

Debian Live Manual

Debian Live Project [\[debian-live@lists.debian.org\]](mailto:debian-live@lists.debian.org)

2015-08-23

Debian Live Manual

Debian Live Project ;debian-live@lists.debian.org;

Copyright © 2006-2015 Live Systems Project, Copyright © 2016-2025 The Debian Live team

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

The complete text of the GNU General Public License can be found in /usr/share/common-licenses/GPL-3 file.

Contents		Utilisateur	10
Debian Live Manual	i	Installation	11
propos	3	3. Installation	11
propos de ce manuel	4	3.1 Exigences	11
1. propos de ce manuel	4	3.2 Installation de live-build	11
1.1 Pour les impatientes	4	3.2.1 partir du dpt Debian	11
1.2 Terminologie	4	3.2.2 partir du code source	11
1.3 Auteurs	5	3.3 Installation de live-boot et live-config	12
1.4 Contribuer ce document	6	3.3.1 partir du dpt Debian	12
1.4.1 Appliquer des modifications	6	3.3.2 partir du code source	12
1.4.2 Traduction	7	Les bases	14
propos du Debian Live Project	8	4. Les bases	14
2. propos du Debian Live Project	8	4.1 Qu'est-ce qu'un systme live?	14
2.1 Motivation	8	4.2 Tlchargement des images precompiles	15
2.1.1 Ce qui ne va pas avec les systmes live actuels	8	4.3 Premires tapes: la construction d'une image ISO hybride	15
2.1.2 Pourquoi crer notre propre systme live?	8	4.4 Utilisation d'une image ISO hybride live	15
2.2 Philosophie	8	4.4.1 Graver une image ISO sur un support physique	15
2.2.1 Only unchanged packages from Debian main and non-free-firmware	8	4.4.2 Copie d'une image ISO hybride sur une cl USB	16
2.2.2 Pas de configuration des paquets du systme live	9	4.4.3 Utilisation de l'espace disponible sur une cl USB	16
2.3 Contact	9	4.4.4 Dmarrer le support live	16
		4.5 Utiliser une machine virtuelle pour les tests	17
		4.5.1 Test d'une image ISO avec QEMU	17
		4.5.2 Test d'une image ISO avec VirtualBox	18
		4.6 Construire et utiliser une image HDD	18
		4.7 Construction d'une image netboot	19
		4.7.1 Serveur DHCP	19

4.7.2 Serveur TFTP	20	Personnalisation des contenus	27
4.7.3 Serveur NFS	20		
4.7.4 Guide pratique pour exprimer avec une image Net- boot	21	7. Vue d'ensemble de la personnalisation	27
4.7.5 Qemu	21	7.1 Configuration pendant la construction vs. l'amorçage	27
4.8 Webbooting	21	7.2 tapes de la construction	27
4.8.1 Obtenir les fichiers webboot	21	7.3 Supplément lb config avec des fichiers	27
4.8.2 Démarrer images webboot	21	7.4 Tâches de personnalisation	28
Aperçu des outils	23	Personnalisation de l'installation de paquets	29
5. Aperçu des outils	23	8. Personnalisation de l'installation de paquets	29
5.1 Le paquet live-build	23	8.1 Sources des paquets	29
5.1.1 La commande lb config	23	8.1.1 Distribution, zones d'archive et mode	29
5.1.2 La commande lb build	24	8.1.2 Miroirs de distribution	30
5.1.3 La commande lb clean	24	8.1.3 Miroirs de distribution utilisés lors de la construction	30
5.2 Le paquet live-boot	24	8.1.4 Miroirs de distribution utilisés pendant l'exécution	30
5.3 Le paquet live-config	24	8.1.5 Dpts additionnels	30
		8.2 Choisir les paquets à installer	31
		8.2.1 Listes de paquets	31
		8.2.2 Utilisation des mtaquets	31
		8.2.3 Listes de paquets locaux	32
		8.2.4 Listes de paquets locaux pour l'tape binary	32
		8.2.5 Listes de paquets gnres	32
		8.2.6 Utiliser des conditions dans les listes de paquets	33
		8.2.7 Suppression de paquets lors de l'installation	33
		8.2.8 Summary	33
		8.2.9 Tâches de bureau et de langue	34
		8.2.10 Version et type de noyau	34
		8.2.11 Noyaux personnalisés	35
Gestion d'une configuration	25		
6. Gestion d'une configuration	25		
6.1 Gérer les modifications de la configuration	25		
6.1.1 Pourquoi utiliser des scripts auto? Que font-ils?	25		
6.1.2 Utiliser les scripts auto d'exemple	25		
6.2 Cloner une configuration publiée via Git	26		

8.3 Installation de paquets modifis ou tiers	35	10.3 Persistance	44
8.3.1 Utiliser packages.chroot pour installer des paquets personnalis	36	10.3.1 Le fichier persistence.conf	46
8.3.2 Utiliser un dpt APT pour installer des paquets per- sonnalis.	36	10.3.2 Utilisation de plusieurs dispositifs de persistance .	46
8.3.3 Les paquets personnalis et APT	36	10.3.3 Utilisation de la persistance avec chiffrement . . .	47
8.4 Configuration d'APT pendant la construction	37	Personnalisation de l'image binaire	49
8.4.1 Choisir apt ou aptitude	37	11. Personnalisation de l'image binaire	49
8.4.2 Utilisation d'un proxy avec APT	37	11.1 Chargeurs d'amorage	49
8.4.3 Rgler APT pour conomiser de l'espace	37	11.2 Mtadonnes ISO	49
8.4.4 Passer des options apt ou aptitude	38	Personnalisation de l'installateur Debian	50
8.4.5 APT pinning	38	12. Personnalisation du contenu pour l'installateur Debian .	50
Personnalisation des contenus	40	12.1 Types d'installateur Debian	50
9. Personnalisation des contenus	40	12.2 Personnalisation de l'installateur Debian par prconfigura- tion	51
9.1 Includes	40	12.3 Personnalisation de contenu pour l'Installateur Debian .	51
9.1.1 Live/chroot local includes	40	Projet	52
9.1.2 Binary local includes	41	Contribuer au projet	53
9.2 Hooks	41	13. Contribuer au projet	53
9.2.1 Chroot local hooks	41	13.1 Traduction des pages de manuel	53
9.2.2 Binary local hooks	41		
9.2.3 Hooks pendant le dmarrage	41		
9.3 Prconfigurer questions de debconf	42		
Personnalisation des comportements pendant l'excution	43		
10. Personnalisation des comportements pendant l'excution	43		
10.1 Personnalisation de l'utilisateur live	43		
10.2 Personnalisation des paramtres rgionaux et de la langue	43		

