Full Circle

LE MAGAZINE INDÉPENDANT DE LA COMMUNAUTÉ UBUNTU LINUX

NUMÉRO SPÉCIAL - SÉRIE DÉVELOPPEMENT UBUNTU

ÉDITION SPÉCIALE DÉVELOPPEMENT UBUNTU

photo: FreeFotouk (Flickr.com)

ARTICLES EXTRAITS DES NUMÉROS 49 À 52 DU MAGAZINE

full circle magazine n'est affilié en aucune façon à Canonical Ltd.

Au sujet du Full Circle

Le Full Circle est un magazine gratuit, libre et indépendant, consacré à toutes les versions d'Ubuntu, qui fait partie des systèmes d'exploitation Linux. Chaque mois, nous publions des tutoriels, que nous espérons utiles, et des articles proposés par des lecteurs. Le Podcast, un complément du Full Circle, parle du magazine même, mais aussi de tout ce qui peut vous intéresser dans ce domaine.

Clause de non-responsabilité :

Cette édition spéciale vous est fournie sans aucune garantie ; les auteurs et le magazine Full Circle déclinent toute responsabilité pour des pertes ou dommages éventuels si des lecteurs choisissent d'en appliquer le contenu à leurs ordinateur et matériel ou à ceux des autres.

Full Circle Magazine spécial

Full Circle

LE MAGAZINE INDÉPENDANT DE LA COMMUNAUTÉ UBUNTU LINUX

Bienvenue dans une nouvelle édition spéciale consacrée à un sujet unique !"

En réponse à des demandes de nos lecteurs, nous rassemblons le contenu de certains de nos articles publiés sur plusieurs mois pour en faire des Numéros spéciaux.

Cette fois-ci, il ne s'agit de rien d'autre que la réimpression de la série « Développement Ubuntu », parties 1 à 4, écrite par Daniel Holbach et tirée des numéros 49 à 52 du FCM. Pas de chichi, juste les faits.

Veuillez garder à l'esprit la date de la première publication ; les versions actuelles du matériel et des logiciels peuvent différer de ceux que nous présentons, ainsi vérifiez bien votre matériel et la version de vos logiciels avant d'émuler les tutoriels de cette édition spéciale. Il se peut qu'une version plus récente de ces logiciels soit déjà installée chez vous ou disponible dans les dépôts de votre distribution.

Amusez-vous bien !

Nos coordonnées :

Site web :
http://www.fullcirclemagazine.org/

Forums :
http://ubuntuforums.org/forumdisplay.php?f=270

IRC : #fullcirclemagazine onchat.freenode.net

Équipe éditoriale :

Rédacteur en chef : Ronnie Tucker (pseudo : RonnieTucker)
ronnie@fullcirclemagazine.org
Webmaster : Rob Kerfia (pseudo : admin / linuxgeekery)
admin@fullcirclemagazine.org
Podcaster : Robin Catling (pseudo : RobinCatling)
podcast@fullcirclemagazine.org
Dir. Comm. : Robert Clipsham (pseudo : mrmonday)
mrmonday@fullcirclemagazine.org

SOME RIGHTS RESERVED

Les articles contenus dans ce magazine sont publiés sous la licence Creative Commons Attribution-Share Alike 3.0 Unported license. Cela signifie que vous pouvez adapter, copier, distribuer et transmettre les articles mais uniquement sous les conditions suivantes : vous devez citer le nom de l'auteur d'une certaine manière (au moins un nom, une adresse e-mail ou une URL) et le nom du magazine (« Full Circle Magazine ») ainsi que l'URL www.fullcirclemagazine.org (sans pour autant suggérer qu'ils approuvent votre utilisation de l'œuvre). Si vous modifiez, transformez ou adaptez cette création, vous devez distribuer la création qui en résulte sous la même licence ou une similaire.

Full Circle Magazine est entièrement indépendant de Canonical, le sponsor des projets Ubuntu. Vous ne devez en aucun cas présumer que les avis et les opinions exprimés ici aient reçus l'approbation de Canonical.

TUTORIEL

Introduction au développement d'Ubuntu

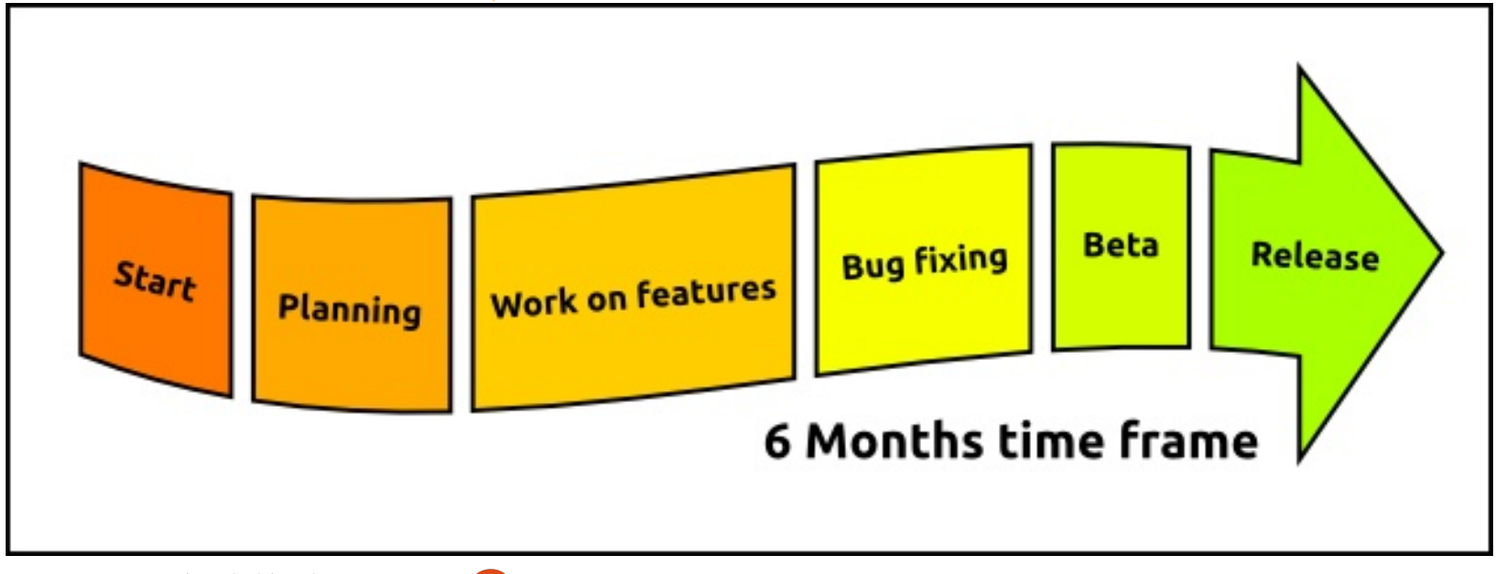
Écrit par Daniel Holbach

Ubuntu est fait de milliers de composants différents, écrits dans de nombreux langages de programmation différents. Chaque composant, que ce soit une bibliothèque logicielle, un outil ou une application graphique, est disponible en tant que paquet source. Les paquets sources, pour la plupart, sont constitués de deux parties : le code source en lui-même et des méta-données. Ces méta-données incluent les dépendances du paquet, des informations de copyright et de licence, et des instructions pour construire (build) le paquet. Une fois que ce paquet source est compilé, le processus de construction produit des paquets binaires, qui sont les fichiers .deb que les utilisateurs peuvent installer.

