvent être facilitée par l’utilisation d’outils sur Alioth (voir ‘Debian *Forge : Alioth’ page 32).

5.13 La distribution *testing*

5.13.1 Bases

Les paquets sont habituellement installés dans la distribution *testing* après avoir atteint un certain degré de test 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 n’avoir aucun bogue bloquant l’inclusion du paquet dans une version stable (« release-critical ») au moment où ils sont installés dans *testing*. Ainsi, *testing* devrait toujours être prête pour être une version candidate stable. Veuillez voir ci-dessous pour les détails.

5.13.2 Mises à jour depuis *unstable*

Les scripts qui mettent à jour la distribution *testing* sont exécutés chaque jour 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 2, 5 ou 10 jours selon le champ d’urgence de l’envoi (élevée, moyenne ou basse). Veuillez noter que cette urgence est « collante » (« sticky »), ce qui veut dire que l’envoi avec l’urgence la plus élevée depuis la précédente transition dans *testing* est prise en compte. Ces délais peuvent être doublés lors d’un gel de distribution, ou les transitions dans *testing* peuvent être complètement désactivées ;
– Il doit avoir autant ou moins de bogues empêchant l’intégration dans la distribution que la version actuellement disponible dans *testing* ;
– Il doit être disponible pour toutes les architectures pour lesquelles il a été auparavant construit. ‘L’outil `madison`’ page 27 peut être intéressant pour vérifier cette information ;
– Il ne doit pas casser les dépendances d’un paquet qui est 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).

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 (http://www.debian.org/devel/testing) ou utilisez le programme `grep-excuses` inclus dans le paquet `devscripts`. Si vous voulez rester informé de la progression de vos paquets dans *testing*, vous pouvez facilement le mettre dans la `crontab(5)`.

Le fichier `update_excuses` ne donne pas toujours la raison précise pour laquelle un paquet est refusé ; on peut avoir à la chercher soi-même en regardant ce qui serait cassé avec l'inclusion du paquet. La page web de la distribution testing (http://www.debian.org/devel/testing) donne plus d'informations à propos des problèmes courants qui peuvent 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://bjorn.haxx.se/debian/ — mais, attention, cette page affiche également les dépendances de construction qui ne sont pas considérées par britney.

Désynchronisation

Pour le script de migration dans *testing*, « désynchronisé » (*outdated* veut dire ceci : il y a différentes versions dans *unstable* pour les architectures de version (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 :

```
foo      | alpha | arm
---------+-------+----
testing  |   1   |  -
unstable |   1   |  2
```

Le paquet est désynchronisé pour alpha dans *unstable* et n'entrera pas dans *testing*. Supprimer foo 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) :

```
foo      | 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 version).

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

Suppressions 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. Considérons, par exemple, qu'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 RC est suffisant pour être dans cet état).

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

Dépendances circulaires

Une situation qui n'est pas très bien gérée par britney est si un paquet *a* dépend de la nouvelle version d'un paquet *b* et vice versa.

Voici un exemple :

```
  | testing          |  unstable
--+------------------+------------
a | 1; dépend: b=1 |  2; dépend: b=2
b | 1; dépend: a=1 |  2; dépend: 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 en envoyant un courrier électronique à debian-release@lists.debian.org si cela se produit pour l'un de vos paquets.

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 RC du paquet est réduit, le paquet peut migrer même s'il a encore des bogues RC. 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 veut dire : la seule influence qu'un paquet de *testing* a sur la nouvelle version du même paquet est que la nouvelle version peut rentrer plus facilement.

Détails

Si vous êtes intéressé par les détails, voici comment fonctionne britney :

Les paquets sont examinés pour savoir si ce sont des candidats valides. Cela donne le fichier « update excuses ». Les raisons les plus communes pour lesquelles un paquet n'est pas considéré sont la jeunesse du paquet, le nombre de bogues RC et la désynchronisation pour certaines architectures. Pour cette partie de britney, les responsables de version ont des marteaux de toute taille pour forcer britney à considérer un paquet. (Le gel de la base est également codé dans cette partie de britney.) (Il y a une chose semblable pour les mises à jour binaires pures, mais cela n'est pas décrit ici. Si vous êtes intéressé par cela, veuillez étudier attentivement le code.)

Maintenant, la partie la plus complexe se produit : britney tente de mettre à jour *testing* avec des candidats valides ; en premier, chaque paquet individuellement, puis des groupes de plus en plus larges de paquets ensemble. Chaque tentative est acceptée si *testing* n'est pas moins non installable après la mise à jour qu'avant celle-ci. (Avant et après cette partie, certains coups de pouce (« hints ») sont traités ; mais, comme seuls les responsables de version peuvent positionner des coups de pouce, cela n'est probablement pas très important pour vous.)

Si vous voulez voir plus de détails, vous pouvez le voir dans merkel :/org/ftp.debian.org/testing/update_out/ (ou dans ~aba/testing/update_out pour voir une configuration avec un fichier de paquets plus petit). Par le web, c'est à `http://ftp-master.debian.org/testing/update_out_code/`.

Les coups de pouce sont visibles sur `http://ftp-master.debian.org/testing/hints/`.

5.13.3 Mises à jour directes dans *testing*

La distribution *testing* est peuplée avec des 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 du responsable 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 distribution, vous devriez lire les instructions données qu'ils envoient régulièrement sur `<debian-devel-announce@lists.debian.org>`.

Vous ne devriez pas envoyer un paquet à *testing-proposed-updates* quand vous pouvez le mettre à jour par *unstable*. Si vous ne pouvez faire autrement (par exemple, parce que vous avez une nouvelle version de développement dans *unstable*), vous pouvez utiliser cette facilité, mais il

est recommandé de demander l'autorisation du responsable de distribution 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 1.2sarge1 pour le premier envoi dans *testing-proposed-updates* du paquet en version 1.2.

Veuillez vous assurer que vous n'oubliez aucun des éléments suivants lors de votre envoi :

- vérifiez que votre paquet doit vraiment aller dans *testing-proposed-updates* et ne peut pas passer par *unstable* ;
- vérifiez que vous n'incluez que le minimum de changements ;
- vérifiez que vous incluez une explication appropriée dans le changelog ;
- vérifiez que vous avez bien indiqué *testing* ou *testing-proposed-updates* comme votre distribution cible ;
- vérifiez que vous avez construit et testé votre paquet dans *testing* et non dans *unstable* ;
- vérifiez que votre numéro de version est plus élevé que les versions de *testing* et de *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 Questions fréquemment posées

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 critiques, graves et sérieux.

Certains bogues sont supposés avoir un impact sur les chances que le paquet a d'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 compte des bogues d'*unstable* est effectué avec tous les bogues bloquants sans étiquette de version (comme *potato*, *woody*) ou avec une étiquette *sid* et également s'ils ne sont ni corrigés ou marqués avec *sarge-ignore*. Le compte des bogues de *testing* pour un paquet est considéré comme à peu près le nombre de bogues d'*unstable* lors du dernier pointage quand la version *testing* a été égale à la version *unstable*.

Cela changera après la sortie de *Sarge* dès que nous aurons des versions dans le système de suivi des bogues.

Comment est-ce que l'installation d'un paquet dans *testing* peut 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 (par suite, principalement des bibliothèques). Cependant, cela va aussi toucher des paquets sur lesquels une dépendance versionnée a été déclarée du type ==, <= ou «.

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 de la nouvelle version à la place. Comme l'installation d'un tel paquet source dans *testing* casse tous les paquets qui en dépendent dans *testing*, une certaine attention doit y être portée : tous les paquets en dépendant doivent être mis à jour et prêts à être installés eux-même pour qu'ils ne cassent pas et, une fois que tout est prêt, une intervention manuelle du responsable de version ou d'un de ses assistants est normalement requise.

Si vous avez des problèmes avec des groupes compliqués de paquets comme ceci, contactez debian-devel ou debian-release en demandant de l'aide.

Chapitre 6

Les meilleurs pratiques pour la construction des paquets

La qualité de Debian est principalement due à la charte Debian (http://www.debian.org/doc/debian-policy/) qui définit explicitement les obligations que tous les paquets doivent suivre. Mais c'est également grâce à une histoire partagée d'expériences qui va au-delà de la charte Debian, une accumulation d'années d'expérience pour la construction des paquets ; des développeurs de grand talent ont créé de bons outils qui vous aideront, vous, responsable Debian, à créer et maintenir d'excellents paquets.

Ce chapitre fournit les meilleurs pratiques pour les développeurs Debian. Ce ne sont que des recommandations, non pas des obligations ou des règles. Ce sont seulement des astuces et conseils subjectifs et des liens collectés pour les développeurs Debian. Prenez et choisissez ce qui vous convient le mieux.

6.1 Les meilleures pratiques pour le fichier `debian/rules`

Les recommandations suivantes s'appliquent au fichier `debian/rules`. Comme ce fichier contrôle le processus de compilation et qu'il sélectionne les fichiers inclus dans le paquet (directement ou indirectement), il s'agit habituellement du fichier sur lequel les responsables passent le plus de temps.

6.1.1 Scripts d'aide

La raison sous-jacente à l'utilisation des scripts d'aide dans le fichier `debian/rules` est que cela permet aux responsables d'utiliser et de partager une logique commune pour un grand nombre de paquets. Prenez, par exemple, la question de l'installation des entrées de menu : vous avez besoin de placer un fichier dans `/usr/share/menu` (ou `/usr/lib/menu` pour des fichiers de menu binaires exécutables, si nécessaire) et d'ajouter des commandes aux scripts de maintenance pour enregistrer et désenregistrer ces entrées de menu. Comme il s'agit d'une

chose très commune, pourquoi chaque responsable devrait-il écrire sa propre version, parfois avec des bogues ? Supposons également que le répertoire de menu soit modifié, chaque paquet devrait être modifié.

Les scripts d'aide peuvent résoudre ces problèmes. En supposant que vous vous conformiez aux conventions attendues par le script d'aide, celui-ci prend soin de tous les détails. Les changements dans la charte peuvent alors être faits dans le script d'aide ; les paquets ont alors simplement besoin d'être reconstruits avec la nouvelle version du script et sans aucun changement supplémentaire.

L''Aperçu des outils du responsable Debian' page 107 contient plusieurs systèmes d'aide. Le plus courant et le meilleur (à notre avis) système est `debhelper`. Les précédents systèmes d'aide comme `debmake` étaient « monolithiques » : vous ne pouviez pas choisir quelles parties intéressantes vous pouviez utiliser, mais vous étiez obligé de choisir le système pour tout faire. `debhelper`, à l'inverse, consiste en un certain nombre de petits programmes `dh_*`. Par exemple, `dh_installman` installe et compacte les pages de manuel, `dh_installmenu` installe les fichiers de menu et ainsi de suite. Ainsi, il offre une flexibilité suffisante pour pouvoir utiliser les scripts d'aide quand ils sont utiles, en complément de commandes définies manuellement dans le fichier `debian/rules`.