Signaler des bogues	54	Exemples	62
14. Signaler des bogues	54	16. Exemples	62
14.1 Problèmes connus	54	16.1 Utiliser les exemples	62
14.2 Effectuer une recherche	54	16.2 Tutoriel 1: Une image par défaut	62
14.3 Reconstruire partir de zéro	54	16.3 Tutoriel 2: Un utilitaire d'un navigateur Web	63
14.4 Utiliser des paquets mis à jour	55	16.4 Tutoriel 3: Une image personnalisée	63
14.5 Recueillir l'information	55	16.4.1 Première révision	63
14.6 Isoler le cas qui choue, si possible	55	16.4.2 Deuxième révision	64
14.7 Utiliser le paquet adéquat pour rapporter un bogue	56	16.5 Un client kiosque VNC	65
14.7.1 Pendant la construction durant l'amorçage	56	16.6 A minimal image for a 512MB USB key	66
14.7.2 Pendant la construction durant l'installation de paquets	56	16.7 Un bureau GNOME localisé avec un installateur	66
14.7.3 Pendant le démarrage	56	Appendix	68
14.7.4 Pendant l'exécution	56	Guide de style	69
14.8 Ordonner les bogues	56	17. Guide de style	69
Style de code	58	17.1 Lignes directrices pour les auteurs	69
15. Style du code	58	17.1.1 Caractéristiques linguistiques	69
15.1 Compatibilité	58	17.1.2 Procédures	71
15.2 Indentation	58	17.2 Lignes directrices pour les traducteurs	72
15.3 Adaptateur	58	17.2.1 Conseils de traduction	73
15.4 Variables	59	SiSU Metadata, document information	74
15.5 Autres	60		
Exemples	61		

₁ Debian Live Manual

propos

propos de ce manuel

1. propos de ce manuel

This manual serves as a single access point to all documentation related to the Debian Live Project and in particular applies to the software produced by the project for the Debian bookworm release. An up-to-date version can always be found at <https://live-team.pages.debian.net/live-manual/>

Ce manuel a principalement pour but de vous aider construire un système live et non pas de s'articuler autour des sujets relatifs à l'utilisateur final. Toutefois, l'utilisateur final peut trouver des informations utiles dans ces sections: **Les Bases** couvrent le téléchargement des images précompilées et la préparation des images pour être démarrées sur les supports ou sur le réseau, soit en utilisant le constructeur web, soit en exécutant live-build directement sur le système. **Personnalisation des comportements pendant l'exécution** décrit certaines options qui peuvent être indiquées à l'invite de démarrage, telles que la sélection d'un clavier, des paramètres régionaux et la persistance.

Certaines commandes mentionnées dans le texte doivent être exécutées avec les privilèges de super-utilisateur, qui peuvent être obtenus à l'aide de la commande `su` ou en utilisant `sudo`. Afin de distinguer les commandes qui peuvent être exécutées par un utilisateur sans privilège de celles nécessitant les privilèges de super-utilisateur, les commandes sont précédées respectivement par `$` ou `#`. Notez que ce symbole ne fait pas partie de la commande.

1.1 Pour les impatientes

Même si nous croyons que tout dans ce manuel est important pour au

moins certains de nos utilisateurs, nous nous rendons compte qu'il y a beaucoup de matière à couvrir et que vous pouvez vouloir exprimer avant d'entrer dans les détails. Par conséquent, nous vous suggérons de lire dans l'ordre suivant.

Tout d'abord, lisez ce chapitre **propos de ce manuel** dès le début et finissant avec la section **Terminologie**. Ensuite, passez aux trois tutoriels l'avant de la section **Exemples** destinée à vous apprendre la construction de l'image et les bases de la personnalisation. Lisez en premier **En utilisant les exemples**, puis **Tutoriel 1: Une image par défaut**, **Tutoriel 2: Un logiciel de navigateur Web** et finalement **Tutoriel 3: Une image personnalisée**. À la fin de ces tutoriels, vous aurez un avant-goût de ce qui peut être fait avec les systèmes live.

Nous vous encourageons à revenir à l'étude plus approfondie du manuel, en poursuivant par exemple votre lecture par **Les bases**, **Construire une image netboot** et finissant par la lecture de la **Vue d'ensemble de la personnalisation** et les sections suivantes. Après cela, nous espérons que vous serez complètement enthousiaste par ce qui peut être fait avec les systèmes live et motivés pour lire le reste du manuel, du début à la fin.

1.2 Terminologie

Système Live : Un système d'exploitation pouvant être démarré sans installation préalable sur un disque dur. Les systèmes live ne modifient pas le système d'exploitation local ou les fichiers installés sur le disque dur sans qu'on leur en donne explicitement l'instruction. Habituellement, les systèmes live sont démarrés à partir des supports tels que des CDs, DVDs, ou des clés USB. Certains systèmes peuvent également être démarrés sur le réseau (via des images netboot, voir **Construction d'une image netboot**), et sur l'Internet (via le paramètre d'amorçage `fetch=URL`, voir **Webbooting**).

14	Support live : la diffrence du systme live, le support live se rfere au CD, DVD ou cl USB o l'image binaire produite par live-build et utilise pour dmarrer le systme live est crite. D'une manire gnrale, le terme dsigne galement tout emplacement o rside l'excutable qui permet de dmarrer le systme live, tel que l'emplacement des fichiers de dmarrage sur le rseau.	26
15	Debian Live Project : Le projet qui maintient, entre autres, les paquets live-boot, live-build, live-config live-tools et live-manual.	
16	Systme hte : L'environnement utilis pour crer le systme live.	
17	Systme cible : L'environnement utilis pour faire fonctionner le systme live.	
18	live-boot : Une collection de scripts utilis pour lancer des systmes live.	
19	live-build : Une collection de scripts utilis pour construire des systmes live personnalis.	
20	live-config : Une collection de scripts utilis pour configurer un systme live pendant le processus d'amorage.	
21	live-tools : Une collection de scripts supplmentaires utilis pour effectuer des tches utiles dans un systme live en fonctionnement.	
22	live-manual : Ce document est maintenu dans un paquet nomm live-manual.	
23	Debian Installer (d-i) : Le systme d'installation officiel pour la distribution Debian.	
24	Paramtres d'amorage : Les paramtres qui peuvent tre entrs l'invite de dmarrage afin de modifier le noyau ou live-config.	
25	chroot : Le logiciel chroot, chroot(8), nous permet d'excuter plusieurs instances concurrentes de l'environnement GNU/Linux sur un systme sans redmarrage.	
	Binary image : A file containing the live system, such as live-image-amd64.hybrid.iso or live-image-amd64.img.	26
	Distribution cible : La distribution sur laquelle votre systme live sera base. Celle-ci peut tre diffrente de la distribution de votre systme hte.	27
	stable/testing/unstable : La distribution stable , actuellement nomme bookworm , contient la dernire version officielle de Debian. La distribution testing , temporairement nomme trixie , est la prochaine version stable o seulement les paquets suffisamment matures peuvent entrer. Un avantage de cette distribution est qu'elle contient des logiciels de versions plus rcentes que la version stable . La distribution unstable , nomme sid de faon permanente, est en constante volution. En gnral cette distribution est utilise par les dveloppeurs et ceux qui aiment le risque. Tout au long du manuel, nous avons tendance utiliser les noms de code pour les volutions majeures, tels que trixie ou sid , car c'est ce qui est pris en charge par les outils eux-mmes.	28
	1.3 Auteurs	29
	La liste des auteurs (dans l'ordre alphabtique):	30
	Ben Armstrong	31
	Brendan Sleight	32
	Carlos Zuferri	33
	Chris Lamb	34
	Daniel Baumann	35
	Franklin Piat	36
	Jonas Stein	37
	Kai Hendry	38
	Marco Amadori	39

Mathieu Geli
 Matthias Kirschner
 Richard Nelson
 Roland Clobus
 Trent W. Buck

1.4 Contribuer ce document

Ce manuel est conçu comme un projet communautaire et toutes les propositions d'améliorations et de contributions sont bienvenues. Veuillez consulter la section **Contribuer au projet** pour des informations détaillées sur la façon d'obtenir une copie et de faire des livraisons (commits).