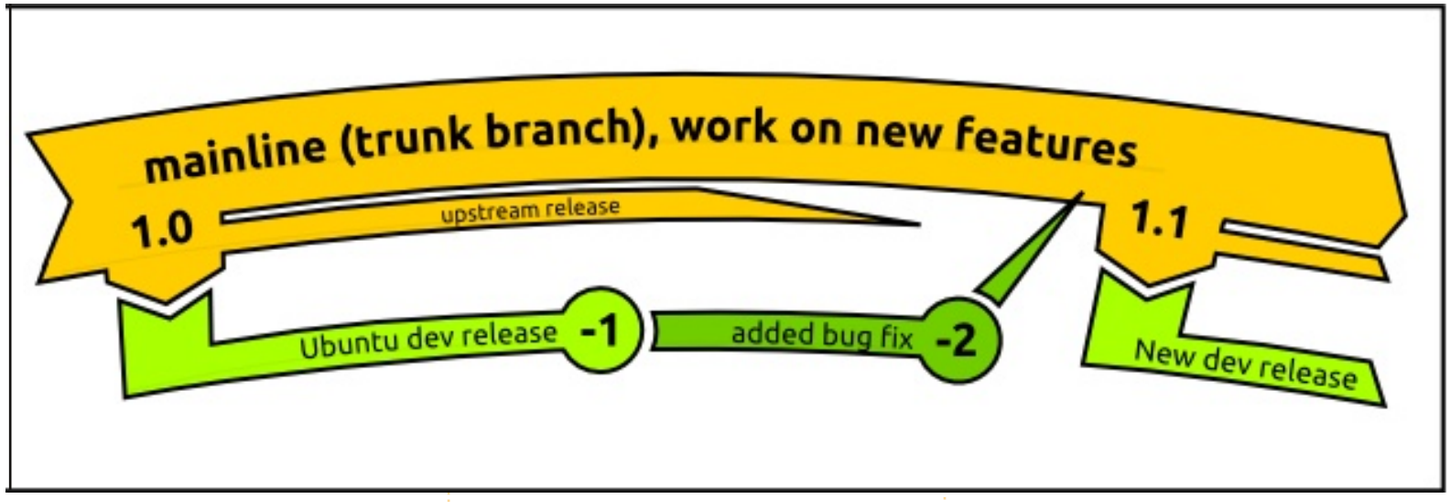
Chaque fois qu'une nouvelle version d'une application sort ou que quelqu'un apporte un changement au code source qui entre dans Ubuntu, le paquet source doit être téléchargé vers les machines de « build » pour être compilé à nouveau. Les paquets binaires qui en résultent sont alors distribués vers l'archive et tous ses miroirs dans différents pays. Les URL dans /etc/apt/source.list pointent vers une archive ou un miroir. Chaque jour sont construites des images CD pour une sélection de différentes versions d'Ubuntu. Ubuntu Desktop, Ubuntu Server, Kubuntu et d'autres spécifient une liste des paquets requis sur chaque CD. Ces images CD sont ensuite utilisées pour des tests d'installation et fournissent un retour pour la prévision des futures sorties.

Le développement d'Ubuntu est très dépendant de l'étape en cours dans le cycle de sortie. Une nouvelle version d'Ubuntu sort tous les six mois, ce qui n'est possible que parce qu'on a fixé des dates de gel très strictes. À chaque fois qu'une date de gel arrive, les développeurs doivent faire moins de modifications qui sont moins intrusives. Le gel des fonctionnalités est la première date de gel importante après que la moitié du cycle est passée. À cette étape, les fonctionnalités doivent être largement implémentées. Le reste du cycle est censé être centré sur la correction de bogues. Après cela, l'interface utilisateur, puis la documentation, le noyau, etc., sont gelés, puis une version bêta est livrée pour être soumise à de nombreux tests. À partir de cette version bêta, seul les bogues critiques sont réparés et une version candidate à la sortie est fabriquée ; si elle ne contient pas de problème sérieux, ce sera elle la version finale.

Des milliers de paquets sources, des milliards de lignes de code et des centaines de contributeurs, tout cela demande beaucoup de communication et de planification afin de maintenir un standard élevé de qualité. Au début de chaque cycle de sortie, se tient le Sommet des Développeurs Ubuntu où les développeurs et les contributeurs se réunissent pour planifier les fonctionnalités des prochaines sorties. Chaque fonctionnalité



est discutée par ses parties prenantes et on rédige une spécification contenant des informations détaillées sur ses hypothèses, son implémentation, les changements nécessaires à d'autres endroits, comment la tester et ainsi de suite. Tout cela est fait d'une manière transparente et ouverte, de façon à ce que ceux qui ne peuvent pas assister à l'événement en personne puissent participer à distance et écouter un flux audio, discuter avec les participants et souscrire aux changements des spécifications, pour être toujours à jour.



Cependant on ne peut pas discuter de chaque changement durant une réunion, en particulier parce qu'Ubuntu repose sur des changements qui seront apportés dans d'autres projets. C'est pourquoi ceux qui contribuent à Ubuntu restent constamment en contact. La plupart des équipes et des projets utilisent des listes de diffusion dédiées pour éviter trop de bruit de fond. Pour une coordination immédiate, les développeurs et les contributeurs utilisent des canaux IRC. Toutes les discussions sont ouvertes et publiques.

Les rapports de bogues sont un autre outil important concernant la communication. Lorsqu'un défaut est découvert dans un paquet ou un morceau de l'infrastructure, un rapport de bogue est rempli sur Launchpad. Toutes les informations sont collectées dans ce rapport et on met à jour si nécessaire son importance, son statut et la personne à qui elle est attribuée. Cela rend cet outil très efficace pour maîtriser les bogues dans un paquet ou un projet et organiser la charge de travail.

La plupart des logiciels disponibles dans Ubuntu ne sont pas écrits par les développeurs Ubuntu eux-mêmes. La plupart sont écrits par des développeurs d'autres projets Open Source, puis sont intégrés dans Ubuntu. Ces projets sont appelés des « Upstream » (flux du dessus), car leur code source coule dans Ubuntu où on se contente de l'intégrer. La relation avec les « Upstream » est d'une importance critique dans Ubuntu. Il ne s'agit pas simplement de code qu'Ubuntu récupère depuis les « Upstream », mais c'est aussi les « Upstream » qui récupèrent des utilisateurs, des rapports de bogues et des correctifs depuis Ubuntu (et d'autres distributions).

Le plus important des « Upstream » pour Ubuntu est Debian. Debian est la distribution sur laquelle est basée Ubuntu et de nombreuses décisions de conception concernant l'infrastructure des paquets en proviennent. Traditionnellement, Debian a toujours eu des personnes dédiées pour la maintenance de chacun de ses paquets ou des équipes de maintenance dédiées. Dans Ubuntu, il y a aussi des équipes qui ont un intérêt pour un sous-ensemble de paquets et, bien entendu, chaque développeur a un domaine particulier d'expertise, mais la participation (et les droits de téléchargement montant) est en général ouverte à tous ceux qui montrent de l'habileté et de la volonté.

Apporter un changement à Ubuntu en tant que nouveau contributeur n'est pas si compliqué qu'il y paraît et peut être une expérience très gratifiante. Il ne s'agit pas seulement d'apprendre quelque chose de nouveau et d'excitant, mais aussi de partager une solution et de résoudre un problème pour des millions d'utilisateurs de par le monde.

Le Développement Open Source se passe dans un monde distribué qui a différents buts et différentes zones d'intérêt. Par exemple, on peut imaginer le cas d'un « Upstream » particulier qui pourrait vouloir travailler

sur une nouvelle fonctionnalité importante, tandis qu'Ubuntu, à cause de son calendrier de sortie serré, pourrait être intéressé par une version solide avec seulement une correction de bogue supplémentaire. C'est pourquoi nous faisons usage du « Développement distribué », dans lequel le code est retravaillé dans plusieurs branches qui sont ensuite fusionnées les unes avec les autres après les critiques et les discussions nécessaires.