Vous pouvez débuter avec `debhelper` en lisant `debhelper(1)` et en regardant les exemples fournis avec le paquet. `dh_make` du paquet `dh-make` (voir 'dh-make' page 110) peut être utilisé pour convertir un paquet source « vierge » en une version utilisant `debhelper`. Ce raccourci ne devrait cependant pas vous faire croire que vous n'avez pas besoin de comprendre les scripts individuels `dh_*`. Si vous comptez utiliser un système d'aide, vous devez prendre le temps d'apprendre à utiliser ce système pour savoir ce que vous pouvez en attendre ainsi que son comportement.

Plusieurs personnes pensent que des fichiers `debian/rules` vierges sont préférables car vous n'avez pas à apprendre les détails internes d'un quelconque système d'aide. La décision vous appartient complètement. Utilisez ce qui vous convient. Plusieurs exemples de fichiers `debian/rules` sont disponibles à `http://arch.debian.org/arch/private/srivasta/`.

6.1.2 Séparer vos correctifs en plusieurs fichiers

Les gros paquets peuvent avoir plusieurs bogues que vous devez gérer. Si vous corrigez sans faire attention les bogues dans les sources, vous ne pourrez pas différencier facilement les nombreux correctifs que vous aurez appliqués. Cela peut devenir très compliqué de mettre à jour le paquet avec une nouvelle version amont qui intègre certains correctifs (mais pas tous). Vous ne pouvez pas prendre l'ensemble des correctifs (par exemple, dans les fichiers `.diff.gz`) et décider quels correctifs il vous faut enlever parce que les bogues ont été corrigés en amont.

Malheureusement, le système de création des paquets tel qu'il est actuellement ne fournit pas de moyen de séparer les correctifs en plusieurs fichiers. Cependant, il existe des moyens de séparer les correctifs : les fichiers correctifs sont livrés dans le fichier correctif Debian (`.diff.gz`), habituellement dans le répertoire `debian/`. La seule différence est qu'ils ne sont pas appliqués immédiatement par dpkg-source, mais par la règle `build` du fichier `debian/rules`. Inversement, ils sont annulés par la règle `clean`.

`dbs` est l'une des approches les plus populaires pour cela. Il fait ce qui est décrit ci-dessus et fournit la possibilité de créer de nouveaux correctifs et de mettre à jour d'anciens correctifs. Veuillez vous reporter au paquet `dbs` pour plus d'informations et au paquet `hello-dbs` pour un exemple.

`Dpatch` fournit également ces fonctions, mais il est prévu pour être encore plus facile d'utilisation. Veuillez voir le paquet `dpatch` pour la documentation et des exemples (dans `/usr/share/doc/dpatch`).

6.1.3 Paquets binaires multiples

Un simple paquet source va souvent permettre de construire plusieurs paquets binaires différents, que ce soit pour fournir plusieurs saveurs du même paquet (par exemple, le paquet source `vim`) ou pour créer plusieurs petits paquets au lieu d'un seul gros (afin, par exemple, que l'utilisateur puisse n'installer que la partie nécessaire et ainsi économiser de l'espace disque).

Le second cas peut facilement être géré dans le fichier `debian/rules`. Vous avez simplement besoin de déplacer les fichiers appropriés du répertoire de construction dans les arborescence temporaires du paquet. Vous pouvez le faire en utilisant `install` ou `dh_install` du paquet `debhelper`. Assurez-vous de vérifier les différentes permutations de paquets, en garantissant que vous avez bien défini les dépendances entre les paquets dans le fichier `debian/control`.

Le premier cas est un peu plus difficile car il implique de multiples recompilations du même logiciel, mais avec différentes options de configuration. Le paquet source `vim` est un exemple de la façon de gérer cela avec un fichier `rules` écrit à la main.

6.2 Les meilleures pratiques pour le fichier `debian/control`

Les pratiques suivantes sont relatives au fichier `debian/control`. Elles viennent en complément des règles pour les descriptions des paquets (`http://www.debian.org/doc/debian-policy/ch-binary.html#s-descriptions`).

La description du paquet, telle qu'elle est définie par le champ correspondant dans le fichier `control`, contient à la fois le résumé du paquet et la description longue pour le paquet. 'Les règles générales pour les descriptions des paquets' de la présente page décrit des lignes générales pour les deux parties de la description. Ensuite, 'Le résumé du paquet ou description courte' page suivante fournit des principes spécifiques pour le résumé et 'La description longue' page 75 contient des principes spécifiques pour la description longue.

6.2.1 Les règles générales pour les descriptions des paquets

La description du paquet devrait être écrite pour l'utilisateur moyen, l'utilisateur qui va utiliser et bénéficier du paquet. Par exemple, les paquets de développement sont pour les développeurs et leur description peut utiliser un langage technique. Pour les applications à but plus général comme un éditeur, la description devrait être écrite pour un non-spécialiste.

Notre critique des descriptions des paquets nous amène à conclure que la plupart des descriptions des paquets sont techniques, c'est-à-dire, qu'elles ne sont pas écrites pour être comprises par les non-spécialistes. À moins que votre paquet ne soit que pour les techniciens, c'est un problème.

Comment écrire pour les non-spécialistes ? Évitez le jargon. Évitez de vous référer à d'autres applications et cadres de travail avec lesquels l'utilisateur n'est pas forcément familier — « GNOME » ou « KDE » sont corrects car les utilisateurs sont probablement familiers avec ces termes, mais « GTK+ » ne l'est probablement pas. Ne supposez aucune connaissance. Si vous devez utiliser des termes techniques, introduisez-les.

Soyez objectif. Les descriptions de paquet ne sont pas un endroit pour promouvoir votre paquet, quel que soit l'amour que vous lui portez. Rappelez-vous que le lecteur n'a pas forcément les mêmes priorités que vous.

Des références aux noms de tout autre paquet de logiciels, noms de protocoles, standards ou spécifications devraient utiliser leurs formes canoniques si elles existent. Par exemple, utilisez « X Window System », « X11 » ou « X » et non « X Windows », « X-Windows » ou « X Window ». Utilisez « GTK+ » et non « GTK » ou « gtk ». Utilisez « GNOME » et non « Gnome ». Utilisez « PostScript » et non « Postscript » ou « postscript ».

Si vous avez des problèmes pour écrire votre description, vous pouvez l'envoyer à `<debian-l10n-english@lists.debian.org>` et demander un retour d'informations.

6.2.2 Le résumé du paquet ou description courte

La ligne de résumé (la description courte) devrait être concise. Elle ne doit pas répéter le nom du paquet (c'est une règle).

C'est une bonne idée de penser au résumé comme à une clause apposée et non une phrase complète. Une clause apposée est définie dans WordNet comme une relation grammaticale entre un mot et une phrase pronominale qui la suit, par exemple « Rudolph, le renne au nez rouge ». La clause apposée ici est « le renne au nez rouge ». Comme le résumé est une clause et non une phrase complète, nous recommandons de ne pas la commencer par une majuscule et de ne pas la finir par un point. Il ne doit pas non plus commencer avec un article, défini (« le ») ou indéfini (« un »).

Cela peut vous aider d'imaginer le résumé combiné avec le nom du paquet de la façon suivante :

```
nom-paquet est un résumé.
```

Sinon, il peut être plus compréhensible de le voir comme

```
nom-paquet est résumé.
```

ou, si le nom du paquet est lui-même un pluriel (comme « developer-tools »)

```
nom-paquet sont résumé.
```

Cette façon de former une phrase à partir du nom du paquet et du résumé devrait être considérée comme une heuristique et non comme une règle stricte. Il y a certains cas où cela n'a aucun sens de former une phrase.

6.2.3 La description longue

La description longue du paquet est la première information dont dispose l'utilisateur avant d'installer un paquet. Aussi, elle devrait fournir toutes les informations nécessaires pour le laisser décider de l'installation du paquet. Vous pouvez supposer que l'utilisateur a déjà lu le résumé du paquet.

La description longue devrait toujours être constituée de phrases complètes.

Le premier paragraphe de la description longue devrait répondre aux questions suivantes : qu'est-ce que fait le paquet ? Quelle tâche aide-t-il l'utilisateur à accomplir ? Il est important de décrire ceci d'une manière non technique, à moins que le paquet ne s'adresse qu'à un auditoire de techniciens.

Les paragraphes suivants devraient répondre aux questions suivantes : Pourquoi, en tant qu'utilisateur, ai-je besoin de ce paquet ? Quelles sont les autres fonctionnalités dont dispose le paquet ? Quelles sont les fonctionnalités marquantes et les déficiences de ce paquet comparé à d'autres paquets (par exemple, « si vous avez besoin de X, utilisez Y à la place ») ? Est-ce que le paquet est lié à d'autres paquets d'une certaine façon qui n'est pas gérée par le gestionnaire de paquet (par exemple, « il s'agit d'un client pour le serveur foo ») ?

Soyez attentif à éviter les fautes d'orthographe et de grammaire. N'oubliez pas votre vérificateur orthographique. Les deux programmes `ispell` et `aspell` possèdent des modes spéciaux pour vérifier les fichiers `debian/control` :

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

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

Les utilisateurs s'attendent habituellement à ce que les réponses aux questions suivantes soient présentes dans la description du paquet :

- Qu'est-ce que fait le paquet ? Si c'est un ajout d'un autre paquet, la description courte du paquet dont c'est un ajout devrait le spécifier.
- Pourquoi est-ce qu'il voudrait installer ce paquet ? Ceci est lié à ce qui est ci-dessus, mais pas tout à fait (c'est un client de messagerie ; il est cool, rapide, s'interface avec PGP, LDAP et IMAP, a les fonctionnalités X, Y et Z).
- Si ce paquet ne devrait pas être installé directement, mais être tiré par un autre paquet, cela devrait être mentionné.
- Si le paquet est expérimental, ou s'il y a d'autres raisons pour lesquelles il ne devrait pas être utilisé, si un autre paquet devrait être utilisé à la place, cela devrait également être présent.

– En quoi le paquet est différent des paquets concurrents ? Est-ce que c'est une meilleure implémentation ? A-t-il plus de fonctionnalités, des fonctionnalités différentes ? Pourquoi devrait-il choisir ce paquet ?

6.2.4 Page d'accueil amont

Nous vous recommandons d'ajouter l'adresse de la page d'accueil du paquet à la description du paquet dans le fichier `debian/control`. Cette information devrait être ajoutée à la fin de la description en utilisant le format suivant :

```
.
 Homepage: http://some-project.some-place.org/
```

Veuillez noter les espaces au début de la ligne, ils servent à séparer les lignes correctement. Pour voir un exemple de ce que cela affiche, veuillez vous reporter à `http://packages.debian.org/unstable/web/wml`.

Ne mettez rien s'il n'existe pas de page pour le logiciel.

Veuillez noter que nous espérons que ce champ sera remplacé par un vrai champ de `debian/control` qui serait compris par `dpkg` et `packages.debian.org`. Si vous ne voulez pas vous embêter à déplacer la page d'accueil depuis la description vers ce nouveau champ, vous devriez probablement attendre qu'il soit disponible. Veuillez vous assurer que cette ligne correspond à l'expression rationnelle `/^ Homepage: [^ ]*$/` car cela permet à `packages.debian.org` de l'analyser correctement.