1.4.1 Appliquer des modifications

Afin d'apporter des modifications au manuel anglais, vous devez modifier les fichiers adéquats dans `manual/en/` mais avant de soumettre votre contribution, veuillez prévisualiser votre travail. Afin de prévisualiser `live-manual`, assurez-vous que les paquets nécessaires sont installés en exécutant :

```
# apt-get install make po4a ruby ruby-nokogiri sisu-complete
```

Vous pouvez compiler `live-manual` dans le répertoire de niveau supérieur de votre Git checkout en exécutant :

```
$ make build
```

Comme il faut un certain temps pour construire le manuel dans toutes les langues prises en charge, les auteurs peuvent trouver pratique d'utiliser l'un des raccourcis de construction rapide lors de la révision de la nouvelle documentation ajoutée au manuel anglais. `PROOF=1` construit `live-manual` au format `html`, mais sans les fichiers `html` segments, et `PROOF=2` construit `live-manual` au format `pdf`, mais seulement les portraits A4 et US. C'est pourquoi l'utilisation de l'une ou l'autre des possibilités peut sauver une quantité considérable de temps, par exemple :

```
$ make build PROOF=1
```

Lors de la révision d'une des traductions, il est possible de construire une seule langue en exécutant, par exemple :

```
$ make build LANGUAGES=de
```

Il est également possible de construire par type de document, par exemple,

```
$ make build FORMATS=pdf
```

Ou combiner les deux, par exemple :

```
$ make build LANGUAGES=de FORMATS=html
```

Après avoir relu votre travail et vous être assuré que tout va bien, n'utilisez pas `make commit` moins que vous mettiez à jour les traductions dans le commit. Dans ce cas, ne mélangez pas les modifications apportées au manuel en anglais et les traductions dans la même livraison, mais

utilisez des commits spars. Consultez la section [Traduction](#) pour plus de dtails.

1.4.2 Traduction

Note: Pour la traduction des pages de manuel, voir Traduction des pages de manuel

Afin de traduire live-manual, procdez comme suit, selon que vous commencez une traduction partir de zro ou vous travaillez sur une traduction dj existante:

Commencer une nouvelle traduction partir de zro

Traduisez les fichiers about`manual.ssi.pot , about`project.ssi.pot et index.html.in.pot dans manual/pot/ dans votre langue avec votre diteur prfr (comme poedit) . Envoyez les fichiers .po traduits la liste de diffusion pour vrifier leur intgrit. La vrification d'intgrit de live-manual garantit non seulement que les fichiers .po sont 100% traduits, mais elle dtecte galement des erreurs possibles.

Once checked, to enable a new language in the autobuild it is enough to add the initial translated files to manual/po/\$-LANGUAGE"/ and edit manual`sisu/home/index.html adding the name of the language and its name in English between brackets. And then, add the folder manual/\$-LANGUAGE"/ to the file .gitignore. Finally, run make commit.

Continuer avec une traduction dj commence

Si votre langue cible a dj t ajoute, vous pouvez continuer avec la traduction des fichiers .po dans manual/po/\$-LANGUAGE"/ de faon alatoire avec votre diteur prfr (comme poedit) .

N'oubliez pas que vous devez faire un make commit pour assurer que la traduction des manuels est mise jour partir des fichiers .po,

alors vous pouvez rviser vos modifications avec make build avant git add ., git commit -m Translating... et git push. Gardez l'esprit que make build peut prendre un temps considrable, vous pouvez relire les langues individuellement comme expliqu dans [Appliquer des modifications](#)

Aprs l'excution de make commit, vous verrez beaucoup de texte sur l'cran. Il s'agit essentiellement de messages informatifs sur l'tat du processus et de quelques indications sur ce qui peut tre fait pour amliorer live-manual. Si vous ne voyez aucune erreur fatale, vous pouvez gnralement continuer et soumettre votre contribution.

live-manual contient deux utilitaires qui peuvent grandement aider les traducteurs trouver les textes non traduits et modifis. Le premier est make translate. Il lance un script qui vous indique en dtail le nombre de messages non traduits qu'il y a dans chaque fichier .po. Le second, make fixfuzzy, n'agit que sur les messages modifis, mais il vous aide les trouver et les rsoudre un par un.

Gardez l'esprit que mme si ces utilitaires peuvent tre vraiment utiles pour faire un travail de traduction sur la ligne de commande, l'utilisation d'un outil spcialis comme poedit est la mthode recommande pour effectuer la tche. C'est aussi une bonne ide de lire la documentation sur localisation de debian (l10n) et, plus particulirement pour live-manual, les [Lignes directrices pour les traducteurs](#).

Remarque: Vous pouvez utiliser make clean pour nettoyer votre arbre git avant de faire un push. Cette tape n'est pas obligatoire grce au fichier .gitignore mais c'est une bonne pratique pour viter d'envoyer certains fichiers involontairement.

propos du Debian Live Project ⁸⁷

⁸⁸

2. propos du Debian Live Project

2.1 Motivation

2.1.1 Ce qui ne va pas avec les systmes live actuels

When Debian Live Project was initiated (around 2006), there were already several Debian based live systems available and they are doing a great job. From the Debian perspective most of them have one or more of the following disadvantages:

Ce ne sont pas des projets Debian et ils manquent donc de soutien au sein de Debian.

Ils mlangent des distributions diffrentes comme testing et unstable .

Ils ne prennent en charge que i386.

Ils modifient le comportement et/ou l'apparence des paquets en les dpouillant pour conomiser de l'espace.

Ils comprennent des paquets ne provenant pas de l'archive Debian.

Ils offrent des noyaux personnalis avec des correctifs supplmentaires qui ne font pas partie de Debian.

Ils sont gros et lents en raison de leur dimension et donc pas recommandés comme systmes de sauvetage.

Ils ne sont pas disponibles en diffrents formats (CDs, DVDs, clés USB et images netboot).

2.1.2 Pourquoi crer notre propre systme live?

Debian est le systme d'exploitation universel: Debian a un systme live pour servir de vitrine et pour représenter le vrai, seul et unique systme Debian avec les principaux avantages suivants:

C'est un sous-projet de Debian.

Il reflète l'état (actuel) d'une distribution.

Il fonctionne sur le plus grand nombre d'architectures possible.

Il ne se compose que de paquets Debian inchangés.

Il ne contient pas de paquets qui n'appartiennent pas à l'archive Debian.

Il utilise un noyau Debian inchang, sans correctifs supplémentaires.

2.2 Philosophie

2.2.1 Only unchanged packages from Debian main and non-free-firmware

Nous n'utiliserons que les paquets du dépôt Debian dans la section main. La section non-free ne fait pas partie de Debian et ne peut donc pas être utilisée pour les images officielles du système live.