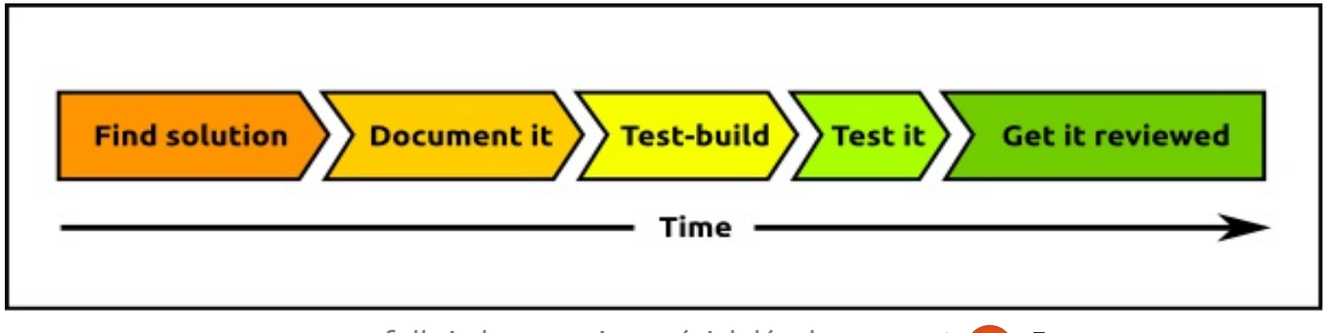
Dans l'exemple précédent, cela aurait un sens de sortir Ubuntu avec la version actuelle du projet, d'ajouter la correction de bogue, la placer parmi les « Upstream » pour la version suivante et la fournir (si elle est prête) dans la version suivante d'Ubuntu. Ce serait le meilleur compromis possible et une situation dans laquelle tout le monde gagne.

Pour corriger un bogue dans Ubuntu, vous devez d'abord récupérer le code source du paquet, puis travailler pour le réparer, écrire une documentation pour que les autres développeurs et utilisateurs comprennent facilement ce que vous avez fait, puis construire un paquet pour le tester. Après l'avoir testé, vous pouvez facilement proposer que les changements soient intégrés dans la version d'Ubuntu en cours de développement. Un développeur qui a les droits de téléchargement montant vous renverra une critique puis l'intégrera dans Ubuntu.

Lorsqu'on cherche une solution, c'est souvent une bonne idée de vérifier avec l'« Upstream » pour voir si le problème (ou une solution possible) est déjà connu et, si ce n'est pas le cas, de faire de votre mieux pour que la solution soit un effort concerté. Des étapes supplémentaires peuvent intégrer le fait de faire le changement sur une ancienne version d'Ubuntu encore supportée, puis de le faire suivre à l'« Upstream ».

Les prérequis les plus importants pour réussir dans le développement d'Ubuntu sont d'avoir le don de « faire fonctionner les choses à nouveau », de ne pas craindre la lecture des documentations et de poser des questions, d'avoir un bon esprit d'équipe et d'aimer le travail de détective.

Voici quelques bons endroits pour poser vos questions : ubuntu-motu-mentors@lists.ubuntu.com et **#ubuntu-motu** sur irc.freenode.net. Vous trouverez facilement plein de nouveaux amis et des gens qui ont la même passion que vous : rendre le monde meilleur en fabriquant de meilleurs logiciels Open Source.



UN APPEL EN FAVEUR DU PARTI PODCAST

Comme vous l'aurez entendu dans l'épisode n° 15 du podcast, nous lançons un appel à propos des sujets d'opinion pour la partie de l'émission du même nom.

Au lieu de vous attendre à ce que nous déclamions nos opinions sur tout ce qui nous passe par la tête, vous pourriez peut-être nous souffler un sujet et, ensuite, guetter l'apparition des champignons atomiques à l'horizon ! Il est fort probable que nous ne serons pas tous les trois du même avis. Ou une idée encore plus radicale, envoyez-nous une opinion de façon contributive.

Vous pouvez poster des commentaires et des avis sur la page du podcast sur fullcirclemagazine.org, dans notre section des Forums Ubuntu, ou nous écrire à podcast@fullcirclemagazine.org.

Vous pouvez aussi faire un commentaire audio d'une durée de moins de 30 secondes et nous l'envoyer à la même adresse. Les commentaires et l'audio peuvent être modifiés pour une question de longueur. Veuillez vous rappeler qu'il s'agit d'une émission tout public.

Ce serait super d'avoir des contributeurs qui viendraient dans l'émission et exprimeraient leur avis en personne.

Robin

TUTORIEL Développement Ubuntu Partie 2 - Préparatifs

Écrit par Daniel Holbach

Il y a pas mal de choses qu'il faut faire pour pouvoir commencer à développer pour Ubuntu. Cet article est conçu pour que vous puissiez paramétrer votre ordinateur afin de travailler avec des paquets, puis télécharger vos paquets vers Launchpad. Voici ce que nous allons traiter :

• L'installation de logiciels en rapport avec la création de paquets, y compris :
• des utilitaires de création de paquets spécifiques à Ubuntu ;
• un logiciel de cryptage qui permettra de vérifier que c'est vous qui avez fait le travail ;
• un logiciel de cryptage supplémentaire qui vous permettra de transférer des fichiers en toute sécurité.
• La création et la configuration de votre compte sur Launchpad.
• Le paramétrage de votre environnement de développement pour vous aider à créer des paquets localement, à interagir avec d'autres développeurs et à proposer vos modifications sur Launchpad.

N.B. Il est recommandé de travailler sur les paquets en utilisant la version de développement d'Ubuntu actuelle. Cela vous permettra de tester les modifications dans le même environnement où ces modifications seront appliquées et utilisées.

Ne vous inquiétez pas, cependant, car, sur la page wiki des versions développement d'Ubuntu (https://wiki.ubuntu.com/UsingDevelopmentReleases), vous verrez plein de différentes façons de vous servir de la version développement en toute sécurité.

Installer les logiciels de base pour créer des paquets

Il existe de nombreux outils qui rendront votre vie de développeur d'Ubuntu beaucoup plus facile. Vous allez rencontrer ces outils plus tard dans ce guide. Pour installer la plupart des outils requis, exécutez cette commande :

```
sudo apt-get install gnupg
pbuilder ubuntu-dev-tools
bzr-builddeb apt-file
```

Cette commande installera les logiciels suivants :

gnupg - GNU Privacy Guard contient les outils dont vous aurez besoin pour créer une clé cryptographique avec laquelle vous signerez les fichiers que vous voudrez télécharger vers Launchpad.
pbuilder - un outil qui permet la création de paquets reproductibles dans un environnement propre et isolé.
ubuntu-dev-tools (et devscripts, une dépendance directe) - un ensemble d'outils qui rendent les tâches de création de paquets plus faciles.
bzr-builddeb (et bzr, une dépendance) - des outils distribués de contrôle de version qui font que beaucoup de développeurs peuvent collaborer et travailler sur le même code facilement, tout en rendant la fusion du travail de chacun extrêmement aisée.
apt-file vous donne la possibilité de trouver facilement le paquet binaire qui contient un fichier spécifique.
apt-cache (qui fait partie du paquet apt) vous donne encore plus d'informations sur les paquets sur Ubuntu.

Créer votre clé GPG

GPG signifie GNU Privacy Guard et il exécute l'Open PGP standard qui vous permet de signer et de crypter des messages et des fichiers. Ceci est utile pour de nombreuses raisons. Dans ce cas-ci, il est important de pouvoir signer des fichiers au moyen de votre clé, afin qu'ils puissent être identifiés comme quelque chose sur lequel vous avez travaillé. Si vous téléchargez un paquet source vers Launchpad, il l'acceptera uniquement s'il peut déterminer de façon absolue qui a téléchargé le paquet.