6.3 Les meilleures pratiques pour le fichier `debian/changelog`

Les pratiques suivantes viennent en complément de la directive sur les fichiers changelog (`http://www.debian.org/doc/debian-policy/ch-docs.html#s-changelogs`).

6.3.1 Écrire des entrées de changelog utiles

L'entrée de changelog pour une révision de paquet documente les changements dans cette révision et seulement ceux-ci. Concentrez-vous sur la description des changements significatifs et visibles de l'utilisateur qui ont été effectués depuis la dernière version.

Ciblez *ce* qui a été changé — qui, comment et quand cela a été fait est généralement de moindre importance. Ceci dit, rappelez-vous de nommer poliment les personnes qui ont fourni une aide notable pour réaliser le paquet (par exemple, ceux qui ont envoyé des correctifs).

Vous n'avez pas besoin de détailler les changements triviaux et évidents. Vous pouvez également regrouper plusieurs de ces changements dans une seule entrée. D'un autre côté, ne soyez pas trop concis si vous avez entrepris un changement majeur. Soyez tout spécialement clair s'il

y a des changements qui modifient le comportement du programme. Pour plus d'explications, utilisez le fichier `README.Debian`.

Utilisez un langage anglais commun pour que la majorité des lecteur puissent le comprendre. Évitez les abréviations, le parler technique et le jargon quand vous expliquez des changements fermant un bogue, spécialement pour les rapports de bogue créés par des utilisateurs qui ne vous paraissent pas particulièrement à l'aise techniquement. Vous devez être poli et ne pas jurer.

Il est parfois désirable de préfixer les entrées de changelog avec le nom des fichiers qui ont été modifiés. Cependant, il n'est pas besoin de lister explicitement tous les fichiers modifiés, particulièrement si la modification est petite ou répétitive. Vous pouvez utiliser les caractères génériques.

Quand vous faites référence aux bogues, ne supposez rien a priori. Dites ce qu'était le problème, comment il a été résolu et ajoutez la chaîne de caractères « closes : #nnnnn ». Veuillez voir 'Quand les rapports de bogue sont-ils fermés par une mise à jour ?' page 45 pour plus d'informations.

6.3.2 idées fausses communes sur les entrées de changelog

Les entrées de changelog **ne devraient pas** documenter des problèmes génériques d'empaquetage (« Hé, si vous cherchez foo.conf, il est dans /etc/blah/. ») car les administrateurs et utilisateurs sont supposés être au moins vaguement rompus à la façon dont les choses sont arrangées sur un système Debian. Mentionnez cependant tout changement d'emplacement d'un fichier de configuration.

Les seuls bogues fermés par une entrée de changelog devraient être ceux qui sont vraiment corrigés dans la même révision du paquet. Fermer des bogues non liés par le fichier changelog est considéré comme une très mauvaise pratique. Veuillez voir 'Quand les rapports de bogue sont-ils fermés par une mise à jour ?' page 45.

Les entrées de changelog **ne devraient pas** être le lieu de discussions avec les émetteurs de bogues (« Je ne vois pas de segfaults lors du lancement de foo avec l'option bar ; envoyez-moi plus d'informations. »), ni celui de phrases génériques sur la vie, l'univers et tout le reste (« Désolé, cet envoi m'a pris du temps, mais j'avais attrapé la grippe ») ou celui de demandes d'aide (« La liste des bogues sur ce paquet est énorme, donnez-moi un coup de main »). Ceci ne sera généralement pas remarqué par les personnes ciblées, mais peut ennuyer les personnes qui désirent lire des informations sur les changements réels du paquet. Veuillez vous reporter à 'Répondre à des rapports de bogue' page 42 pour plus d'informations sur la façon d'utiliser le système de suivi des bogues.

C'est une vieille tradition de valider les bogues fixés par une mise à jour indépendante dans la première entrée du changelog de l'envoi du vrai responsable. Comme nous avons maintenant le suivi des versions, il est suffisant de garder les entrées de changelog des mises à jour indépendantes et de simplement mentionner ce fait dans votre propre entrée de changelog.

6.3.3 Erreurs communes dans les entrées de changelogs

Les exemples suivants montrent des erreurs communes ou des exemples de mauvais style dans les entrées de changelog[1].

```
* Corrige tous les bogues restants.
```

Ceci n'indique visiblement rien d'utile au lecteur.

```
* Correctif de Jane Random appliqué.
```

Sur quoi portait le correctif ?

```
* Révision de cible d'installation la nuit dernière.
```

Qu'a accompli la révision ? Est-ce que la mention de la nuit dernière est supposée nous rappeler que nous ne devons pas faire confiance à ce code ?

```
* Corrige MRD vsync av. anciens CRTs.
```

Trop d'acronymes et il n'est pas très clair de ce qu'était vraiment cette... euh... merde (oups, un mot interdit !) ou comment cela a été corrigé.

```
* Ceci n'est pas un bogue. Closes: #nnnnnn.
```

Premièrement, il n'y a absolument pas besoin d'envoyer un paquet pour communiquer cette information ; à la place, utilisez le système de suivi des bogues. Deuxièmement, il n'y a aucune explication concernant la raison pour laquelle le rapport n'était pas un bogue.

```
* A été fixé il y a longtemps, mais j'ai oublié de le fermer. Closes: #5432
```

Si, pour toute raison, vous n'aviez pas indiqué le numéro du bogue dans une précédente entrée de changelog, ceci n'est pas un problème, fermez simplement le bogue normalement dans le BTS. Il n'y a pas besoin de modifier le fichier changelog, en supposant que la description de la correction est déjà intégrée (ceci s'applique aux correctifs par les auteurs/responsables amont également, vous n'avez pas à suivre les bogues qui ont été corrigés il y a longtemps dans votre changelog).

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

Où est la description ? ! Si vous n'arrivez pas à trouver un message descriptif, commencez par insérer le titre de chacun des différents bogues.

[1] NdT : Les entrées de changelog sont ici affichées en français pour faciliter la compréhension, mais vos entrées devront naturellement être rédigées en anglais.

6.3.4 Compléter les changelogs avec les fichiers NEWS.Debian

Les nouvelles importantes à propos des changements dans un paquet peuvent également être placées dans les fichiers NEWS.Debian. Ces nouvelles seront affichées par des outils comme apt-listchanges, avant le reste des changelogs. Ceci est le moyen préféré pour informer les utilisateurs des changements significatifs dans un paquet. Il est préférable d'utiliser ce fichier plutôt que d'utiliser des notes debconf car c'est moins ennuyeux et l'utilisateur peut y revenir et se référer au fichier NEWS.Debian après l'installation. Et c'est mieux que de lister les changements principaux dans README.Debian car l'utilisateur peut facilement rater de telles notes.

Le format du fichier est le même que pour un fichier de changelog Debian, mais il n'utilise pas d'astérisques et décrit chaque élément de nouvelle dans un paragraphe complet si nécessaire plutôt que les résumés concis qui iraient dans un changelog. C'est une bonne idée de passer votre fichier par dpkg-parsechangelog pour vérifier son formatage car il n'est pas vérifié automatiquement pendant la construction comme le changelog. Voici un exemple d'un vrai fichier NEWS.Debian :

```
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 comme /usr/share/doc/<package>/NEWS.Debian.gz. Il est compressé et a toujours ce nom même dans les paquets natifs Debian. Si vous utilisez debhelper, dh_installchangelogs installera les fichiers debian/NEWS pour vous.

À la différence des fichiers changelog, vous n'avez pas besoin de mettre à jour les fichiers NEWS.Debian à chaque nouvelle version. Ne les mettez à jour que si vous avez quelque chose de particulièrement important que l'utilisateur a besoin de savoir. Si vous n'avez pas de nouvelles du tout, il n'est pas nécessaire de fournir de fichier NEWS.Debian dans votre paquet. Pas de nouvelles, bonne nouvelle !

6.4 Les meilleures pratiques pour les scripts de maintenance

Les scripts de maintenance incluent les fichiers `debian/postinst`, `debian/preinst`, `debian/prerm` et `debian/postrm`. Ces scripts prennent en charge la configuration d'installation ou de désinstallation des paquets, ce qui n'est pas simplement créer ou supprimer des fichiers et des répertoires. Les instructions suivantes complètent la charte Debian (http://www.debian.org/doc/debian-policy/).

Les scripts de maintenance doivent être idempotents. Cela veut dire que vous devez vous assurer que rien de grave ne se produit si un script est appelé deux fois là où il ne devrait habituellement être appelé qu'une fois.

Les entrée et sortie standard peuvent être redirigées (par exemple, dans des tubes[2]) pour des besoins d'enregistrement d'activité, donc vous ne devez pas supposer que ce sont des tty.

Tous les affichages et les configurations interactives devraient être minimisées. Quand cela est nécessaire, vous devriez utiliser le paquet `debconf` pour l'interface. Rappelez-vous que, dans tous les cas, l'affichage ne doit se faire que dans l'étape de configuration, `configure` du script de post-installation, `postinst`.

Gardez les scripts de maintenance aussi simples que possible. Nous vous suggérons d'utiliser des scripts shell POSIX purs. Rappelez-vous, que si vous avez besoin d'une fonctionnalité de Bash, le script de maintenance doit préciser bash dans sa première ligne. Un shell POSIX ou Bash sont préférés à Perl car ils permettent à `debhelper` d'ajouter facilement des parties aux scripts.

Si vous modifiez les scripts de maintenance, assurez-vous de tester la suppression du paquet, la double installation et la purge complète. Assurez-vous qu'il ne reste rien d'un paquet purgé, c'est-à-dire, que la purge doit enlever tout fichier créé, directement ou indirectement, par les scripts de maintenance.

Si vous avez besoin de vérifier l'existence d'une commande, vous devriez utiliser quelque chose comme :