Starting with Debian 12 bookworm we added the [non-free-firmware](#) section for better support of modern hardware.

Nous ne changerons pas les paquets. Chaque fois que nous aurons besoin de changer quelque chose, nous le ferons en coordination avec le responsable du paquet dans Debian.

À titre d'exception, nos propres paquets tels que live-boot, live-build ou live-config peuvent être utilisés temporairement à partir de notre propre dépôt pour des raisons de développement (par exemple pour créer des instantanés de développement). Ils seront téléchargés sur Debian régulièrement.

2.2.2 Pas de configuration des paquets du systme live

Dans cette phase, nous n'offrirons pas de configurations alternatives. Tous les paquets sont utilis dans leur configuration par dfaut comme ils sont aprs une installation standard de Debian.

Chaque fois que nous aurons besoin d'une configuration par dfaut diffrente, nous la ferons en coordination avec le responsable du paquet dans Debian.

Un systme de configuration des paquets est fourni avec debconf permettant la personnalisation des paquets installs sur vos images live, mais pour les **images live prcompiles** seulement une configuration par dfaut sera utilise sauf si c'est absolument ncessaire pour fonctionner dans l'environnement live. Autant que possible, nous prfrons adapter les paquets dans l'archive Debian de sorte qu'ils fonctionnent mieux dans un systme live plutt que faire des changements l'ensemble d'outils live ou les **configurations des images live**. Pour plus d'informations, veuillez consulter **Vue d'ensemble de la personnalisation**.

2.3 Contact

Mailing list : The primary contact for the project is the mailing list at <https://lists.debian.org/debian-live/>. You can email the list directly by addressing your mail to debian-live@lists.debian.org. The list archives are available at <https://lists.debian.org/debian-live/>.

IRC : Un certain nombre d'utilisateurs et de dveloppeurs sont prsents dans le canal **#debian-live** sur irc.debian.org (OFTC). Quand vous posez une question sur IRC, s'il vous plat soyez patient en attendant une rponse. Si aucune rponse n'est donne, veuillez envoyer un courriel la liste de diffusion.

BTS : Le **Rapporter des bogues**.

Installation

3. Installation

3.1 Exigences

Building live system images has very few system requirements for the host system:

Accs super-utilisateur (root)

Une version mise à jour de live-build

Un shell POSIX, comme bash ou dash

debootstrap

Linux 2.6 or newer

A mount point with dev and exec rights.

```
# mount -i your-mount-point -o dev,exec,remount
```

Notez que l'utilisation de Debian ou d'une distribution dérivée de Debian n'est pas nécessaire - live-build fonctionne sur presque toutes les distributions remplissant les exigences ci-dessus.

3.2 Installation de live-build

Vous pouvez installer live-build d'un certain nombre de façons différentes:

partir du dépôt Debian

partir du code source

partir des instantanés

Si vous utilisez Debian, la méthode recommandée consiste à installer live-build à partir du dépôt Debian.

3.2.1 partir du dépôt Debian

Il suffit d'installer live-build comme n'importe quel autre paquet:

```
# apt-get install live-build
```

3.2.2 partir du code source

live-build est développé en utilisant le système de contrôle de version Git. Dans les systèmes basés sur Debian, il est fourni par le paquet git. Pour examiner le dernier code, exécutez:

```
$ git clone https://salsa.debian.org/live-team/live-build.git
```

Vous pouvez compiler et installer votre propre paquet Debian en exécutant:

```
$ cd live-build
$ dpkg-buildpackage -b -uc -us
$ cd ..
```

Maintenant, installez les fichiers récemment construits qui vous intéressent, par exemple

```
# dpkg -i live-build_4.0-1_all.deb
```


Vous pouvez galemment installer live-build directement sur votre systme en excutant:

```
$ git clone https://salsa.debian.org/live-team/live-boot.git
$ git clone https://salsa.debian.org/live-team/live-config.git
```

```
# make install
```

et le dsinstaller avec:

```
# make uninstall
```

Consultez les pages de manuel de live-boot et live-config pour plus de dtails sur la personnalisation si la raison pour laquelle vous crez vos paquets partit des sources.

Crer les fichiers .deb de live-boot et live-config

Vous devez crer sur votre distribution cible ou dans un chroot contenant votre plateforme cible: cela signifie que si votre cible est trixie alors vous devez crer sur trixie .

3.3 Installation de live-boot et live-config

Remarque: Vous n'avez pas besoin d'installer live-boot ou live-config sur votre systme afin de crer des systmes live. Cependant, cela ne fera aucun mal et est utile des fins de rfrence. Si vous voulez seulement la documentation, vous pouvez maintenant installer les paquets live-boot-doc et live-config-doc sparment.

Utilisez un systme de construction automatique personnel tel que pbuilder ou sbuild si vous avez besoin de crer live-boot pour une distribution cible qui diffre de votre systme de construction. Par exemple, pour les images live de trixie , crez live-boot dans un chroot trixie . Si votre distribution cible correspond votre distribution vous pouvez crer directement sur le systme de construction en utilisant dpkg-buildpackage (fourni par le paquet dpkg-dev) :

3.3.1 partir du dpt Debian

live-boot et live-config sont tous les deux disponibles dans le dpt Debian comme expliqu dans [Installation de live-build](#).

```
$ cd live-boot
$ dpkg-buildpackage -b -uc -us
$ cd ../live-config
$ dpkg-buildpackage -b -uc -us
```

3.3.2 partir du code source

Pour utiliser les dernires sources du git, vous pouvez suivre la procudre ci-dessous. Veuillez vous assurer que vous tes familiaris avec les termes mentionns dans [Termes](#).

Examiner les sources de live-boot et live-config

Utiliser les fichiers .deb gnrs ncessaires.

Comme live-boot et live-config sont installs par le systme de construction live-build, l'installation de ces paquets dans le systme hte ne suffit pas: vous devez traiter les fichiers .deb gnrs comme d'autres paquets personnalis. Comme votre objectif pour la construction partit du code source est de tester de nouvelles choses court terme avant leur publication officielle, suivez [Installation de paquets modifis ou de tiers](#) pour inclure

temporairement les fichiers pertinents dans votre configuration. En particulier, remarquez que les deux paquets sont divisés en une partie générique, une partie de documentation et un ou plusieurs back-ends. Incluez la partie générique, un seul back-end et éventuellement la documentation. En supposant que vous construisiez une image live dans le répertoire courant et ayez généré tous les fichiers .deb pour une version unique des deux paquets dans le répertoire ci-dessus, ces commandes bash copieraient tous les paquets appropriés, y compris les back-ends par défaut :

157

```
$ cp ../live-boot-*, -initramfs-tools, -doc "*.deb config/packages.↵
chroot/
$ cp ../live-config-*, -sysvinit, -doc "*.deb config/packages.chroot↵
/
```

Les bases

4. Les bases

Ce chapitre contient un bref aperçu du processus de construction et des instructions pour utiliser les trois types d'images les plus couramment utilisés. Le type d'image le plus polyvalent, iso-hybrid, peut être utilisé sur une machine virtuelle, un support optique ou un périphérique USB de stockage portable. Dans certains cas particuliers, comme expliqué plus loin, le type hdd peut être plus approprié. Le chapitre contient des instructions détaillées pour la construction et l'utilisation d'une image netboot, qui est un peu plus compliquée en raison de la configuration requise sur le serveur. C'est un sujet un peu avancé pour tous ceux qui ne connaissent pas déjà le démarrage sur le réseau, mais est inclus ici car une fois la configuration terminée, c'est un moyen très pratique pour tester et déployer des images pour le démarrage sur le réseau local sans le tracas des supports d'images.