Pour générer une nouvelle clé GPG, exécutez :

```
gpg --gen-key
```

D'abord, GPG vous demandera quel type de clé vous voulez générer. La sélection du type par défaut (RSA et DSA) est très bien. Puis, il vous demandera la taille voulue. Celle par défaut (actuellement 2048) est bien, mais 4096 est plus sûre. Ensuite, il vous demandera si la clé doit expirer à un moment quelconque. Vous pouvez répondre « 0 » en toute sécurité, ce qui veut dire que la clé n'expirera jamais. Les dernières questions concerneront vos nom et adresse mail. Il

suffit de choisir ceux que vous utiliserez pour le développement Ubuntu ; vous pourrez rajouter d'autres adresses mail plus tard. Il n'est pas nécessaire de rajouter un commentaire. Enfin, il vous demandera de choisir une phrase de passe. Choisissez une phrase sûre.

Maintenant, GPG créera une clé pour vous, ce qui peut prendre un peu de temps ; il lui faut des octets au hasard donc si vous donnez du travail à faire au système ça lui conviendra très bien. Bougez le curseur un peu partout !

Une fois la clé faite, vous aurez un message qui ressemble à celui-ci :

```
pub   4096R/43CDE61D 2010-12-06
Key fingerprint = 5C28 0144 FB08 91C0 2CF3  37AC 6F0B F90F 43CD E61D
uid                  Daniel Holbach <dh@mailempfang.de>
sub   4096R/51FBE68C 2010-12-06
```

Ici, l'ID de la clé est 43CDE61D.

Vous devez ensuite télécharger la partie publique de votre clé vers un serveur de clés afin que tout le monde puisse identifier des messages et des fichiers comme étant les vôtres. Pour ce faire, saisissez :

```
gpg --send-keys <KEY ID>
```

Votre clé sera envoyée à un serveur de clés, mais un réseau de serveurs de clé synchronisera automatiquement la clé entre eux ; une fois cette synchronisation terminée, votre clé publique signée sera prête pour vérifier vos contributions partout dans le monde.

Créer votre clé SSH

SSH est l'abréviation de « Secure Shell » ; c'est un protocole qui vous permet d'échanger des données en sécurité sur un réseau. Habituellement, on utilise SSH pour accéder à un autre ordinateur, ouvrir un « shell » (ou interpréteur de commandes) dessus et l'utiliser pour transférer des fichiers en toute sécurité. Pour ce qui nous concerne, nous allons nous servir de SSH principalement pour sécuriser nos communications avec Launchpad.

Pour générer une clé SSH, saisissez :

```
ssh-keygen -t rsa
```

Le nom de fichier par défaut est, la plupart du temps, assez compréhensible et vous pouvez donc le laisser tel quel. Pour les besoins de la sécurité, il est vivement recommandé d'utiliser une phrase de passe.

Installer et configurer pbuilder

Pbuilder vous donne la possibilité de compiler des paquets localement, sur votre machine. Il a deux ou trois raisons d'être principales :

• La création se fera dans un environnement propre et minimal. Cela vous aide à vous assurer que les paquets créés par vos soins réussissent de façon reproductible, mais sans modifier votre système.

• Vous n'avez pas besoin d'installer localement toutes les dépendances nécessaires.

• Vous pouvez configurer des instances multiples pour diverses versions d'Ubuntu et de Debian.

Configurer pbuilder est très facile. Éditez ~/.pbuilderrc et ajoutez-y la ligne suivante :

```
COMPONENTS="main universe multiverse restricted"
```

Ainsi toutes les dépendances seront satisfaites au moyen de toutes les composantes. Exécutez ensuite :

```
pbuilder-dist <release> create
```

où <release> est, par exemple, natty, maverick, lucid ou, dans le cas de Debian, peut-être sid. Cela va prendre pas mal de temps, car il téléchargera tous les paquets nécessaires pour une « installation minimale ». Toutefois, ceux-ci seront mis en cache.

Configurer votre machine pour travailler avec Launchpad

Le paramétrage local de base accompli, la prochaine étape sera de configurer votre système pour que vous puissiez travailler avec Launchpad. Cette partie se concentrera sur les sujets suivants :

• Ce qu'est Launchpad et la création d'un compte Launchpad ;

• Le téléchargement de vos clés GPG et SSH vers Launchpad ;

• Comment configurer Bazaar pour travailler avec Launchpad ;

• Comment configurer Bash pour travailler avec Bazaar.

À propos de Launchpad

Launchpad est l'infrastructure centrale dont nous nous servons avec Ubuntu. Il garde précieusement non seulement nos paquets et codes, mais aussi des choses telles que des traductions, des rapports de bogue et des renseignements sur les gens qui travaillent sur Ubuntu et sur les équipes dont ils sont membres. Vous utiliserez également Launchpad pour publier les réparations que vous proposez et pour demander à d'autres développeurs Ubuntu de les examiner et les parrainer.

Vous aurez besoin de vous inscrire sur Launchpad et fournir quelques renseignements. Ainsi, vous pourrez télécharger du code de et vers Launchpad, soumettre des rapports de bogues et plus encore.

Obtenir votre compte Launchpad

Si vous n'avez pas encore de compte Launchpad, vous pouvez en créer un facilement à : https://launchpad.net/+login. Si vous avez un compte Launchpad, mais n'arrivez pas à vous souvenir de votre nom d'utilisateur, vous pouvez le retrouver en allant à https://launchpad.net/people/+me et en cherchant la partie qui vient après le ~ dans l'URL.

Le procédé d'inscription sur Launchpad vous demandera de choisir un nom d'affichage. L'utilisation de votre vrai nom ici est souhaitable afin que vos collègues développeurs Ubuntu puissent apprendre à mieux vous connaître.

Lorsque vous créez un nouveau compte, Launchpad vous enverra un mail contenant un lien qu'il faudra ouvrir dans votre navigateur pour vérifier votre adresse mail. Si vous ne le recevez pas, vérifiez dans votre dossier de pourriels.

La page aide pour les nouveaux comptes sur Launchpad (https://help.launchpad.net/YourAccount/NewAccount) contient d'autres informations au sujet du procédé et d'autres paramètres que vous pourrez changer.

Télécharger votre clé GPG vers Launchpad

Pour découvrir votre « fingerprint » (empreinte digitale) GPG, exécutez :

```
gpg --fingerprint
<email@address.com>
```

et vous verrez quelque chose comme :

```
pub   4096R/43CDE61D 2010-12-
06
Key fingerprint = 5C28 0144
FB08 91C0 2CF3  37AC 6F0B
F90F 43CD E61D
uid                  Daniel
Holbach <dh@mailempfang.de>
sub   4096R/51FBE68C 2010-12-
06
```

Allez alors à https://launchpad.net/people/+me/+editpgpkeys et copiez la partie concernant votre « Key fingerprint » dans la zone de texte. Dans le cas ci-dessus, ce serait 5C28 0144 FB08 91C0 2CF3 37AC 6F0B F90F 43CD E61D. Cliquez maintenant sur « Import Key ».

Launchpad utilisera la « fingerprint » pour voir si votre clé se trouve sur le serveur de clés Ubuntu et, si c'est le cas, vous enverra un mail crypté vous demandant de confirmer l'importation de votre clé. Vérifiez votre compte mail et lisez le mail envoyé par Launchpad. Si votre client mail prend le cryptage OpenPGP en charge, il vous demandera le mot de passe [Ndt : la phrase de passe] que vous avez choisi pour la clé lorsqu'elle fut générée. Saisissez le mot de passe [Ndt : la phrase de passe], puis cliquez sur le lien pour confirmer que la clé est à vous.

Launchpad crypte le mail en utilisant votre clé publique, afin de s'assurer que la clé est la vôtre. Si votre logiciel mail ne prend pas le cryptage OpenPGP en charge, copiez le contenu du mail crypté, saisissez gpg dans un terminal, puis collez le contenu du mail dans la fenêtre du terminal.