```
if [ -x /usr/sbin/install-docs ]; then ...
```

Si vous ne désirez pas mettre en dur le chemin d'une commande dans le script de maintenance, la fonction de shell suivante conforme à POSIX peut vous 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 le `$PATH` pour un nom de commande passé en paramètre. Il renvoie vrai (zéro) si la commande a été trouvée et faux sinon. Il s'agit réellement de la façon la plus portable de faire car `command -v`, `type` et `which` ne sont pas POSIX.

[2] pipes

Bien que `which` soit une alternative acceptable car il est présent dans le paquet classé *required* `debianutils`, il n'est pas présent dans la partition racine. Autrement dit, il est placé dans `/usr/bin` au lieu de `/bin`, il n'est donc pas possible de l'utiliser dans les scripts qui sont exécutés avant que la partition `/usr` soit montée. Cependant, la plupart des scripts n'auront pas ce problème.

6.5 Gestion de la configuration avec `debconf`

`Debconf` est un système de gestion de configuration qui peut être utilisé par les divers scripts de maintenance (principalement en post-installation dans le fichier `postinst`) pour demander à l'utilisateur des informations concernant la configuration du paquet. Il faut maintenant éviter les interactions directes avec l'utilisateur et préférer les interactions utilisant `debconf`. Ceci permettra à l'avenir des installations non interactives.

Debconf est un bon outil, mais il est souvent mal utilisé. Plusieurs erreurs communes sont référencées dans la page de manuel `debconf-devel(7)`. Vous devriez vraiment lire cette page si vous décidez d'utiliser debconf. Nous documentons également plusieurs des meilleures pratiques ici.

Ces lignes directives incluent plusieurs recommandations de style d'écriture et de typographie, des considérations générales à propos de l'utilisation de debconf ainsi que des recommandations plus spécifiques pour certaines parties de la distribution (par exemple, le système d'installation).

6.5.1 N'abusez pas de debconf

Depuis que debconf est apparu dans Debian, il a été largement abusé et plusieurs critiques reçues par la distribution Debian proviennent d'utilisation abusive de debconf avec la nécessité de répondre à un grand nombre de questions avant d'avoir n'importe quel petit logiciel d'installé.

Garder les notes d'utilisation à leur place : le fichier NEWS.Debian ou le fichier README.Debian. N'utilisez des notes que pour des notes importantes qui peuvent directement concerner l'utilisation du paquet. Rappelez-vous que les notes bloqueront toujours l'installation avant leur confirmation ou qu'elles embêtent l'utilisateur par un courriel.

Choisissez avec soin les priorités des questions dans les scripts de responsable. Reportez-vous à `debconf-devel(7)` pour plus de détails sur les priorités. La plupart des questions devraient utiliser un priorité moyenne ou basse.

6.5.2 Recommandations générales pour les auteurs et traducteurs

Écrivez un anglais correct

La plupart des responsables de paquets Debian n'ont pas l'anglais comme langue maternelle. Écrire des modèles correctement rédigés peut donc ne pas être facile pour eux.

Veuillez utiliser (et abuser de) la liste de discussions <debian-l10n-english@lists.debian.org>. Faites relire vos questionnaires.

Des questionnaires écrits incorrectement donnent une pauvre image de votre paquet, de votre travail… ou même de Debian elle-même.

Évitez le jargon technique autant que possible. Si certains termes vous semblent courants, ils peuvent être impossibles à expliquer à d'autres personnes. Si vous ne pouvez pas les éviter, essayez de les expliquer (en utilisant la description étendue). Quand vous faites cela, essayez d'équilibrer la verbosité et la simplicité.

Être courtois avec les traducteurs

Les questionnaires debconf peuvent être traduits. Debconf, avec son paquet-frère po-debconf, offre un cadre de travail simple pour obtenir des questionnaires traduits par les équipes de traduction ou même par des individus isolés.

Veuillez utiliser les questionnaires basés sur gettext. Installez po-debconf sur votre système de développement et lisez sa documentation (« man po-debconf » est un bon début).

Évitez de changer vos questionnaires trop souvent. Modifier le texte des questionnaires entraîne plus de travail pour les traducteurs dont les traductions seront rendues « floues » (« fuzzy »). Si vous prévoyez des modifications dans vos questionnaires d'origine, veuillez contacter les traducteurs. La plupart des traducteurs actifs sont très réactifs et obtenir leur travail inclus avec vos questionnaires modifiés vous économisera des envois supplémentaires. Si vous utilisez des modèles basés sur gettext, le nom et l'adresse électronique du traducteur sont mentionnés dans les en-têtes des fichiers PO.

L'utilisation de podebconf-report-po du paquet po-debconf est hautement recommandée pour avertir les traducteurs qui ont des traductions incomplètes et pour leur demander des mises à jour.

En cas de doutes, vous pouvez également contacter l'équipe de traduction pour une langue donnée (debian-l10n-xxxxx@lists.debian.org) ou la liste de discussions <debian-i18n@lists.debian.org>.

Les appels à traductions postés sur <debian-i18n@lists.debian.org> avec le fichier debian/po/templates.pot attaché ou référencé dans une URL sont encouragés. Assurez-vous de mentionner dans ces appels pour de nouvelles traductions quelles sont les langues pour lesquelles vous avez déjà des traductions existantes, pour éviter toute duplication de travail.

« Dé-fuzzy-fiez » les traductions complètes lors des corrections de typos et d'orthographe

Quand le texte d'un questionnaire debconf est corrigé et que vous êtes **certain** que les changements **n'ont aucun** effet sur les traductions, soyez courtois avec les traducteurs et dé-fuzzy-fiez leurs traductions.

Si vous ne le faites pas, le questionnaire entier ne sera pas traduit jusqu'à ce qu'un traducteur vous envoie une mise à jour.

Pour **dé-fuzzy-fier** les traductions, vous pouvez procéder ainsi :

1 enlevez tous les fichiers PO incomplets. Vous pouvez vérifier si les fichiers sont complets en utilisant (il faut que le paquet `gettext` soit installé) :

```
for i in debian/po/*po; do echo -n $i: ; msgfmt -o /dev/null
--statistics $i; done
```

2 déplacez tous les fichiers ayant des chaînes floues (« fuzzy ») dans un emplacement temporaire. Les fichiers n'ayant aucune chaîne floue (seulement des chaînes traduites et non traduites) seront conservées où ils sont placés,

3 maintenant **et seulement maintenant**, corrigez les typos dans le questionnaire et vérifiez que les traductions ne sont pas impactées (les typos, les fautes d'orthographe et parfois les corrections de typographie sont généralement acceptables),

4 exécutez `debconf-updatepo`. Cela va fuzzifier toutes les chaînes que vous avez modifiées dans les traductions. Vous pouvez constater cela en exécutant à nouveau la commande ci-dessus,

5 utilisez la commande suivante :

```
for i in debian/po/*po; do msgattrib --output-file=$i --clear-fuzzy $i;
```

6 déplacez dans debian/po les fichiers qui avaient des chaînes floues dans la première étape,

7 exécutez à nouveau `debconf-updatepo`.

Ne faites pas de suppositions à propos des interfaces

Le texte des modèles ne doit pas faire référence aux composants appartenant à l'une des interfaces debconf. Des phrases comme « If you answer Yes… » n'a aucun sens pour les utilisateurs d'interfaces graphiques qui utilisent des cases à cocher pour les questions booléennes.

Les modèles de chaînes de caractères devraient éviter de mentionner les valeurs par défaut dans leur description. Tout d'abord, parce que cela est redondant avec les valeurs lues par les utilisateurs. Ensuite, parce ces valeurs par défaut peuvent être différentes selon les choix du responsable (par exemple, quand la base de données debconf a été préremplie).

Plus généralement, essayez d'éviter de vous référer à toute action de l'utilisation. Donnez simplement des faits.

N'utilisez pas la première personne

Vous devriez éviter d'utiliser la première personne (« I will do this… » ou « We recommend… »). L'ordinateur n'est pas une personne et les questionnaires debconf ne parlent pas pour les développeurs Debian. Vous devriez utiliser une construction neutre et souvent une forme passive. Pour ceux d'entre vous qui écrivent déjà des publications scientifiques, écrivez simplement vos questionnaires comme vous écririez un papier scientifique.

Soyez neutre dans le genre

Le monde est fait d'hommes et de femmes. Veuillez utiliser des constructions neutres dans le genre dans votre texte. Ce n'est pas du politiquement correct, c'est simplement montrer du respect pour toute l'humanité.

6.5.3 Définition des champs de questionnaires

Cette partie donne plusieurs informations qui sont principalement extraites de la page de manuel `debconf-devel(7)`.

Type

string : (chaîne de caractères) Résulte en un champ d'entrée libre dans lequel l'utilisateur peut écrire toute chaîne.

password : (mot de passe) Demande un mot de passe à l'utilisateur. Veuillez utiliser ce champ avec précaution ; soyez conscient que le mot de passe que l'utilisateur entre sera écrit dans la base de données debconf. Vous devriez probablement enlever cette valeur de la base de données dès que possible.

boolean : (booléen) Un choix vrai/faux. Rappelez-vous : vrai/faux, **pas oui/non**...

select : (sélection) Un choix parmi un certain nombre de valeurs. Les choix doivent être spécifiés dans un champ nommé « Choices ». Séparez les valeurs possibles par des virgules et des espaces ainsi : Choices : yes, no, maybe

multiselect : (sélection multiple) Comme le type de données select, excepté que l'utilisateur peut choisir un nombre quelconque d'éléments dans la liste de choix (ou aucun d'entre eux).

note : (note) Plutôt que d'être une question en tant que telle, ce type de donnée indiqué une note qui peut être affichée à l'utilisateur. Il ne devrait être utilisé que pour des données importantes que l'utilisateur doit réellement voir, car debconf fera tout ce qu'il peut pour s'assurer que l'utilisateur le voit ; il stoppera l'installation en attendant que l'utilisateur appuie sur une touche ou il leur enverra même la note par courrier dans certains cas.

text : (texte) Ce type est maintenant considéré comme obsolète : ne l'utilisez pas.

error : (erreur) CE TYPE DE MODÈLE N'EST PAS ENCORE GÉRÉ PAR DEBCONF.

Il a été ajouté à cdebconf, la version C de debconf, utilisé en premier dans l'installateur Debian.

Veuillez ne pas l'utiliser à moins que debconf ne le prenne en charge.

Ce type est conçu pour gérer des messages d'erreur. Il est presque semblable au type « note ». Des interfaces peuvent le présenter différemment (par exemple, l'interface dialog de cdebconf affiche un écran rouge au lieu de l'écran bleu habituel).

Description : description courte et étendue

Les descriptions des modèles sont composées de deux parties : une courte et une étendue. La description courte est dans la ligne « Description : » du questionnaire.

La description courte devrait être gardée courte (50 caractères ou moins) pour qu'elle puisse être ajustée par la plupart des interfaces debconf. La garder courte aide également les traducteurs, car les traductions ont tendance à être plus longues que l'original.

La description courte devrait se suffire à elle-même. Certaines interfaces n'affichent pas la description longue par défaut ou seulement si l'utilisateur la demande explicitement ou même ne l'affichent pas du tout. Évitez des choses comme « What do you want to do ? ».

Il n'est pas nécessaire que la description courte soit une phrase complète. Cela fait partie de la recommandation « gardez-la courte et efficace ».

La description étendue ne devrait pas répéter la description courte mot pour mot. Si vous ne trouvez pas de description longue, commencez par à y réfléchir plus. Envoyez un message sur debian-devel. Demandez de l'aide. Suivez un cours d'écriture ! La description étendue est importante. Si, après tout cela, vous ne trouvez toujours rien, laissez-la vide.

La description étendue devrait utiliser des phrases complètes. Des paragraphes devraient être gardés court pour améliorer la lisibilité. Ne mélangez pas deux idées dans le même paragraphe, mais utilisez plutôt un autre paragraphe.

Ne soyez pas trop verbeux. Les utilisateurs ont tendance à ignorer les écrans trop longs. Par expérience, 20 lignes est la limite à éviter de dépasser car cela veut dire sinon que, dans l'interface dialogue classique, les utilisateurs devront faire défiler le texte et un grand nombre de personnes ne le font simplement pas.

Pour des règles spécifiques selon le type de questionnaire (chaîne de caractères, booléen, etc.), veuillez lire ci-dessous.

Choices (choix)

Ce champ devrait utilisé pour des types Select et Multiselect. Il contient les choix possibles qui seront présentés aux utilisateurs. Ces choix devraient être séparés par des virgules.

Default (valeur par défaut)

Ce champ est facultatif. Il contient la réponse par défaut pour les modèles chaîne de caractères, sélection et multi-sélection. Pour des questionnaires multi-sélection, il peut contenir une liste de choix séparés par des virgules.

6.5.4 Guide de style spécifique par champ de questionnaires

Champ Type

Aucune indication spécifique excepté : utilisez le type approprié en vous référant à la section précédente.

Champ Description

Voici ci-dessous des instructions spécifiques pour écrire correctement la description (courte et étendue) selon le type de questionnaire.

questionnaire chaîne de caractères/mot de passe
- La description courte est une invite et **non** un titre. Évitez des invites de style question (« IP Address ? ») en faveur d'invites « ouvertes »à (« IP address : »). L'utilisation des deux-points est recommandée.
- La description étendue est un complément à la description courte. Dans la partie étendue, expliquez ce qui est demandé, plutôt que de poser la même question à nouveau en utilisant des mots plus longs. Utilisez des phrases complètes. Un style d'écriture trop concis est fortement découragé.

modèles booléens
- La description courte devrait être rédigée sous la forme d'une question qui devrait être gardée courte et devrait généralement se terminer par un point d'interrogation. Un style d'écriture concis est permis et même encouragé si la question est plutôt longue (rappelez-vous que les traductions sont souvent plus longue que les versions d'origine)
- La description étendue ne devrait **pas** inclure de question.
- À nouveau, veuillez éviter de vous référer à des composants d'interface spécifiques. Une erreur courante pour de telles questionnaires est les constructions du type « if you answer Yes ».

sélection/multi-sélection
- La description courte est une invite et **non** un titre. N'utilisez **pas** des constructions inutiles du type « Please choose… ». Les utilisateurs sont assez intelligents pour réaliser qu'ils doivent choisir quelque chose… :)

– La description étendue devra compléter la description courte. Elle peut se référer aux choix disponibles. Elle peut également mentionner que l'utilisateur peut choisir plus d'un des choix disponibles si le questionnaire est du type sélection multiple (bien que l'interface rende souvent cela clair).

Notes
– La description courte devrait être considéré comme un *titre*.
– La description étendue est ce qui sera affiché comme une description plus détaillée de la note. Faites des phrases, n'utilisez pas un style d'écriture trop concis.
– **N'abusez pas de debconf.** Les notes sont le moyen le plus courant d'abus de debconf. Comme il est écrit dans la page de manuel de debconf-devel : il est préférable de ne les utiliser que pour des avertissements à propos de problèmes très sérieux. Les fichiers NEWS.Debian ou README.Debian sont les endroits appropriés pour un grand nombre de notes. Si, en lisant ceci, vous envisagez de convertir vos modèles de type note en entrées dans NEWS/Debian ou README.Debian, veuillez considérer de conserver les traductions existantes pour une utilisation future.

Champ de choix

S'il est probable que les choix changent souvent, veuillez considérer l'utilisation de l'astuce « __Choices ». Cela séparera chaque choix individuel en une chaîne de caractères seule, ce qui aidera considérablement les traducteurs à faire leur travail.

Champ par défaut

S'il est probable que la valeur par défaut d'un modèle « select » change selon la langue de l'utilisateur (par exemple, s'il s'agit d'un choix de langue), veuillez utiliser l'astuce « _Default-Choice ».

Ce champ spécial permet aux traducteurs de positionner le choix le plus approprié selon leur propre langue. Cela deviendra le choix par défaut quand leur langue sera sélectionnée alors votre choix par défaut sera utilisé pour l'anglais.

Un exemple tiré des modèles du paquet geneweb :