La section se termine par une brève introduction au webbooting qui est, peut-être, la meilleure façon d'utiliser des images différentes à des fins différentes, changeant de l'une à l'autre en fonction des besoins en utilisant l'Internet comme un moyen.

Tout au long du chapitre, nous ferons souvent référence à la valeur par défaut des noms des fichiers produits par live-build. Si vous **téléchargez une image prcompilé**, les noms des fichiers peuvent varier.

4.1 Qu'est-ce qu'un système live?

Un système live signifie généralement un système d'exploitation démarré sur un ordinateur à partir d'un support amovible, tel qu'un CD-ROM, une

clé USB ou sur un réseau, permettant l'emploi sans aucune installation sur le disque habituel, avec auto-configuration faite lors de l'exécution (voir **Termes**).

With live systems, it's an operating system, built for one of the supported architectures (currently amd64 and arm64). It is made from the following parts:

Image du noyau Linux, d'habitude appelée `vmlinuz*`

Image du RAM-disque initiale (`initrd`) : Un disque virtuel RAM configuré pour le démarrage de Linux, contenant possiblement des modules nécessaires pour monter l'image du système et certains scripts pour le faire.

Image du système : L'image du système de fichiers du système d'exploitation. Habituellement, un système de fichiers SquashFS comprimé est utilisé pour réduire au minimum la taille de l'image live. Notez qu'il est en lecture seule. Ainsi, lors du démarrage le système live va utiliser un disque RAM et un mécanisme union pour permettre l'écriture de fichiers dans le système en marche. Cependant, toutes les modifications seront perdues lors de l'arrêt, à moins que l'option persistance soit utilisée (voir **Persistance**).

Chargeur d'amorçage : Un petit morceau de code conçu pour démarrer à partir du support choisi, il peut présenter un menu rapide ou permettre la sélection des options/configurations. Il charge le noyau Linux et son `initrd` pour fonctionner avec un système de fichiers associé. Différentes solutions peuvent être utilisées, selon le support de destination et le format du système de fichiers contenant les composants mentionnés précédemment: `isolinux` pour démarrer à partir d'un CD ou DVD au format ISO9660, `syslinux` pour démarrer un disque dur ou une clé USB à partir d'une partition VFAT, `extlinux` pour partitions `ext2/3/4` et `btrfs`, `pxelinux` pour netboot PXE, GRUB pour partitions `ext2/3/4`, etc.

You can use live-build to build the system image from your specifications, set up a Linux kernel, its initrd, and a bootloader to run them, all in one medium-dependent format (ISO9660 image, disk image, etc.).

Ensuite, exécutez la commande `lb config`. Cela va créer une hiérarchie `config/` dans le répertoire courant pour être utilisée par d'autres commandes:

```
$ lb config
```

4.2 Téléchargement des images précompilées

You can download one of the prebuilt images from <https://www.debian.org/CD/live/>. For many of the popular desktop environments (GNOME, Xfce, KDE, etc.) a specific live image is prepared.

Aucun paramètre n'est passé à ces commandes, donc les défauts seront utilisés pour l'ensemble de ses diverses options. Consultez [La commande lb config](#) pour plus de détails.

If you are unsure which file to download, use the 'Live GNOME' image from the 'stable' release. You can then skip reading the next sections and run the image in a [virtual machine](#).

Maintenant que la hiérarchie `config/` existe, créez l'image avec la commande `lb build` :

```
# lb build
```

4.3 Premières tapes: la construction d'une image ISO hybride

Quel que soit le type d'image, vous devrez effectuer les mêmes tapes de base pour créer une image chaque fois. Comme premier exemple, créez un répertoire de travail, passez dans ce répertoire et exécutez la séquence suivante de commandes live-build pour créer une image ISO hybride de base contenant tout le système Debian par défaut sans X.org. Elle est appropriée pour être gravée sur CD ou DVD, et peut également être copiée sur une clé USB.

This process can take a while, depending on the speed of your computer and your network connection. When it is complete, there should be a `live-image-amd64.hybrid.iso` image file, ready to use, in the current directory.

Remarque: Si vous construisez sur un système amd64 le nom de l'image résultante sera `live-image-amd64.hybrid.iso`. Gardez l'esprit cette convention de nommage tout au long du manuel.

Le nom du répertoire de travail dépend totalement de vous, mais si vous jetez un œil aux exemples utilisés dans `live-manual`, c'est une bonne idée d'utiliser un nom qui vous aide à identifier l'image avec laquelle vous travaillez dans chaque répertoire, surtout si vous travaillez ou expérimentez avec différents types d'images. Dans ce cas, vous allez construire un système par défaut, nous allons donc l'appeler, par exemple, `live-default`.

4.4 Utilisation d'une image ISO hybride live

After either building or downloading an ISO hybrid image the usual next step is to prepare your medium for booting, either CD-R(W) or DVD-R(W) optical media or a USB stick.

4.4.1 Graver une image ISO sur un support physique

Graver une image ISO est facile. Il suffit d'installer `xorriso` et de

```
$ mkdir live-default && cd live-default
```

l'utiliser partir de la ligne de commande pour graver l'image. Par exemple:

```
# apt-get install xorriso
$ xorriso -as cdrecord -v dev=/dev/sr0 blank=as-needed live-image-↔
amd64.hybrid.iso
```

partition on the device will be filled up by the live system. To use the remaining free space, use a partitioning tool such as gparted or parted to create a new partition on the stick.

```
# gparted $-USBSTICK"
```

4.4.2 Copie d'une image ISO hybride sur une clé USB

Les images ISO préparées avec xorriso peuvent être simplement copiées sur une clé USB avec cp ou un logiciel équivalent. Branchez une clé USB avec une capacité suffisamment grande pour votre fichier image et déterminez quel périphérique elle est, que nous appelons ci-dessous `$-USBSTICK`". C'est le fichier de périphérique de votre clé, tel que `/dev/sdb`, pas une partition, telle que `/dev/sdb1`! Vous pouvez trouver le nom du périphérique en regardant la sortie de `dmesg` après avoir branché le périphérique, ou mieux encore, `ls -l /dev/disk/by-id`.

Une fois que vous êtes sûr d'avoir le nom correct de l'appareil, utilisez la commande `cp` pour copier l'image sur la clé. Ceci écrasera tout fichier déjà existant sur votre clé!

```
$ cp live-image-amd64.hybrid.iso $-USBSTICK"
$ sync
```

Remarque: La commande `sync` est utilisée pour s'assurer que toutes les données qui sont stockées dans la mémoire par le noyau lors de la copie de l'image, sont écrites sur la clé USB.