De retour sur le site Web de Launchpad, utilisez le bouton « Confirm » et Launchpad terminera l'importation de votre clé OpenPGP.

Vous trouverez plus de renseignements à https://help.launchpad.net/YourAccount/ImportingYourPGPKey

Télécharger votre clé SSH vers Launchpad

Ouvrez https://launchpad.net/people/+me/+editsshkeys dans un navigateur web et ouvrez également ~/.ssh/id_rsa.pub dans un éditeur de texte. Il s'agit de la partie publique de votre clé SSH ; vous pouvez donc la partager avec Launchpad en toute sécurité. Copiez le contenu du fichier et collez-le dans la zone de texte « Add an SSH key ». Maintenant cliquez sur « Import Public Key ».

Pour plus de renseignements sur ce processus, visitez la page « Crea-

ting an SSH KeyPair » sur Launchpad (https://help.launchpad.net/YourAccount/CreatingAnSSHKeyPair).

Configurer Bazaar

Bazaar est l'outil utilisé pour enregistrer des modifications de code de façon logique, pour pouvoir échanger des modifications proposées et les fusionner, même si le développement se fait simultanément. Pour vous présenter à Bazaar, il suffit d'exécuter :

```
bzr whoami "Bob Dobbs <subgenius@example.com>"
```

```
bzr launchpad-login subgenius
```

whoami dira à Bazaar à quels nom et adresse mail il devra envoyer vos messages « commit ». Avec launchpad-login, vous lui signifiez votre nom d'utilisateur Launchpad. De cette manière, le code que vous publiez sur Launchpad vous sera attribué.

N.B. Si vous n'arrivez pas à vous rappeler votre nom d'utilisateur, allez à https://launchpad.net/people/+me et notez où il vous renvoie. La partie après le ~ dans l'URL est votre nom d'utilisateur Launchpad.

Configurer votre « shell » (interpréteur de commandes)

Tout comme Bazaar, les outils de création de paquets Debian/Ubuntu ont besoin d'apprendre des choses sur vous aussi. Dans un éditeur, ouvrez votre ~/.bashrc et rajoutez quelque chose comme ceci à la fin :

```
export DEBFULLNAME="Bob Dobbs"

export DEBEMAIL="subgenius@example.com"
```

Maintenant sauvegardez le fichier et, soit redémarrez votre terminal, soit exécutez :

```
source ~/.bashrc
```

(Si votre interpréteur de commandes est différent de celui par défaut (autrement dit, bash) veuillez éditer le fichier de configuration pour le vôtre en conséquence.)

LE MOIS PROCHAIN :
La réparation de bogues.

UN APPEL EN FAVEUR DU PARTI PODCAST

Comme vous l'aurez entendu dans l'épisode n° 15 du podcast, nous lançons un appel à propos des sujets d'opinion pour la partie de l'émission du même nom.
Au lieu de vous attendre à ce que nous déclamions nos opinions sur tout ce qui nous passe par la tête, vous pourriez peut-être nous souffler un sujet et, ensuite, guetter l'apparition des champignons atomiques à l'horizon ! Il est fort probable que nous ne serons pas tous les trois du même avis. Ou une idée encore plus radicale, envoyez-nous une opinion de façon contributive.
Vous pouvez poster des commentaires et des avis sur la page du podcast sur fullcirclemagazine.org, dans notre section des Forums Ubuntu, ou nous écrire à podcast@fullcirclemagazine.org.
Vous pouvez aussi faire un commentaire audio d'une durée de moins de 30 secondes et nous l'envoyer à la même adresse. Les commentaires et l'audio peuvent être modifiés pour une question de longueur. Veuillez vous rappeler qu'il s'agit d'une émission tout public.
Ce serait super d'avoir des contributeurs qui viendraient dans l'émission et exprimeraient leur avis en personne.

Robin

TUTORIEL Développement Ubuntu p. 3 - Corriger un bug

Écrit par Daniel Holbach

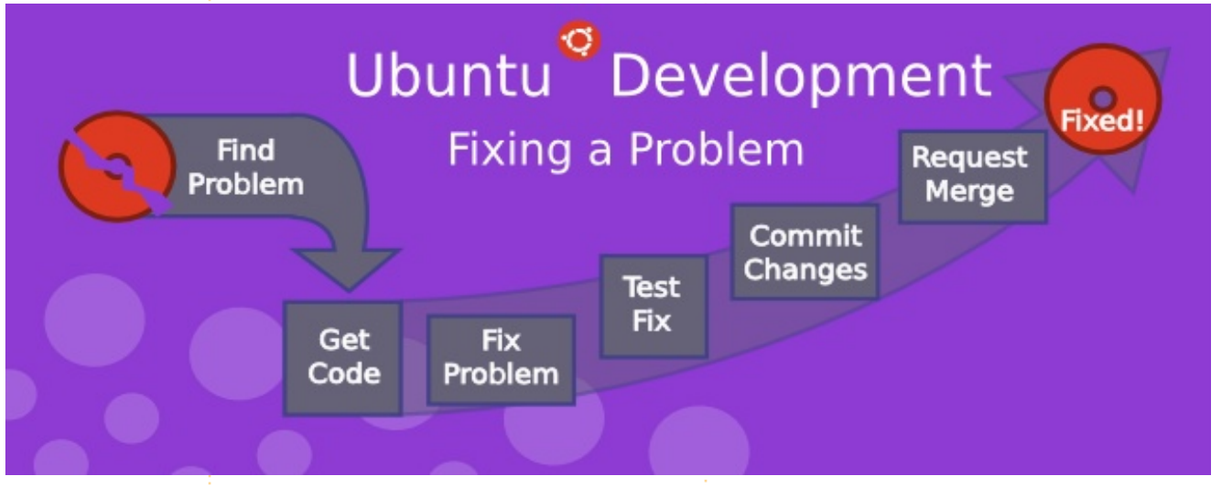
Si vous avez suivi les instructions pour vous préparer au développement d'Ubuntu, vous devriez être prêt à démarrer.

Comme vous pouvez le voir dans l'image de droite, il n'y a pas de surprises dans le processus de correction des bugs dans Ubuntu : vous avez trouvé un problème, vous récupérez le code, travaillez sur le correctif, le testez, téléchargez vos modifications sur Launchpad et demandez que votre travail soit examiné et fusionné. Dans ce guide, nous allons passer par toutes les étapes nécessaires une par une.

Trouver le problème

Il y a beaucoup de façons différentes de trouver des choses sur lesquelles travailler. Ça peut être un rapport de bogue que vous déposez vous-même (ce qui vous donne une bonne occasion de tester le correctif) ou un problème que vous avez trouvé ailleurs, peut-être dans un rapport de bogue.

Harvest [Ndt : la récolte] est l'endroit où nous gardons la trace de différentes listes de choses à faire concernant le développement d'Ubuntu. Cela répertorie les bogues déjà corrigés en amont ou dans Debian, cela liste de petits bogues (nous les appelons « Bite-size » [Ndt : « de la taille d'un octet »]), et ainsi de suite. Jetez-y un coup d'œil et trouvez votre premier bogue sur lequel travailler.



Comprendre ce qu'il y a à réparer

Si vous ne connaissez pas le paquet source contenant le code où se situe le problème, mais que vous connaissez le chemin du programme affecté sur votre système, vous pouvez découvrir le paquet source sur lequel vous aurez besoin de travailler.