```
Template: geneweb/lang
Type: select
__Choices: Afrikaans (af), Bulgarian (bg), Catalan (ca), Chinese (zh), Czech
# This is the default choice. Translators may put their own language here
# instead of the default.
# WARNING : you MUST use the ENGLISH FORM of your language
# For instance, the french translator will need to put "French (fr)" here.
_DefaultChoice: English (en)[ translators, please see comment in PO files]
_Description: Geneweb default language:
```

Notez l'utilisation de crochets qui permettent des commentaires internes dans les champs debconf. Notez également l'utilisation de commentaires qui apparaîtront dans les fichiers sur lesquels travailleront les traducteurs.

Les commentaires sont nécessaires car l'astuce DefaultChoice est un peu déroutante : les traducteurs peuvent y placer leur propre choix.

Champ par défaut

N'utilisez PAS de champ par défaut vide. Si vous ne voulez pas utiliser de valeurs par défaut, n'utilisez simplement pas du tout Default.

Si vous utilisez po-debconf (et vous **devriez** le faire, voir 2.2), veuillez considérer de rendre ce champ traduisible, si vous pensez qu'il peut être traduit.

Si la valeur par défaut peut varier selon la langue ou le pays (par exemple, la valeur par défaut pour un choix de langue), veuillez considérer l'utilisation du type spécial « _DefaultChoice » documenté dans `po-debconf(7)`).

6.6 Internationalisation

6.6.1 Gestion des traductions de debconf

Comme les porteurs, les traducteurs ont une tâche difficile. Ils travaillent sur un grand nombre de paquets et doivent donc collaborer avec plusieurs responsables différents. De plus, la plupart du temps, l'anglais n'est pas leur langue maternelle, il se peut que vous deviez être particulièrement patient avec eux.

Le but de `debconf` était de faciliter la configuration des paquets aux responsables et aux utilisateurs. À l'origine, les traductions des questionnaires debconf étaient gérées avec `debconf-mergetemplate`. Cependant, cette technique est maintenant déconseillée ; la meilleure façon pour l'internationalisation de `debconf` est d'utiliser le paquet `po-debconf`. Cette méthode est plus facile pour le responsable et pour les traducteurs ; des scripts de transition sont fournis.

Lors de l'utilisation de `po-debconf`, la traduction est stockée dans des fichiers `po` (à l'instar des techniques de traduction de `gettext`). Des fichiers modèles contiennent les messages d'origine et indiquent quels sont les champs traduisibles. Quand vous modifiez l'état d'un champ traduisible en appelant `debconf-updatepo`, la traduction est marquée comme ayant besoin de l'attention des traducteurs. Puis, lors de la construction du paquet, le programme `dh_installdebconf` prendra soin de toute la magie nécessaire à l'ajout du modèle avec les traductions à jour dans les paquets binaires. Veuillez vous reporter à la page de manuel `po-debconf(7)` pour les détails.

6.6.2 Documentation traduite

La traduction de la documentation est cruciale pour les utilisateurs, mais elle représente un important travail. Il n'existe aucun moyen d'éliminer ce travail, mais vous pouvez faciliter les choses pour les traducteurs.

Si vous êtes responsable d'une documentation quelle que soit sa taille, il est plus facile pour les traducteurs d'avoir accès à un système de contrôle de source. Ceci permet aux traducteurs de voir les différences entre deux versions de la documentation, pour, par exemple, qu'ils puissent voir ce qui a besoin d'être retraduit. Il est recommandé que le document traduit inclue une note à propos de la révision de contrôle du source sur laquelle la traduction est basée. Un système intéressant est fourni par doc-check (`http://cvs.debian.org/boot-floppies/documentation/doc-check?rev=HEADcontent-type=text/vnd.viewcvs-markup`) dans le paquet `boot-floppies` qui donne un aperçu de l'état de la traduction pour une langue donnée, en utilisant les commentaires structurés pour une révision donnée du fichier à traduire et, pour un fichier traduit, la révision du fichier d'origine sur laquelle la traduction est basée. Vous pouvez vouloir adapter et fournir ceci dans votre référentiel CVS.

Si vous maintenez de la documentation au format XML ou SGML, nous vous suggérons d'isoler les informations indépendantes de la langue et de les définir comme des entités dans un fichier séparé qui sera inclus par les différentes traductions. Ceci aide, par exemple, à garder à jour les adresses pour plusieurs fichiers.

6.7 Pratiques communes d'empaquetage

6.7.1 Paquets utilisant `autoconf/automake`

Conserver à jour les fichiers d'`autoconf`, `config.sub` et `config.guess`, est critique pour les porteurs, spécialement pour les architectures plus changeantes. De très bonnes pratiques d'empaquetage utilisant `autoconf` et/ou `automake` ont été regroupées dans `/usr/share/doc/autotools-dev/README.Debian.gz` du paquet `autotools-dev`. Vous êtes vivement encouragé à lire ce fichier et à suivre les recommandations indiquées.

6.7.2 Bibliothèques

Pour diverses raisons, il a toujours été difficile d'empaqueter les bibliothèques. La charte impose plusieurs contraintes pour faciliter la maintenance et pour garantir que les mises à jour sont aussi simples que possible quand une nouvelle version amont est disponible. Une erreur dans une bibliothèque peut rendre défectueux une douzaine de paquets qui en dépendent.

Les bonnes pratiques d'empaquetage des bibliothèques ont été regroupées dans le guide d'empaquetage des bibliothèques (`http://www.netfort.gr.jp/~dancer/column/libpkg-guide/`).

6.7.3 Documentation

Assurez-vous de suivre les règles sur la documentation (http://www.debian.org/doc/debian-policy/ch-docs.html).

Si votre paquet contient de la documentation construite à partir de XML ou SGML, nous vous recommandons de ne pas fournir le source XML ou SGML dans le(s) paquet(s) binaire(s). Si les utilisateurs désirent avoir le source de la documentation, ils peuvent récupérer le paquet source.

La Charte spécifie que la documentation doit être fournie au format HTML. Nous vous recommandons également de la fournir au format PDF et texte simple si c'est adapté et qu'une sortie de qualité raisonnable est possible. Cependant, il n'est généralement pas approprié de fournir des versions en texte simple pour la documentation dont le format source est du HTML.

Les principaux manuels fournis devraient être enregistrés par le paquet `doc-base` à l'installation. Reportez-vous à la documentation du paquet `doc-base` pour plus d'information.

6.7.4 Types spécifiques de paquets

Plusieurs types spécifiques de paquets ont des sous-chartes spéciales et des règles et pratiques d'empaquetage correspondantes :

- Les paquets liés à Perl ont leur charte Perl (http://www.debian.org/doc/packaging-manuals/perl-policy/), quelques exemples de paquets qui suivent cette charte sont `libdbd-pg-perl` (module perl binaire) ou `libmldbm-perl` (module perl indépendant de l'architecture).
- Les paquets liés à Python ont leur charte Python ; voir `/usr/share/doc/python/python-policy.txt.gz` dans le paquet `python`.
- Les paquets liés à Emacs ont leur charte Emacs (http://www.debian.org/doc/packaging-manuals/debian-emacs-policy).
- Les paquets liés à Java ont leur charte Java (http://www.debian.org/doc/packaging-manuals/java-policy/).
- Les paquets liés à Ocaml ont leur propre charte que l'on trouve dans `/usr/share/doc/ocaml/ocaml_packaging_policy.gz` du paquet `ocaml`. Un bon exemple est le paquet source `camlzip`.
- Les paquets fournissant des DTD XML ou SGML devraient se conformer aux recommandations que l'on peut trouver dans le paquet `sgml-base-doc`
- Les paquets Lisp devraient s'enregistrer avec le paquet `common-lisp-controller` pour lequel vous pouvez vous reporter au fichier `/usr/share/doc/common-lisp-controller/README.packaging`.

6.7.5 Données indépendantes de l'architecture

Il n'est pas rare d'avoir une grande quantité de données indépendantes de l'architecture fournie avec un programme. Par exemple, des fichiers audio, une collection d'icônes, des motifs de

papiers peints ou autres fichiers graphiques. Si la taille des données est négligeable par rapport à la taille du reste du paquet, il est probablement mieux de tout garder dans le même paquet.

Cependant, si la taille des données est considérable, vous devez envisager de les partager dans un paquet séparé et indépendant de l'architecture (« _all.deb »). Ainsi, en faisant cela, vous évitez une duplication inutile des mêmes données dans onze fichiers .debs pour chaque architecture. Bien que ceci ajoute une surcharge supplémentaire aux fichiers `Packages`, ceci sauve beaucoup d'espace disque sur les miroirs Debian. Séparer les données indépendantes de l'architecture réduit également le temps de traitement de `lintian` ou de `linda` (voir 'Outils du paquet lint' page 108) quand ils sont exécutés sur l'ensemble de l'archive Debian.

6.7.6 Avoir besoin d'une certaine locale pendant la construction

Si vous avez besoin d'une certaine locale pendant la construction, vous pouvez créer un fichier temporaire par cette astuce :

Si vous positionnez LOCPATH à l'équivalent de /usr/lib/locale, et LC_ALL au nom de la locale que vous générez, vous devriez obtenir ce que vous voulez sans être root. Quelque chose comme ceci :