4.4.3 Utilisation de l'espace disponible sur une clé USB

After copying the `live-image-amd64.hybrid.iso` to a USB stick, the first

Quand la partition est créée, vous devez y créer un système de fichiers où `$-PARTITION`" est le nom de la partition, comme `/dev/sdb2`. Un choix possible serait `ext4`.

```
# mkfs.ext4 $-PARTITION"
```

Remarque: Si vous voulez utiliser l'espace supplémentaire avec Windows, ce système d'exploitation ne peut accéder normalement à aucune partition part la première. Certaines solutions à ce problème ont été discutées sur notre [liste de diffusion](#), mais il semble qu'il n'y ait pas de réponse facile.

Remember: Every time you install a new `live-image-amd64.hybrid.iso` on the stick, all data on the stick will be lost because the partition table is overwritten by the contents of the image, so back up your extra partition first to restore again after updating the live image.

4.4.4 Démarrer le support live

La première fois que vous démarrez votre support live, qu'il s'agisse de CD, DVD, clé USB, ou du démarrage par PXE, une certaine configuration dans le BIOS de votre ordinateur peut être d'abord nécessaire. Puisque les BIOS varient grandement en fonctionnalités et raccourcis clavier, on ne peut pas aborder le sujet en profondeur ici. Certains BIOS fournissent une touche pour ouvrir un menu d'amorçage au démarrage, qui est le moyen le plus facile si elle est disponible sur votre système. Sinon, vous avez besoin

d'entrer dans le menu de configuration du BIOS et modifier l'ordre de dmarrage pour placer le dispositif de dmarrage pour le systme live devant votre priphrique de dmarrage normal.

Une fois que vous avez dmarr le support, un menu de dmarrage vous est prsent. Si vous appuyez simplement sur entre ici, le systme va dmarrer en utilisant l'entre par dfaut, Live. Pour plus d'informations sur les options de dmarrage, consultez l'entre Help dans le menu et aussi les pages de manuel de live-boot et live-config dans le systme live.

Assuming you've selected Live and booted a default desktop live image, after the boot messages scroll by, you should be automatically logged into the user account and see a desktop, ready to use. If you have booted a console-only image, you should be automatically logged in on the console to the user account and see a shell prompt, ready to use.

4.5 Utiliser une machine virtuelle pour les tests

Excuter les images live dans une machine virtuelle (VM) peut faire gagner beaucoup de temps. Cela ne vient pas sans avertissements:

L'excution d'une VM demande assez de RAM pour le systme d'exploitation client et l'hte et un CPU avec support matriel pour la virtualisation est recommand.

Il y a quelques limitations inhrentes l'excution sur une VM, par exemple des performances vido mdiocres, ou un choix limit de matriel mul.

Lors du dveloppement d'un matriel spcifique, il n'existe aucun substitut pour l'excution que le matriel lui-mme.

Certains ne deviennent visibles que pendant l'excution dans une VM. En cas de doute, testez votre image directement sur le matriel.

condition de pouvoir travailler avec ces contraintes, examinez les logiciels VM disponibles et choisissez celui qui convient vos besoins.

4.5.1 Test d'une image ISO avec QEMU

La VM la plus polyvalente de Debian est QEMU. Si votre processeur dispose d'une gestion matrielle de la virtualisation, vous pouvez utiliser le paquet qemu-kvm; La description du paquet qemu-kvm numre brievement les exigences.

Tout d'abord, installez qemu-kvm si votre processeur le gre. Sinon, installez qemu. Dans ce cas, le nom du programme est qemu au lieu de kvm dans les exemples suivants. Le paquet qemu-utils est galement valable pour crer des images de disques virtuels avec qemu-img.

```
# apt-get install qemu-kvm qemu-utils
```

Dmarrer une image ISO est simple:

```
$ kvm -cdrom live-image-amd64.hybrid.iso -m 4G
```

Voir les pages de manuel pour plus de dtails.

Note: For live systems containing a desktop environment that you want to test with qemu, you may wish to include the spice-vdagent package in your live-build configuration. This will automatically adjust the resolution and enable the clipboard between the virtual machine and the host.

```
$ echo "spice-vdagent" >> config/package-lists/spice.list.chroot
```

4.5.2 Test d'une image ISO avec VirtualBox

Afin de tester l'ISO avec virtualbox:

```
# apt-get install virtualbox virtualbox-qt virtualbox-dkms
$ virtualbox
```

Create a new virtual machine, change the storage settings to use live-image-amd64.hybrid.iso as the CD/DVD device, and start the machine.

Remarque: Pour les systmes live contenant X.org que vous voulez essayer avec virtualbox, vous pouvez inclure le paquet des pilotes VirtualBox X.org, virtualbox-guest-dkms et virtualbox-guest-x11, dans votre configuration de live-build. Sinon, la rsolution est limite 800x600.

```
$ echo "virtualbox-guest-dkms virtualbox-guest-x11" >> config/
package-lists/my.list.chroot
```

Pour faire fonctionner le paquet dkms, il faut galement installer le paquet linux-headers pour le noyau utilis dans l'image. Au lieu de lister manuellement le paquet linux-headers correct dans la liste de paquets cre ci-dessus, live-build peut faire cela automatiquement.

```
$ lb config --linux-packages "linux-image linux-headers"
```

4.6 Construire et utiliser une image HDD

Building an HDD image is similar to an ISO hybrid one in all respects except you specify -b hdd and the resulting filename is live-image-amd64.img which cannot be burnt to optical media. It is suitable for

booting from USB sticks, USB hard drives, and various other portable storage devices. Normally, an ISO hybrid image can be used for this purpose instead, but if you have a BIOS which does not handle hybrid images properly, you need an HDD image.

Remarque: si vous avez cr une image ISO hybride avec l'exemple prcdent, vous devrez nettoyer votre rpertoire de travail avec la commande lb clean (voir [La commande lb clean](#)):

```
# lb clean --binary
```

Excutez la commande lb config comme avant, sauf que cette fois en spcifiant le type d'image HDD:

```
$ lb config -b hdd
```

Construisez maintenant l'image avec la commande lb build

```
# lb build
```

When the build finishes, a live-image-amd64.img file should be present in the current directory.

The generated binary image contains a VFAT partition and the syslinux bootloader, ready to be directly written on a USB device. Once again, using an HDD image is just like using an ISO hybrid one on USB. Follow the instructions in [Using an ISO hybrid live image](#), except use the filename live-image-amd64.img instead of live-image-amd64.hybrid.iso.

Likewise, to test an HDD image with Qemu, install qemu as described above in [Testing an ISO image with QEMU](#). Then run kvm or qemu,

depending on which version your host system needs, specifying live-image-amd64.img as the first hard drive.

```
$ kvm -hda live -image -amd64.img
```

4.7 Construction d'une image netboot

La squence de commandes suivante va crer une image NetBoot de base contenant le systme Debian par dfaut sans X.org. Elle peut tre dmarre sur le rseau.