Disons que vous avez trouvé un bug dans Tomboy, une application de bureau de prise de notes. L'application Tomboy peut être démarrée en exécutant /usr/bin/tomboy sur la ligne de commande. Pour trouver le paquet binaire contenant cette application, utilisez cette commande :

```
apt-file find /usr/bin/tomboy
```

Cela devrait afficher :

```
tomboy: /usr/bin/tomboy
```

Remarquez que la partie précédant les deux-points est le nom du paquet binaire. Il arrive souvent que le paquet source et le paquet binaire aient des noms différents. C'est notamment le cas quand un paquet source unique est utilisé pour construire plusieurs paquets binaires différents. Pour trouver le paquet source correspondant à un paquet binaire spécifique, saisissez :

```
apt-cache show tomboy | grep ^Source:
```

Dans ce cas, rien n'est affiché, ce qui signifie que Tomboy est aussi le nom du paquet binaire. Un exemple où les noms des paquets source et binaire sont différents est python-vigra. Ceci est le nom du paquet binaire, mais le paquet source s'appelle réellement libvigraimpex et peut être trouvé avec cette commande (et sa sortie) :

```
apt-cache show python-vigra | grep ^Source:
```

```
Source: libvigraimpex
```

Récupérer le code

Une fois que vous connaîtrez le paquet source sur lequel travailler, vous souhaiterez avoir une copie du code sur votre système, de sorte que vous puissiez le déboguer. Cela se fait en « connectant » la branche des paquets sources correspondant au paquet source qui vous intéresse. Launchpad maintient des branches de paquets sources pour tous les paquets dans Ubuntu. Une fois que vous avez une branche locale du paquet source, vous pouvez étudier le bogue, créer un correctif et télécharger le correctif proposé sur Launchpad, sous la forme d'une branche Bazaar. Lorsque vous êtes satisfait de votre correction, vous pouvez soumettre une « proposition de fusion », qui demande à d'autres développeurs Ubuntu d'examiner et d'approuver votre changement. S'ils sont d'accord avec vos modifications, un développeur Ubuntu va télécharger la nouvelle version du paquet vers Ubuntu afin que chacun obtienne le bénéfice de votre excellente correction - et que vous obteniez un peu de remerciements. Vous êtes maintenant sur la bonne voie pour devenir un développeur Ubuntu ! Dans les sections suivantes, nous allons décrire les détails sur la manière de « brancher » du code, de pousser [Ndt : ou télécharger] votre correction et de demander une révision.

Travailler sur un correctif

Il y a des livres entiers concernant la façon de trouver des bogues, les corriger, les tester, etc. Si vous êtes complètement nouveau en programmation, essayer d'abord de corriger des bogues faciles tels que les fautes de frappe évidentes. Essayez de faire des changements aussi minimes que possible et de documenter vos modifications et vos hypothèses clairement.

Avant de travailler sur un correctif vous-même, assurez-vous d'enquêter pour savoir si quelqu'un d'autre l'a déjà corrigé ou travaille actuellement sur un correctif. De bonnes sources à vérifier sont :

- les listes de bogues (ouverts et fermés) en amont (et sur Debian),

- l'historique de révisions en amont (ou sur une version nouvelle) pourrait avoir résolu le problème,

- des ajouts de paquets ou des bogues de Debian ou d'autres distributions.

Si vous trouvez un patch pour corriger le problème, disons attaché à un rapport de bogue, exécutez cette commande dans le répertoire source pour appliquer le patch :

```
patch -p1 < ../bugfix.patch
```

Reportez-vous à la page 1 du manuel de patch pour les options et les arguments tels que –dry-run, -p<num>, etc.

Tester la correction

Pour construire un paquet de test avec vos modifications, exécutez ces commandes :

```
bzr bd -- -S -us -uc
```

```
pbuilder-dist <release> build
../<paquet>_<version>.dsc
```

Cela va créer un paquet source à partir du contenu de la branche (-us-uc va simplement supprimer l'étape qui signe le paquet source) et pbuilder-dist va construire le paquet depuis la source pour n'importe quelle version (« release ») que vous choisissez.

Une fois la génération réussie, installez le paquet depuis ~/pbuilder/<release>_result/ (en utilisant sudo dpkg -i <paquet>_<version>.deb). Puis testez pour voir si le bogue est corrigé.

Documenter le correctif

Il est très important de documenter suffisamment vos modifications pour que les développeurs qui regarderont le code à l'avenir n'aient pas à deviner quel avait été votre raisonnement et vos hypothèses. Chaque paquet source Debian et Ubuntu inclut un fichier debian/changelog, où les changements de chaque paquet téléchargé sont suivis.

La meilleure façon de mettre cela à jour est d'exécuter :

```
dch -i
```

Cela va ajouter une entrée passe-partout au fichier changelog pour vous et va lancer un éditeur où vous pourrez remplir les blancs. Un exemple de cela pourrait être :

```
paquetspecial (1.2-3ubuntu4)
natty; urgence=low
  * debian/control:
description actualisée pour
inclure frobnicator (LP:
#123456)
 -- Emma Adams
<emma.adams@isp.com> Sat, 17
Jul 2010 02:53:39 +0200
```

dch devrait déjà avoir rempli la première et la dernière ligne d'une telle entrée du changelog pour vous. La ligne 1 se compose du nom du paquet source, du numéro de version, la version d'Ubuntu dans laquelle il sera téléchargé, l'urgence (qui est presque toujours « low » [Ndt : « faible »].

La dernière ligne contient toujours le nom, l'adresse électronique et l'horodatage (au format RFC 5322) de la modification.

Cela étant dit, concentrons-nous sur l'entrée effective du changelog elle-même : il est très important de documenter :

- où le changement a été fait ;
- ce qui a changé ;
- où la discussion sur le changement a eu lieu.

Dans notre exemple (très simple), le dernier point est couvert par (LP: #123456) qui se réfère à Launchpad, bogue n° 123456. Les rapports de bogues, les fils de discussion ou les spécifications, sont généralement de bonnes informations à fournir comme justification pour un changement. En prime, si vous utilisez la notation LP: #<nombre> pour les bogues de Launchpad, le bogue sera automatiquement fermé quand le paquet sera téléchargé vers Ubuntu.

Pousser le correctif

Une fois l'entrée du changelog écrite et enregistrée, il vous suffit d'utiliser :

```
debcommit
```

et le changement sera « commité » [Ndt : enregistré] localement avec votre entrée changelog utilisée comme message de « commit ».

Pour le pousser sur Launchpad, en tant que nom de la branche à distance, vous devez vous conformer à la nomenclature suivante :

```
lp:~<votreIDLaunchpad>/ubuntu/<release>/<package>/<branchname>
```

Cela pourrait, par exemple, être :

```
lp:~emmaadams/ubuntu/natty/specialpackage/fix-for-123456
```

Donc, si vous exécutez simplement :

```
bzr push lp:~emmaadams/ubuntu/natty/specialpackage/fix-for-123456
bzr lp-open
```

vous devriez en avoir terminé. La commande push devrait le pousser sur Launchpad et la seconde commande ouvrira la page Launchpad de la branche à distance dans votre navigateur. Sur cette page, cherchez le lien « (+) Proposer pour la fusion », puis cliquez dessus pour que votre changement soit vérifié par quelqu'un et inclus dans Ubuntu.

Le mois prochain : un aperçu de la structure de répertoires Debian.

Zéro temps d'arrêt

Below Zero est un spécialiste d'hébergeur de serveurs serveurs en implantation de proximité au Royaume-Uni.

Contrairement à d'autres, nous ne fournissons que l'espace rack et la bande passante. Cela rend notre service plus fiable, plus flexible, plus concentré et plus compétitif quant au prix.

Nous nous spécialisons uniquement dans l'hébergement de serveurs près de chez nous et leurs systèmes, au sein des Centres de données écossais.

Au cœur de l'infrastructure de nos réseaux est le routage BGP4, à la pointe de la technologie, qui fournit une livraison optimale des données et aussi un procédé automatique en cas de panne faisant appel à nos multiples pourvoyeurs remarquables. Les clients peuvent être certains que la bande passante proposée est de qualité maximale ; notre politique est de payer plus pour les meilleurs pourvoyeurs et, parce que nous achetons en gros, nos prix extrêmement compétitifs ne sont pas impactés.