```
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_PAT

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

6.7.7 Rendre les paquets de transition conformes avec deborphan

Deborphan est un programme pour aider les utilisateurs à détecter quels paquets peuvent être enlevés sans problème du système, i.e. ceux dont aucun paquet ne dépend. L'opération par défaut est de chercher seulement parmi les sections libs et oldlibs pour débusquer les bibliothèques inutilisées. Mais si l'on passe le bon paramètre, il tente d'attraper d'autres paquets inutiles.

Par exemple, avec –guess-dummy, deborphan cherche tous les paquets de transition qui étaient nécessaires pour une mise à jour, mais qui peuvent sans problème être supprimés. Pour cela, il recherche la chaîne de caractères « dummy » ou « transitional » dans la description courte.

Ainsi, quand vous créez un tel paquet, assurez-vous d'ajouter ce texte dans la description courte. Si vous cherchez des exemples, exécutez simplement :

```
apt-cache search .|grep dummy
```

or

```
apt-cache search .|grep transitional
```

.

6.7.8 Les meilleures pratiques pour les fichiers `orig.tar.gz`

Il existe deux types d'archives tar (« tarball ») source d'origine : les sources vierges et les sources amont réempaquetées.

Sources vierges

La caractéristique définissant une archive source vierge est que le fichier .orig.tar.gz est identique octet-pour-octet à l'archive tar officielle distribuée par l'auteur amont. [3] Cela rend possible d'utiliser des vérifications de somme pour vérifier facilement que tous les changements entre la version Debian et celle de l'amont sont contenus dans le fichier diff Debian. Également, si le source amont est énorme, les auteurs amont et d'autres qui ont déjà l'archive tar amont peuvent économiser du temps de téléchargement s'ils veulent inspecter votre empaquetage en détail.

Il n'y a pas de règles universellement acceptées suivies par les auteurs amont concernant la structure des répertoires dans leur archive tar, mais `dpkg-source` est cependant capable de traiter la plupart des archives tar comme source vierge. Sa stratégie est équivalente à la suivante :

1 Il décompacte l'archive tar dans un répertoire temporaire vide par

```
zcat chemin/vers/<nom-du-paquet>_<version-amont>.orig.tar.gz | tar xf -
```

2 Si, après cela, le répertoire temporaire ne contient qu'un répertoire et pas d'autres fichiers, `dpkg-source` renomme ce répertoire en `<packagename>-<version-amont>(.orig)`. Le nom du répertoire de premier niveau dans l'archive tar n'a pas d'importance et est oublié.

3 Sinon, l'archive tar a dû être empaqueté sans répertoire de premier niveau commun (honte à l'auteur amont !). Dans ce cas, `dpkg-source` renomme le répertoire temporaire *lui-même* en `<packagename>-<version-amont>(.orig)`.

[3] Nous ne pouvons pas empêcher les auteurs amont de changer l'archive tar qu'ils distribuent sans également incrémenter le numéro de version, il ne peut donc pas y avoir de garantie qu'une archive vierge est identique à ce que l'auteur amont distribue *actuellement* à tout moment. La seule chose à laquelle on peut s'attendre est que c'est identique à quelque chose que l'amont *a* un jour distribuée. Si une différence se produit plus tard (par exemple, si l'amont remarque qu'il n'a pas utilisé la compression maximale pour sa distribution d'origine et qu'il la recompresse), c'est vraiment trop dommage. Comme il n'y a pas de bonne façon d'envoyer un nouveau .orig.tar.gz pour la même version, il n'y a même pas de raison de traiter cette situation comme un bogue.

Réempaquetage des sources amont

Vous **devriez** envoyer des paquets sources avec une archive tar vierge si possible, mais il peut y avoir diverses raisons pour lesquelles cela n'est pas possible. C'est le cas si l'amont ne distribue pas le source comme un tar gzipé du tout ou si l'archive tar amont contient du matériel non libre au sens des DFSG que vous devez supprimer avant l'envoi.

Dans tous ces cas, le développeur doit construire un fichier .orig.tar.gz convenable lui-même. Nous nous référerons à une telle archive tar comme un « source amont réempaqueté ». Notez qu'un « source amont réempaqueté » est différent d'un paquet natif Debian. Un source réempaqueté est toujours fourni avec des changements spécifiques Debian dans un fichier `.diff.gz` séparé et il a toujours un numéro de version composé de `<source-amont>` et de `<debian-revision>`.

Il peut y avoir des cas où il est souhaitable de réempaqueter le source même si l'amont distribue un fichier `.tar.gz` qui pourrait en principe être utilisé dans sa forme vierge. Le plus évident est si des économies d'espaces *significatives* peuvent être réalisées en recompressant l'archive tar ou en supprimant des parties fondamentalement inutiles de l'archive source. Agissez à votre guise à cet endroit, mais soyez prêt à défendre votre décision si vous réempaquetez un source qui aurait pu être vierge.

Un .orig.tar.gz réempaqueté :

1 **doit** contenir des informations détaillées sur la façon dont a été obtenu le source réempaqueté et comment cela peut être reproduit dans le fichier `README.Debian-source` ou un fichier similaire. Ce fichier devrait être dans la partie `diff.gz` du paquet source Debian, habituellement dans le répertoire `debian`, *pas* dans le `orig.tar.gz` réempaqueté. C'est également une bonne idée de fournir une cible `get-orig-source` dans votre fichier `debian/rules` qui répète le processus, comme décrit dans la Charte, debian/rules, le principal script de construction (http://www.debian.org/doc/debian-policy/ch-source.html#s-debianrules).

2 **ne devrait pas** contenir de fichiers qui ne viennent pas de l'auteur amont ou dont vous avez changé le contenu. [4]

3 **devrait**, sauf cas impossible pour des raisons légales, préserver l'infrastructure de construction entière et de portabilité fournie par l'auteur amont. Par exemple, ce n'est pas une raison suffisante pour omettre un fichier qui n'est utilisé que pour la construction sur MS-DOS. De manière similaire, un Makefile fourni par l'amont ne devrait pas être réécrit en exécutant un script configure.

   (*Explication :* il est courant que les utilisateurs Debian qui ont besoin de reconstruire un logiciel pour des plates-formes non-Debian récupèrent le source d'un miroir Debian plutôt que de chercher à localiser un point de distribution amont).

[4] Comme exception spéciale, si l'omission d'un fichier non libre devait entraîner l'échec de la compilation du source sans assistance du diff Debian, il peut être approprié au lieu de cela d'éditer les fichiers, en omettant seulement les parties non libres de ceux-ci et/ou d'expliquer la situation dans un fichier README.Debian-source à la racine de l'arborescence du source. Mais dans ce cas, veuillez également demander instamment à l'auteur amont de faciliter la séparation des composants non libres du reste du source.

4 **devrait** utiliser `<nom-du-paquet>-<version-amont>.orig` comme nom du répertoire de premier niveau dans son archive tar. Cela rend possible la distinction entre des archives tar vierges d'archives réempaquetées.

5 **devrait** être gzipé avec une compression maximale.

La façon canonique de réaliser les deux derniers points est de laisser `dpkg-source -b` construire l'archive tar réempaquetée à partir du répertoire décompacté.

Changer des fichiers binaires dans le `diff.gz`

Il est parfois nécessaire de changer des fichiers binaires contenus dans l'archive tar d'origine ou d'ajouter des fichiers binaires qui ne sont pas dans celle-ci. Si cela est fait en copiant simplement les fichiers dans l'arborescence de l'archive debianisée, `dpkg-source` ne pourra pas gérer cela. D'un autre côté, selon les règles détaillées ci-dessus, vous ne pouvez pas inclure un tel fichier binaire modifié dans un fichier `orig.tar.gz` réempaqueté. Au lieu de cela, incluez le fichier dans le répertoire `debian` dans un format `uuencode` (ou semblable) [5]. Le fichier sera ensuite décodé et copié à son emplacement pendant l'étape de construction. Le changement sera donc visible assez facilement.

Certains paquets utilisent `dbs` pour gérer les correctifs à leur source amont et créent toujours un nouveau fichier `orig.tar.gz` contenant le vrai `orig.tar.gz` dans son répertoire de premier niveau. Cela est discutable concernant la préférence pour un source vierge. D'un autre côté, il est facile de modifier ou d'ajouter des fichiers binaires dans ce cas : placez les simplement dans le fichier `orig.tar.gz` nouvellement créé à côté du vrai et copiez les au bon endroit pendant l'étape de construction.

[5] Le fichier devrait avoir un nom qui indique clairement quel fichier binaire il encode. Habituellement, un postfixe indiquant le codage devrait être ajouté au nom du fichier d'origine. Notez que vous n'avez pas besoin de dépendre de `sharutils` pour avoir le programme `uudecode` si vous utilisez la fonction `pack` de `perl`. Le code pourrait ressembler à ceci :

```
uuencode-file : perl -ne 'print(pack "u", $$_);' $(file) > $(file).uuencoded
uudecode-file : perl -ne 'print(unpack "u", $$_);' $(file).uuencoded > $(file)
```

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 Rapporter des bogues

Nous vous encourageons à remplir des rapports de bogue quand vous trouvez des bogues dans les paquets Debian. En fait, les développeurs Debian sont souvent les testeurs de première ligne. Trouver et rapporter les bogues dans les paquets d'autres développeurs améliore la qualité de Debian.