Remarque: Si vous avez ralis quelques-uns des exemples prcdents, vous aurez besoin de nettoyer votre rpertoire de travail avec la commande lb clean:

```
# lb clean
```

Dans ce cas spcifique, un lb clean --binary ne serait pas suffisant pour nettoyer les tapes ncessaires. La raison est que dans les configurations de netboot, une configuration initramfs diffrente doit tre utilise, laquelle live-build excute automatiquement lors de la construction des images netboot. Puisque la cration d'initramfs appartient l'tape chroot, si on change netboot dans un rpertoire existant, il faut reconstruire le chroot. Par consquent, il faut faire un lb clean, (qui permettra d'liminer l'tape chroot, aussi).

Excutez la commande suivante pour configurer votre image pour dmarrer sur le rseau:

```
$ lb config -b netboot --net-root-path "/srv/debian-live" --net-root-server "192.168.0.2"
```

Contrairement aux images ISO et HDD, le dmarrage sur le rseau ne sert pas l'image du systme de fichiers pour le client. Pour cette raison, les fichiers doivent tre servis via NFS. Diffrents systmes de fichiers rseau peuvent tre choisis avec lb config. Les options --net-root-path et --net-root-server indiquent l'emplacement et le serveur, respectivement, du serveur NFS sur lequel l'image du systme de fichiers sera situe au moment du dmarrage. Assurez-vous que ceux-ci sont fixes des valeurs appropries pour votre rseau et serveur.

Construisez maintenant l'image avec la commande lb build

```
# lb build
```

Dans un dmarrage rseau, le client excute un petit morceau de logiciel qui rside habituellement sur l'EPROM de la carte Ethernet. Ce programme envoie une requete DHCP pour obtenir une adresse IP et les informations sur ce qu'il faut faire ensuite. Typiquement, la prochaine tape est d'obtenir un chargeur d'amorage de niveau suprieur via le protocole TFTP. Cela pourrait tre pxelinux, GRUB, ou dmarrer directement un systme d'exploitation comme Linux.

For example, if you unpack the generated live-image-amd64.netboot.tar archive in the /srv/debian-live directory, you'll find the filesystem image in live/filesystem.squashfs and the kernel, initrd and pxelinux bootloader in tftpboot/.

Nous devons maintenant configurer trois services sur le serveur pour activer le dmarrage sur le rseau: le serveur DHCP, le serveur TFTP et le serveur NFS.

4.7.1 Serveur DHCP

Nous devons configurer le serveur DHCP de notre rseau pour tre sr de

donner une adresse IP au client du système du démarrage sur le réseau, et pour annoncer l'emplacement du chargeur d'amorçage PXE.

Voici un exemple source d'inspiration, crit pour le serveur ISC DHCP `isc-dhcp-server` dans le fichier de configuration `/etc/dhcp/dhcpd.conf`:

```
# /etc/dhcp/dhcpd.conf - configuration file for isc-dhcp-server

ddns-update-style none;

option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

log-facility local7;

subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.1 192.168.0.254;
    filename "pxelinux.0";
    next-server 192.168.0.2;
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.0.255;
    option routers 192.168.0.1;
}
```

4.7.2 Serveur TFTP

Cela sert le noyau et le ramdisk initial pour le système pendant l'exécution.

Vous devriez installer le paquet `tftpd-hpa`. Il peut servir tous les fichiers contenus dans un répertoire racine, d'habitude `/srv/tftp`. Pour le laisser servir des fichiers dans `/srv/debian-live/tftpbboot`, exécutez comme utilisateur `root` la commande suivante:

```
# dpkg-reconfigure -plow tftpd-hpa
```

et remplissez le nouveau répertoire du serveur tftp

4.7.3 Serveur NFS

Quand l'ordinateur a chargé et démarré un noyau Linux et chargé son `initrd`, il va essayer de monter l'image du système de fichiers live via un serveur NFS.

Vous devez installer le paquet `nfs-kernel-server`.

Ensuite, rendez l'image du système de fichiers disponible via NFS en ajoutant une ligne comme la suivante `/etc/exports`:

```
/srv/debian-live *(ro,async,no'root'squash,no'subtree'check)
```

et indiquez cette exportation au serveur NFS avec la commande suivante:

```
# exportfs -rv
```

Setting up these three services can be a little tricky. You might need some patience to get all of them working together. For more information, see the syslinux wiki at <https://wiki.syslinux.org/wiki/index.php?title=PXELINUX> or the Debian Installer Manual's TFTP Net Booting section at <https://www.debian.org/releases/stable/amd64/ch04s05.en.html>. They might help, as their processes are very similar.

4.7.4 Guide pratique pour exprimer avec une image Netboot

La cration d'images netboot est facile avec live-build, mais les tests des images sur des machines physiques peuvent prendre vraiment beaucoup de temps.

Afin de rendre notre vie plus facile, nous pouvons utiliser la virtualisation.

4.7.5 Qemu

Installer qemu, bridge-utils, sudo.

diter /etc/qemu-ifup:

```
#!/bin/sh
sudo -p "Password for $0:" /sbin/ifconfig $1 172.20.0.1
echo "Executing /etc/qemu-ifup"
echo "Bringing up $1 for bridged mode..."
sudo /sbin/ifconfig $1 0.0.0.0 promisc up
echo "Adding $1 to br0..."
sudo /usr/sbin/brctl addif br0 $1
sleep 2
```

Obtenir, ou construire un grub-floppy-netboot.

Lancer qemu avec -net nic,vlan=0 -net tap,vlan=0,ifname=tun0

4.8 Webbooting

Webbooting est une manire trs pratique de tlcharger et d'amorcer les systmes live en utilisant l'Internet comme un moyen. Il ya trs peu d'exigences pour webbooting. D'une part, vous avez besoin d'un support avec un chargeur d'amorage, un disque ram initial et un noyau. D'autre part, un serveur web pour stocker les fichiers squashfs qui contiennent le systme de fichiers.

4.8.1 Obtenir les fichiers webboot

As usual, you can build the images yourself or use the **prebuilt files**. Using prebuilt images would be handy for doing initial testing until one can fine tune their own needs. If you have built a live image you will find the files needed for webbooting in the build directory under binary/live/. The files are called vmlinuz, initrd.img and filesystem.squashfs.

Il est galement possible d'extraire les fichiers d'une image iso dj existant. Pour ce faire, on doit monter l'image comme suit:

```
# mount -o loop image.iso /mnt
```

The files are to be found under the live/ directory. In this specific case, it would be /mnt/live/. This method has the disadvantage that you need to be root to be able to mount the image. However, it has the advantage that it is easily scriptable and thus, easily automated.

Mais sans aucun doute, la meilleure faon d'extraire les fichiers d'une image iso et les tlcharger sur le serveur web au mme temps, est d'utiliser le midnight commander ou mc. Si vous avez le paquet genisoimage install, le gestionnaire de fichiers deux panneaux vous permet de voir le contenu d'un fichier iso dans un panneau et de tlcharger les fichiers via FTP dans l'autre panneau. Mme si cette mthode ncessite un travail manuel, elle ne ncessite pas les privilges root.

4.8.2 Dmarrer images webboot

Tandis que certains utilisateurs vont utiliser la virtualisation pour tester le webbooting, nous utilisons du matriel rel ici pour correspondre au possible cas d'utilisation suivant qui seulement devrait tre considr comme un exemple.