Chez **Below Zero,** nous vous aidons à atteindre Zéro temps d'arrêt.

www.zerodowntime.co.uk

TUTORIEL Développement Ubuntu p. 4 - Debian/

Écrit par Daniel Holbach

Cet article va vous expliquer brièvement les différents fichiers nécessaires à la fabrication des paquets Ubuntu et contenus dans le répertoire debian/. Les plus importants en sont changelog, control, copyright et rules. Ils sont obligatoires pour tous les paquets. Certains fichiers supplémentaires dans debian/ peuvent être utilisés afin de personnaliser et de configurer le comportement du paquet. Nous verrons certains de ces fichiers dans cet article, mais ma liste n'est pas censée être exhaustive.

Le fichier changelog

Ce fichier contient, comme son nom l'indique, une liste des modifications apportées à chaque version. Il a un format spécifique qui donne le nom du paquet, sa version, la distribution, les changements, et la personne qui a fait les changements à un moment donné. Si vous avez une clé GPG (voir : Mise en place), veillez à utiliser le même nom et la même adresse de courriel dans changelog que dans votre clé. Voici un exemple de fichier changelog :

```
paquet (version)
distribution;
urgency=urgence
  * détails des
changements
  - encore plus de détails
des changements
  * et encore d'autres
détails des changements

— nom du responsable
<courriel>[deux espaces]
date
```

Le format (surtout celui de la date) est important. La date doit être dans le format RFC 5322, que l'on peut obtenir en utilisant la commande date -R. Pour plus de commodité, la commande dch peut être utilisée pour modifier le fichier changelog ; elle mettra à jour la date automatiquement. Les changements mineurs sont indiqués par un tiret « - » et les changements majeurs par une astérisque « * ». Si vous fabriquez un paquet à partir de zéro, dch –create (dch est dans le paquet devscripts) va créer pour vous un fichier debian/changelog standard.

Voici un exemple de fichier changelog pour le paquet « hello » :

```
hello (2.6-0ubuntu1) natty;
urgency=low

  * Nouvelle version amont,
avec beaucoup de corrections
de bugs et améliorations de
fonctionnalités.
-- Jane Doe
<packager@example.com>  Thu,
21 Apr 2011 11:12:00 -0400
```

Remarquez que le numéro de version se termine par -0ubuntu1, c'est la révision de la distrib., utilisée pour que le paquet puisse être mis à jour (pour corriger les bogues par exemple) lors des nouveaux ajouts au sein des sources de la même version.

Ubuntu et Debian ont des schémas de versions de paquets légèrement différents pour éviter les conflits de paquets au sein des sources d'une même version. Si un paquet Debian a été modifié dans Ubuntu, il a un ubuntuX (où X est le numéro de révision Ubuntu) ajouté à la fin de la version Debian. Ainsi, si le paquet Debian « hello 2.6-1 » a été modifié par Ubuntu, le numéro de version serait 2.6-1ubuntu1. Si un paquet pour une application n'existe pas dans Debian, alors la révision Debian est 0 (par exemple 2.6-0ubuntu1).

Pour plus d'informations, consultez la section changelog (Section 4.4) de la Charte Debian.

Le fichier control

Le fichier control contient les informations que le gestionnaire de paquets (comme apt-get, synaptic, ou adept) utilise, les dépendances de dates de build, des informations du mainteneur et beaucoup plus.

Pour le paquet « hello » d'Ubuntu, le fichier control ressemble à ceci :

```
Source: hello
Section: devel
Priority: optional
Maintainer: Ubuntu Developers
<ubuntu-devel-
discuss@lists.ubuntu.com>
XSBC-Original-Maintainer:
Jane Doe
<packager@example.com>
Standards-Version: 3.9.1
Build-Depends: debhelper (>= 7)
Bzr-Vcs: lp:ubuntu/hello
Homepage:
http://www.gnu.org/software/hello/

Package: hello
Architecture: any
Depends: ${shlibs:Depends}
Description: Le message de
bienvenue classique, ainsi
```

```
qu'un bon exemple
Le programme GNU hello produit
une message de bienvenue
habituel et chaleureux. Ceci
permet aux personnes non
programmeurs d'utiliser un outil
informatique classique dont ils
ne disposeraient pas autrement.
Sérieusement, cette fois : ceci
est un exemple montrant comment
créer un paquet Debian. C'est la
version Debian du programme «
hello world » du projet GNU (qui
sert, lui-même d'exemple pour le
projet GNU).
```

Le premier paragraphe décrit le paquet source - y compris la liste des paquets nécessaires pour construire le paquet depuis les sources dans le champ Build-Depends. Il contient également certaines méta-informations comme le nom du mainteneur, la version de Charte Debian à laquelle se conforme le paquet, l'emplacement du dépôt de contrôle des versions de paquets et la page d'accueil en amont.

Notez que, dans Ubuntu, nous avons indiqué une adresse générique dans le champ Maintainer parce que n'importe qui peut changer n'importe quel paquet (ceci diffère de Debian, où la modification des paquets est généralement limitée à une personne ou une équipe). En général, les paquets dans Ubuntu devraient avoir le champ Maintainer réglé à Ubuntu Developers ubuntu-develdiscuss@lists.ubuntu.com.

Si le champ Maintainer est modifié, l'ancienne valeur doit être enregistrée dans le champ XSBC-Original-Maintainer. Cela peut être fait automatiquement avec le script update-maintainer disponible dans le paquet ubuntu-dev-tools. Pour plus d'informations, voir la spécification Debian du champ Maintainer sur le wiki d'Ubuntu.

Chaque paragraphe supplémentaire décrit un paquet binaire à construire.

Pour plus d'informations, consultez la section fichier de contrôle (chapitre 5) de la Charte Debian.

Le fichier copyright

Ce fichier donne les informations de copyright à la fois pour la source en amont et pour le paquet. Les chartes Ubuntu et Debian (Section 12.5) exigent que chaque paquet installe une copie intégrale des informations de licence et de copyright dans /usr/share/doc/$(nom_du_paquet)/copyright.

En règle générale, les informations de copyright se trouvent dans le fichier COPYING dans le répertoire source du programme. Ce fichier doit contenir des informations telles que les noms de l'auteur et de la personne qui a fait le paquet, l'URL d'où provient la source, une ligne de copyright avec l'année et le détenteur du copyright et, enfin, le texte du copyright. Voici un exemple de modèle :

```
Format:
http://svn.debian.org/wsvn/dep/
web/deps/dep5.mdwn?op=file&rev=
166
Upstream-Name: Hello
Source:
ftp://ftp.example.com/pub/games

Files: *
Copyright: Copyright 1998 John
Doe <jdoe@example.com>
License: GPL-2+

Ce programme est un logiciel
libre ; vous pouvez le
redistribuer et/ou le modifier
conformément aux dispositions de
la Licence Publique Générale GNU,
telle que publiée par la Free
Software Foundation ; version 2
de la licence, ou encore (à votre
choix) toute version ultérieure.
.
Ce programme est distribué dans
l'espoir qu'il sera utile, mais
SANS AUCUNE GARANTIE ; sans même
la garantie implicite de
COMMERCIALISATION ou D'ADAPTATION
A UN USAGE PARTICULIER. Pour plus
de détails, voir la Licence
Publique Générale GNU.
.
Un exemplaire de la Licence
Publique Générale GNU doit être
fourni avec ce programme ; si ce
n'est pas le cas, écrivez à la
Free Software Foundation, Inc.,
51 Franklin St, Fifth Floor,
Boston, MA 02110-1301 USA .
.
Sur les systèmes Debian, le texte
complet de la Licence Générale
Publique GNU version 2 peut être
trouvé dans le fichier
/usr/share/common-licenses/GPL-2.

Files: debian/*
Copyright: Copyright 1998 Jane
Doe <packager@example.com>
License: GPL-2+
```

Cet exemple suit la proposition Debian DEP-5 : debian/copyright analysable par la machine. Vous êtes encouragés à utiliser ce format aussi.