Lisez les instructions pour créer un rapport de bogue (`http://www.debian.org/Bugs/Reporting`) dans le système de suivi des bogues (`http://www.debian.org/Bugs/`) Debian.

Essayez de soumettre le bogue à partir d'un compte utilisateur normal sur lequel vous pouvez recevoir des courriers, pour que les personnes puissent vous joindre s'ils ont besoin de plus d'informations à propos du bogue. Ne soumettez pas de bogues en tant que root.

Vous pouvez utiliser un outil comme `reportbug(1)` pour soumettre des bogues. Il peut automatiser et dans l'ensemble faciliter le processus.

Assurez-vous que le bogue n'a pas déjà été rapporté. Chaque paquet dispose d'une liste de bogues facilement accessible à `http://bugs.debian.org/`*`nom_paquet`*. Des outils comme `querybts(1)` peuvent également vous fournir ces informations (et `reportbug` invoquera également habituellement `querybts` avant l'envoi).

Essayez d'envoyer vos bogues au bon emplacement. Quand, par exemple, votre bogue concerne un paquet qui réécrit des fichiers d'un autre paquet, vérifiez les listes des bogues pour les *deux* paquets pour 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 que vous n'ayez obtenu la permission du responsable).

De temps en temps, vous pourrez vouloir vérifier ce qui s'est passé à propos des bogues que vous avez rapportés. Saisissez cette occasion pour fermer les bogues que vous ne pouvez plus reproduire. Pour trouver tous les bogues que vous avez rapportés, vous avez simplement besoin d'aller à `http://bugs.debian.org/from:<votre-adresse-de-courrier>`.

7.1.1 Ouvrir un grand nombre de rapports en une seule fois (« mass bug filing »)

Ouvrir un grand nombre de rapports pour le même problème sur un grand nombre de paquets — plus de dix — est une pratique que nous déconseillons. 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 ouvrez 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 mentionner le fait 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 qu'ils ne soient pas redirigés vers les listes de diffusion.

7.2 Effort d'assurance qualité

7.2.1 Travail journalier

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` (reportez-vous à '`lintian`' page 108). Si cela vous paraît impossible, vous devriez alors envisager d'abandonner certains de vos paquets (reportez-vous à 'Abandonner un paquet' page 52). 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 (reportez-vous à 'Maintenance collective' page 64).

7.2.2 Les chasses aux bogues

De temps en temps, le groupe d'assurance qualité organise des chasses aux bogues[1] pour essayer de supprimer le plus de problèmes possible. Elles sont annoncées sur `<debian-devel-announce@lists.debian.org>` et l'annonce explique quel domaine sera visé pendant la chasse : habituellement, il s'agit des bogues empêchant l'intégration du paquet dans la distribution (« Release Critical »), mais il peut arriver qu'ils décident 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 sont différentes au cours de la chasse parce que l'annonce de la chasse est considérée comme une annonce préalable pour les mises à jour indépendantes. Si vous avez des paquets qui peuvent être affectées 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 d'une mise à jour indépendante ou 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 mises à jour indépendantes, ils peuvent en faire une sans avertissement préalable s'ils envoient leur paquet avec un délai d'au moins 3 jours dans DELAYED/3-day. Toutes les autres règles de mise à jour indépendante s'appliquent comme d'habitude ; ils devraient envoyer le correctif de la mise à jour dans le BTS (pour l'un des bogues ouverts corrigé par la mise à jour ou pour un nouveau bogue marqué comme fixé). Ils devraient également respecter tout souhait du responsable s'il en a exprimé.

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

7.3 Contacter d'autres responsables

Pendant vos activités dans Debian, vous aurez à contacter 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 qui sont inscrites à un paquet de source donné par le 'Système de suivi des paquets' page 28. Vous pouvez le faire en utilisant l'adresse `<paquet>@packages.qa.debian.org`.

[1] *Bug Squashing Party*

7.4 Gérer les 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 ne se soit pas désenregistré 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 Absents En Exercice (« Missing In Action) sont enregistrées. Quand un membre du groupe QA contacte un responsable inactif ou trouve plus d'informations sur celui-ci, c'est enregistré dans la base de données MIA. Ce système est disponible dans /org/qa.debian.org/mia sur l'hôte qa.debian.org et peut être interrogé avec un outil de nom `mia-query`. Utilisez

```
mia-query --help
```

pour voir comment interroger la base de données. Si vous déterminez qu'aucune information n'a encore été enregistrée pour un responsable inactif ou si vous voulez ajouter plus d'informations, vous deviez 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 (`https://db.debian.org/`), 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-*-changes). Rappelez-vous également de vérifier si le responsable est indiqué comme en vacances dans la base de données.
- Le nombre de paquets de ce responsable et les conditions de ces paquets. En particulier, reste-t-il des bogues empêchant l'intégration du paquet dans la distribution qui sont ouverts depuis des lustres ? De plus, combien de bogues y a-t-il en général ? Un autre point d'information important est si les paquets ont subi des mises à jour indépendantes 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 est probable qu'il sait 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 est au courant d'information sur le responsable manquant. Veuillez mettre en CC : la personne en question.

Une fois que vous avez réuni 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, ils peuvent abandonner un ou tous les paquets du responsable. Si un paquet a subi une mise à jour indépendante, ils peuvent préférer contacter le responsable ayant fait cette mise à jour — il est peut-être intéressé par le paquet.

Un dernier mot : veuillez vous souvenir d'être poli. Nous sommes tous des volontaires et nous ne pouvons dédier l'intégralité de notre temps à Debian. Vous n'êtes pas non plus au courant des circonstances de la personne impliquée. Elle est peut-être sérieusement malade ou pourrait même nous avoir quitté — vous ne savez pas qui recevra vos courriers. Imaginez comme un proche se sentira s'il lit un courrier pour la personne décédée et trouve un message très impoli, en colère et accusateur !

D'un autre côté, bien que nous soyons tous volontaires, nous avons une responsabilité. 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 documentées et contactez <mia@qa.debian.org>.

7.5 Interagir avec de futurs développeurs Debian

Le succès de Debian dépend de sa capacité à attirer et retenir de nouveaux et talentueux volontaires. Si vous êtes un développeur expérimenté, nous vous recommandons de vous impliquer dans le processus d'apport des nouveaux responsables. Cette section décrit comment aider les futurs développeurs.

7.5.1 Parrainer des paquets

Parrainer un paquet veut dire envoyer un paquet pour un responsable qui n'est pas encore autorisé à le faire lui-même, un candidat nouveau responsable. Parrainer un paquet veut aussi dire que vous en acceptez la responsabilité.

Les nouveaux responsables ont habituellement certaines difficultés à créer des paquets Debian — ceci est bien compréhensible. C'est pourquoi le parrain est là pour vérifier que le paquet parrainé est assez bon pour être inclus dans Debian. (Notez que si le paquet parrainé est nouveau, les ftpmasters devront également l'inspecter avant de l'intégrer)

Effectuer un parrainage en signant simplement l'envoi ou en recompilant le paquet **n'est définitivement pas recommandé**. Vous devez construire le paquet source comme si vous vouliez construire l'un de vos paquets. Rappelez-vous que cela ne change rien si vous avez laissé le

nom du candidat développeur dans le changelog et dans le fichier de contrôle, il est toujours possible de savoir que c'est vous qui avez fait l'envoi.

Si vous êtes un gestionnaire de candidature pour un futur développeur, vous pouvez également être son parrain. Ainsi, vous pourrez vérifier comment le candidat gère la partie « Tâches et Capacités »[2] de sa candidature.

7.5.2 Gérer les paquets parrainés

En envoyant un paquet sponsorisé vers Debian, vous certifiez que le paquet atteint les standards minimum de Debian. Ceci implique que vous devez construire et tester le paquet sur votre propre système avant l'envoi.

Vous ne pouvez pas simplement envoyer un fichier `.deb` binaire d'un filleul. En théorie, vous devriez seulement demander le fichier diff et l'emplacement de l'archive source d'origine et ensuite vous devriez récupérer le source et appliquer le diff vous-même. En pratique, vous pouvez vouloir utiliser le paquet source construit par votre filleul. Dans ce cas, vous devez vérifier qu'il n'a pas modifié les fichiers sources du fichier `.orig.tar.gz` qu'il fournit.

N'ayez pas peur de répondre au filleul et de lui indiquer les changements qu'il doit faire. Cela prend souvent plusieurs échanges de courriers avant que le paquet ne soit dans un état acceptable. Être un parrain veut dire être un mentor.

Une fois que le paquet a atteint les standards Debian, construisez et signez le paquet avec

```
dpkg-buildpackage -kKEY-ID
```

avant de l'envoyer dans le répertoire incoming. Bien sûr, vous pouvez également utiliser toute partie de votre *KEY-ID*, tant qu'elle est unique dans votre porte-clés secret.

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. Celui-ci recevra donc tous les courriers du système de suivi des bogues (BTS) à propos de son paquet.

Si vous préférez laisser une trace plus visible de votre travail de parrainage, vous pouvez ajouter une ligne l'indiquant dans la plus récente entrée du changelog.

Vous êtes encouragé à garder un œil sur le suivi des paquets que vous parrainez en utilisant le 'Système de suivi des paquets' page 28.

7.5.3 Recommander un nouveau développeur

Reportez-vous à la page sur les recommandations pour un développeur prospectif (`http://www.debian.org/devel/join/nm-advocate`) sur le site web Debian.

[2] Tasks and Skills

7.5.4 Gérer les candidatures des nouveaux candidats

Veuillez vous reporter à la liste à vérifier pour les responsables de candidatures (http://www.debian.org/devel/join/nm-amchecklist) sur le site web Debian.

Chapitre 8

Internationalisation, traduction, être internationalisé et être traduit

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 dans 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 (http://www.debian.org/doc/manuals/intro-i18n/) de Tomohiro KUBOTA, « I18N (internationalisation) veut dire la modification d'un logiciel ou de 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) veut dire 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 dbuild pour le portage. Le plus gros du travail doit donc être réalisé manuellement.

8.1 Comment sont gérées les 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 (http://www.iro.umontreal.ca/contrib/po/HTML/), le projet de traduction de Gnome (http://developer.gnome.org/projects/gtp/) ou celui de KDE (http://i18n.kde.org/). La seule ressource centralisée dans Debian est les statistiques de traduction Debian centralisées (http://www.debian.org/intl/l10n/) 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 (i.e., seul APT peut les utiliser quand il est configuré convenablement). Les responsables n'ont rien à faire de particulier pour gérer les traductions des descriptions de paquets ; les traducteurs devraient utiliser le DDTP (http://ddtp.debian.org/).

Pour les questionnaires debconf, les responsable 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). On peut trouver certaines statistiques à la fois sur le site du DDTP (à propos de ce qui est vraiment traduit) et sur le site des statistiques de traduction Debian centralisées (http://www.debian.org/intl/l10n/) (à propos de ce qui est intégré dans les paquets).

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

Pour la documentation générale à propos de Debian, le processus est plus ou moins le même que pour les pages web (les traducteurs ont accès au CVS), 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.

Le plus notablement, 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 Debian au sein d'un dépôt CVS spécifique (http://cvs.debian.org/manpages/?cvsroot=debian-doc).

8.2 FAQ I18N & 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. Une fois qu'ils ont fini, vous recevrez de leur part votre document traduit dans votre boîte aux lettres.

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 devriez avoir plus confiance dans 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 votre 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 « suivi au développeur amont » (*forwarded to upstream*) 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 & 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 gagnera 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 le codage 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 rendent pas compte des erreurs qu'ils trouvent, 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 :)
- Dans tous les 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 manière 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 Debian. 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_paquet>`.

A.1 Outils de base

Les outils suivants sont pratiquement nécessaires à tout responsable.

A.1.1 `dpkg-dev`

Le paquet `dpkg-dev` contient les outils (y compris `dpkg-source`) nécessaires pour déballer, fabriquer et livrer des paquets Debian source. 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`

Le paquet `debconf` fournit une interface unifiée pour configurer les paquets interactivement. 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 trouverez la documentation de ce paquet dans le paquet `debconf-doc`.

Beaucoup pensent que ce système devrait être utilisé pour tout paquet nécessitant une configuration interactive ; reportez-vous à la 'Gestion de la configuration avec `debconf`' page 81. `debconf` n'est pas requis par la *charte Debian* pour le moment ; cependant, cela pourra changer.

A.1.3 `fakeroot`

`fakeroot` simule les privilèges *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 avec : `dpkg-buildpackage -rfakeroot`.

A.2 Outils du paquet 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 du paquet lint aide les responsables de paquet à trouver automatiquement des problèmes habituels et des 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.

Veuillez vous reporter à 'Tester le paquet' page 37 pour plus d'informations sur comment et quand utiliser Lintian.

Vous pouvez aussi obtenir un résumé de tous les problèmes reportés par Lintian sur vos paquets à `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 `linda`

`linda` est un autre vérificateur de paquet. Il est semblable à `lintian`, mais il a un ensemble de tests différents. Il est écrit en Python plutôt qu'en Perl.

A.2.3 `debdiff`

`debdiff` (du paquet `devscripts`, 'devscripts' page 112) compare des listes de fichiers et des fichiers de contrôle de deux paquets. C'est un simple test de régression qui peut vous 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 qu'il indique sont normaux, mais il peut aider à empêcher différents accidents.

Vous pouvez l'exécuter sur un couple de paquets binaires :

```
debdiff package_1-1_arch.deb package_2-1_arch.deb
```

Ou même sur un couple de fichiers de changements :

```
debdiff package_1-1_arch.changes package_2-1_arch.changes
```

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

A.3 Aides pour le fichier `debian/rules`

Des outils de construction de paquets rendent le processus d'écriture du fichier `debian/rules` plus facile. Veuillez voir les 'Scripts d'aide' page 71 pour plus d'informations sur les raisons pour lesquelles ils peuvent être désirables ou non.

A.3.1 `debhelper`

Le paquet `debhelper` regroupe un ensemble de programmes qui peuvent être utilisés dans `debian/rules` pour automatiser les tâches courantes relatives à la fabrication des 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 debian/rules* ».

Il existe aussi un certain nombre de petites extensions `debhelper` trop éphémères pour être documentées. Vous en listerez la plupart en faisant `apt-cache search ^dh-`.

A.3.2 debmake

debmake — un précurseur de debhelper — est un assistant moins modulaire pour manipuler le fichier debian/rules. Il inclut deux programmes principaux : deb-make, utile au développeur Debian pour convertir un paquet source normal (non-Debian) en paquet source Debian, et debstd qui regroupe le type de fonction que l'on trouve dans debhelper.

Le consensus actuel est que l'utilisation de debmake est déconseillée au profit de debhelper. C'est un bogue d'utiliser debmake pour les nouveaux paquets. Les nouveaux paquets utilisant debmake seront rejetés de l'archive.

A.3.3 dh-make

Le paquet 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.3.4 yada

Le paquet yada est un autre assistant pour la création de paquets. Il utilise un fichier debian/packages pour générer debian/rules et d'autres fichiers nécessaires dans le sous-répertoire debian/. Le fichier debian/packages contient des instructions pour construire les paquets et il n'y a pas besoin de créer de fichiers Makefile. Il existe la possibilité d'utiliser un moteur de macros semblable à celui utilisé dans les fichiers SPECS des paquets source RPM.

Pour plus d'informations, voir le site de YADA (http://yada.alioth.debian.org/).

A.3.5 equivs

equivs est un autre paquet pour faire des paquets. Il est souvent conseillé pour un usage local, si vous avez besoin de faire un paquet pour satisfaire des dépendances. Il est aussi parfois utilisé pour faire des « méta-paquets » qui sont des paquets dont l'unique objet est de dépendre d'autres paquets.

A.4 Constructeurs de paquets

Les paquets suivants aident le processus de construction des paquets, en général en contrôlant dpkg-buildpackage ainsi que la gestion du support de tâches.

A.4.1 `cvs-buildpackage`

Le paquet `cvs-buildpackage` permet de mettre à jour ou de récupérer des paquets sources dans un référentiel CVS, il permet de fabriquer un paquet Debian depuis le référentiel CVS et il assiste le développeur 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 probablement *experimental* et de bénéficier des avantages d'un système de contrôle de version.

A.4.2 `debootstrap`

Le paquet `debootstrap` vous permet d'amorcer un système Debian de base à n'importe quel endroit dans votre système de fichiers. Par « système de base », nous entendons le strict minimum nécessaire pour fonctionner et installer le reste du système.

Avoir un système comme celui-ci peut vous servir de différentes manières. Vous pouvez, par exemple, déplacer votre racine dans ce système avec `chroot` pour tester vos dépendances de construction. Vous pouvez aussi l'utiliser pour voir comment se comporte votre paquet quand il est installé dans un environnement minimum.

A.4.3 `pbuilder`

`pbuilder` construit un système « chrooté » et compile des paquets dans ce système. Ceci est très utile pour vérifier que les dépendances de compilation de votre paquet sont correctes et pour vous assurer qu'aucune dépendance de construction inutile ou incorrecte n'existe dans le paquet résultant.

Un paquet lié est `pbuilder-uml`, 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 aussi 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, c'est une partie du système utilisé par les porteurs pour construire des paquets binaires pour toutes les architectures disponibles. Veuillez vous reporter à '`buildd`' page 57 pour plus d'informations et à `http://buildd.debian.org/` pour voir le système en fonctionnement.

A.5 Téléchargeurs de paquets

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

A.5.1 dupload

Le paquet dupload contient un script du même nom pour mettre à jour des paquets dans l'archive Debian, tracer les mises à jour et les annoncer par courrier électronique automatiquement. Vous pouvez le configurer pour faire des mises à jour à d'autres endroits et avec d'autres méthodes.

A.5.2 dput

Le script dput fait à peu près la même chose que dupload, mais il le fait différemment. Il a aussi quelques fonctions supplémentaires telles que la possibilité de vérifier la signature GnuPG et les sommes de contrôles avant le téléchargement et la possibilité de démarrer dinstall en mode simulation (*dry-run*) après le téléchargement.

A.5.3 dcut

Le script dcut (faisant partie du paquet 'dput' de la présente page) aide à la suppression de fichiers du répertoire d'envoi ftp.

A.6 Automatisation de la maintenance

Les outils suivants aident à automatiser les différentes tâches de maintenance par l'ajout des entrées de changelog ou de lignes de signatures, par la recherche de bogues à partir d'Emacs et par l'utilisation du fichier officiel config.sub le plus récent.

A.6.1 devscripts

Le paquet devscripts contient des scripts et outils qui sont très utiles pour maintenir vos paquets Debian. Parmi ces scripts, vous trouverez debchange et dch qui manipulent votre fichier debian/changelog depuis la ligne de commande et debuild qui est construit au-dessus de dpkg-buildpackage. L'outil bts est également très utile pour mettre à jour l'état des rapports de bogue depuis la ligne de commande. Le programme uscan peut être utilisé pour suivre les nouvelles versions amont de vos paquets. Le programme debrsign peut être utilisé pour signer à distance un paquet avant de l'envoyer, ce qui est agréable quand la machine sur laquelle vous construisez le paquet est différente de celle où résident vos clés GPG.

Vérifiez 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 des personnes assurant la maintenance de paquets qui utilisent autoconf et/ou automake. Contient également des fichiers cano-

niques `config.sub` et `config.guess` qui sont connus pour fonctionner avec tous les portages Debian.

A.6.3 dpkg-repack

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

Cet utilitaire peut faciliter la copie de paquet d'un ordinateur vers un autre ou la re-création de paquets installés sur un système, mais 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 leur somme de hachage MD5. Veuillez noter que tous les paquets n'ont pas de sommes MD5 car elles ne sont pas requises par la charte.

A.6.6 dpkg-dev-el

`dpkg-dev-el` fournit des macros Emacs Lisp qui apportent une aide lors de l'édition des fichiers du répertoire `debian` de votre paquet. Par exemple, il y a des fonctions pratiques pour lister les bogues actuels d'un paquet et pour finaliser la dernière entrée d'un fichier `debian/changelog` file.

A.6.7 dpkg-depcheck

`dpkg-depcheck` (du paquet `devscripts`, 'devscripts' page précédente) exécute une commande sous `strace` pour déterminer tous les paquets utilisés par la commande.

Pour les paquets Debian, c'est utile quand vous devez créer une ligne `Build-Depends` pour votre nouveau paquet : exécuter le processus de compilation avec `dpkg-depcheck` vous 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, particulièrement si votre 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 facilitent le travail des porteurs et la compilation croisée.

A.7.1 `quinn-diff`

`quinn-diff` est utilisé pour localiser les différences d'une architecture à l'autre. Il pourrait vous dire, 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[1] d'une manière similaire à `dpkg`. De plus, les paquets `dpkg-buildpackage` et `dpkg-shlibdeps` ont été améliorés 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 `debiandoc-sgml`

`debiandoc-sgml` fournit la DTD DebianDoc SGML qui est habituellement utilisée pour la documentation Debian. Ce manuel, par exemple, est écrit en DebianDoc. Il fournit également des scripts pour construire et décliner le source en de multiples formats de sortie.

La documentation sur la DTD peut être trouvée dans le paquet `debiandoc-sgml-doc`.

A.8.2 `debian-keyring`

Contient les clés publiques GPG et PGP des développeurs Debian. Voir 'Gérer votre clé publique' page 9 et la documentation du paquet pour plus d'informations.

[1] *cross-compilation*

A.8.3 `debview`

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