Le fichier rules

Le dernier fichier que nous devons examiner est rules [Ndt : les règles]. Celui-ci fait tout le travail pour la création de notre paquet. Il s'agit d'un Makefile avec des cibles pour compiler et installer l'application, puis créer le fichier .deb à partir des fichiers installés. Il a également une cible pour nettoyer tous les fichiers de construction de sorte que vous vous retrouviez seulement avec un paquet source à nouveau.

Voici une version simplifiée du fichier rules créé par dh_make (que vous trouverez dans le paquet dh-make) :

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

```
# décommentez la ligne
suivante pour passe en mode
verbeux
#export DH_VERBOSE=1

%:
	dh  $@
```

Regardons ce fichier en détail. Il va passer chaque cible de construction correspondant à l'un des arguments passés lors de l'appel à debian/rules à /usr/bin/dh, qui à son tour appellera toutes les commandes dh_* nécessaires.

dh exécute une séquence de commandes debhelper. Les séquences supportées correspondent aux cibles d'un fichier debian/rules : « build » (construire), « clean » (nettoyer), « install » (installer), « binary-arch », « binary-indep », et « binary ». Pour voir quelles commandes sont exécutées pour chaque cible, lancez :

```
dh binary-arch --no-act
```

Les commandes dans la séquence binary-indep sont appelées avec l'option « -i » pour s'assurer qu'elles ne fonctionnent que sur des paquets binaires indépendants, et les commandes dans les séquences binary-arch sont appelées avec l'option « -a » pour s'assurer qu'elles fonctionnent uniquement sur les paquets dépendant de l'architecture.

Chaque commande debhelper va l'indiquer dans debian/package.debhelper.log lorsqu'elle s'exécute avec succès (ce fichier est effacé par dh_clean). Ainsi dh sait quelles commandes ont déjà été exécutées, pour quels paquets, et ne relance pas ces commandes. À chaque fois que dh est exécuté, le journal est examiné pour trouver la dernière commande qui a été exécutée parmi celles de la séquence spécifiée. dh continue ensuite avec la commande suivante dans la séquence. Les options –until (jusqu'à), –before (avant), –after (après) et –remaining (restantes) peuvent modifier ce comportement.

Si debian/rules contient une cible avec un nom comme override_dh_command, quand il arrive à cette commande dans la séquence, dh exécutera cette cible à partir du fichier rules plutôt que d'exécuter la commande effective. La cible contournée peut alors exécuter la commande avec des options supplémentaires ou exécuter des commandes entièrement différentes à la place. (Notez que, pour utiliser cette fonctionnalité, vous devrez indiquer un Build-depend [Ndt: une dépendance] sur debhelper 7.0.50 ou supérieur.

Jetez un oeil à /usr/share/doc/debhelper/examples/ et « man dh » pour plus d'exemples. Voyez aussi la section rules (Section 4.9) de la Charte Debian.

Fichiers supplémentaires

Le fichier d'installation

Le fichier d'installation est utilisé par dh_install pour installer les fichiers dans le paquet binaire. Il y a deux cas d'utilisation standard :

• pour installer les fichiers dans votre paquet qui ne sont pas traités par le système de build en amont ;
• pour fractionner un seul paquet source de grande taille en plusieurs paquets binaires.

Dans le premier cas, le fichier d'installation devra contenir une ligne par fichier installé, spécifiant le fichier et le répertoire d'installation. Par exemple, le fichier d'installation suivant installerait le script foo du répertoire racine du paquet source dans usr/bin et un fichier desktop [Ndt: icône sur le bureau] du répertoire debian dans usr/share/applications :

```
foo usr/bin
debian/bar.desktop
usr/share/applications
```

Quand un paquet source produit plusieurs paquets binaires, dh installera les fichiers dans debian/tmp plutôt que directement dans debian/<paquet>. Les fichiers installés dans debian/tmp peuvent alors être déplacés dans des paquets binaires différents en utilisant plusieurs fichiers $nom_du_paquet.ins-tall. On s'en sert souvent pour séparer de grandes quantités de données indépendantes de l'architec-

ture de paquets dépendants de l'architecture et les placer dans Architecture: tous les paquets. Dans ce cas, seul le nom des fichiers (ou répertoires) à installer sont nécessaires, sans le répertoire d'installation. Par exemple, foo.install contenant uniquement les fichiers dépendants de l'architecture pourrait ressembler à :

```
usr/bin/
usr/lib/foo/*.so
```

Alors que foo-common.install ne contenant que les fichiers indépendants de l'architecture pourrait ressembler à :

```
/usr/share/doc/
/usr/share/icons/
/usr/share/foo/
/usr/share/locale/
```

Ceci créerait deux paquets binaires, foo et foo-commun. Les deux auraient besoin de leur propre paragraphe dans debian/control.

Voyez « man dh_install » et la section d'installation de fichiers (article 5.11) du Guide du Nouveau Responsable Debian pour plus de détails.

Le fichier watch

Le fichier debian/watch nous permet de vérifier automatiquement les nouvelles versions en amont en utilisant l'outil uscan du paquet devscripts. La première ligne du fichier watch doit être la version du format (3, au moment d'écrire ces lignes), tandis que les lignes suivantes contiennent toutes les URL à traiter. Par exemple :

```
version=3
http://ftp.gnu.org/gnu/hello/
hello-(.*).tar.gz
```

Exécuter uscan dans le répertoire racine source comparera maintenant le numéro de la version en amont dans debian/changelog avec la dernière version en amont disponible. Si une nouvelle version est retrouvée en amont, elle sera automatiquement téléchargée. Par exemple :

```
$ uscan
hello: Nouvelle version (2.7)
disponible sur le site
distant :

http://ftp.gnu.org/gnu/hello/
hello-2.7.tar.gz
  (la version locale est la
2.6)
hello: téléchargement du
paquet hello-2.7.tar.gz mis à
jour réussi
    et création d'un lien
symbolique à
hello_2.7.orig.tar.gz
```

Pour plus d'informations, voyez « man uscan » et la section du fichier watch (Section 4.11) de la Charte Debian. Pour une liste des paquets pour lesquels le fichier watch signale qu'ils ne sont pas synchronisés avec l'amont, voir État de Santé Externe d'Ubuntu.

Le fichier source/format

Ce fichier indique le format du paquet source. Actuellement, la valeur par défaut du paquet source est 1.0 si ce fichier n'existe pas. Il est préférable d'utiliser le plus récent format de source 3.0. Dans ce cas, le fichier devrait contenir une seule ligne indiquant le format souhaité :

- 3.0 (native) pour les paquets Debian natifs (pas de version en amont) ou
- 3.0 (quilt) pour les paquets avec une archive distincte en amont

Si pour une raison quelconque vous souhaitez continuer à utiliser l'ancien format, vous êtes prié de créer ce fichier et d'y placer la valeur 1.0 pour être explicite au sujet de la version du paquet source. Ceci permettra la suppression future de la valeur par défaut de 1.0 pour le format paquet source.

http://wiki.debian.org/Projects/DebSrc3.0 résume l'information et les avantages du passage aux formats 3.0 pour le paquet source.

Voyez « man dpkg-source » et la section source/format (Section 5.21) du Guide du Nouveau Responsable Debian pour plus de détails.

Ressources supplémentaires

En plus des liens vers la Charte Debian dans chaque section ci-dessus, le Guide du Nouveau Responsable Debian contient des descriptions plus détaillées de chaque fichier. Le chapitre 4, « Fichiers obligatoires dans le répertoire debian » présente en détail les fichiers control, changelog, copy-right et rules. Le chapitre 5, « Autres fichiers dans le répertoire debian » parle des fichiers supplémentaires qui peuvent être utilisés.