Afin de dmarrer une image webboot il suffit d'avoir les lments men- 291
tionns ci-dessus, c'est--dire, vmlinuz et initrd.img sur une cl usb dans un
rpertoire nomm live/ et installer syslinux comme chargeur de dmarrage.
Ensuite, dmarrer partir de la cl usb et taper fetch=URL/CHEMIN/-
DU/FICHER aux options de dmarrage. live-boot va tlcharger le fichier
squashfs et le stocker dans la ram. De cette faon, il est possible d'utiliser
le systme de fichiers compress tlcharg comme un systme live normal. Par
exemple:

292

```
append boot=live components fetch=http://192.168.2.50/images/↵  
webboot/filesystem.squashfs
```

293

Cas d'utilisation: Vous avez un serveur web dans lequel vous avez
stock deux fichiers squashfs, un qui contient un bureau complet, comme
par exemple gnome, et un d'un systme standard. Si vous avez besoin
d'un environnement graphique pour une machine, vous pouvez brancher
votre cl usb et tlcharger l'image qui contient gnome. Si vous avez besoin
des outils inclus dans le deuxime type d'image, peut-tre pour une autre
machine, vous pouvez tlcharger celle du systme standard.

Aperu des outils

5. Aperu des outils

Ce chapitre fournit un aperu des trois principaux outils utilis dans la construction des systmes live: live-build, live-boot et live-config.

5.1 Le paquet live-build

live-build est une collection de scripts pour construire des systmes live. Ces scripts sont aussi appels commandes.

L'ide derriere live-build est de constituer un cadre qui utilise un rpertoire de configuration pour automatiser et personnaliser compltement tous les aspects de la construction d'une image Live.

Plusieurs concepts sont similaires ceux utilis pour construire des paquets Debian avec debhelper:

Les scripts ont un emplacement central pour la configuration de leur fonctionnement. Avec debhelper, c'est le sous-rpertoire debian/ d'un arbre de paquets. Par exemple, dh-install cherchera, entre autres, un fichier appel debian/install pour dterminer quels fichiers doivent exister dans un paquet binaire particulier. De la mme manire, live-build enregistre sa configuration entirement dans un sous-rpertoire config/.

Les scripts sont indpendants, c'est--dire qu'il est toujours sr d'excuter chaque commande.

Contrairement debhelper, live-build contient des outils pour gnrer une arborescence de configuration. Cela pourrait tre considr comme similaire des outils tels que dh-make. Pour plus d'informations sur ces outils, continuer la lecture, parce que le reste de cette section est sur les quatre

commandes les plus importantes. Notez que la commande lb est une fonction gnrique pour les commandes live-build.

lb config : Responsable de l'initialisation d'un rpertoire de configuration pour un systme Live. Voir **La commande lb config** pour plus d'informations.

lb build : Responsable du dmarrage d'un systme de construction Live. Voir **La commande lb build** pour plus d'informations.

lb clean : Responsable de la suppression des parties d'un systme de construction Live. Voir **La commande lb clean** pour plus d'informations.

5.1.1 La commande lb config

Comme indiqu dans **live-build**, les scripts qui composent live-build lisent leur configuration avec la commande source partir d'un seul rpertoire nomm config/. Comme la construction de ce rpertoire la main serait fastidieuse et source d'erreurs, la commande lb config peut tre utilise pour crer une arborescence de configuration.

Excuter la commande lb config sans aucun argument cre le sous-rpertoire config/ qui est peuple avec certains paramtres dans fichiers de configuration, et deux sous-rpertoires auto/ et local/ avec une arborescence de fichiers.

```
$ lb config
[2025-02-15 12:34:56] lb config
P: Using http proxy: http://127.0.0.1:3142
P: Creating config tree for a debian/testing/amd64 system
P: Symlinking hooks...
```

L'utilisation de lb config sans aucun argument serait appropriée pour les utilisateurs qui ont besoin d'une image de base, ou qui ont l'intention

de fournir plus tard une configuration plus complte via auto/config (voir **Gestion d'une configuration** pour plus de dtails).

Normalement, vous voulez indiquer certaines options. Par exemple, pour spcifier le gestionnaire de paquets utiliser lors de la construction de l'image:

```
$ lb config --apt aptitude
```

Il est possible d'indiquer plusieurs options, telles que:

```
$ lb config --binary-images netboot --bootappend-live "boot=live ↵
  components hostname=live -host username=live -user" ...
```

Une liste complte des options est disponible dans la page de manuel de lb'config.

5.1.2 La commande lb build

La commande lb build lit dans votre configuration partir du rpertoire config/. Elle excute alors les commandes de niveau infrieur ncessaires la construction de votre systme Live.

5.1.3 La commande lb clean

Le rle de la commande lb clean est d'enlever les diffrentes parties d'une construction afin que autres compilations ultrieures puissent commencer partir d'un tat propre. Par dfaut, les tapes chroot, binary et source sont nettoyes, mais le cache est laiss intact. En outre, les tapes peuvent tre nettoyes individuellement. Par exemple, si vous avez effectu des changements qui affectent uniquement la phase binaire, utilisez lb clean --binary

avant de construire un nouveau binaire. Si vos modifications invalident le bootstrap et/ou les caches de paquets, par exemple, modifications aux options --mode, --architecture ou --bootstrap, vous devez utiliser lb clean --purge. Voir la page de manuel de lb'clean pour une liste complte des options.

5.2 Le paquet live-boot

live-boot est une collection de scripts fournissant des hooks pour initramfs-tools. Il est utilis pour gnrer un initramfs capable de dmarrer des systmes live, comme ceux crs par live-build. Cela inclut ISOs, netboot tarballs, et les images pour cls USB.

At boot time it will look for read-only media containing a /live/ directory where a root filesystem (often a compressed filesystem image like squashfs) is stored. If found, it will create a writable environment, using OverlayFS, for Debian like systems to boot from.

More information on initial ramfs in Debian can be found in the Debian Linux Kernel Handbook at <https://kernel-team.pages.debian.net/kernel-handbook/> in the chapter on initramfs.

5.3 Le paquet live-config

live-config se compose des scripts qui s'excutent au dmarrage aprs live-boot pour configurer le systme live automatiquement. Il gre les tches telles que l'tablissement du nom d'hte, des paramtres rgionaux et du fuseau horaire, la cration de l'utilisateur live, l'inhibition des cron jobs et l'autologin de l'utilisateur live.

Gestion d'une configuration

6. Gestion d'une configuration

Ce chapitre explique comment gérer une configuration d'un système live à partir d'une création initiale, à travers des révisions successives et des versions successives du logiciel live-build et de l'image live elle-même.

6.1 Gérer les modifications de la configuration

Les configurations live sont rarement parfaites du premier coup. Il peut être bon de passer des options `lb config` à partir de la ligne de commande pour effectuer une construction unique, mais il est plus courant de réviser ces options et de construire un nouveau jusqu'à ce que vous soyez satisfait. Afin de prendre en charge ces changements, vous aurez besoin de scripts automatiques qui assurent le maintien de votre configuration dans un état cohérent.

6.1.1 Pourquoi utiliser des scripts auto? Que font-ils?

La commande `lb config` enregistre les options que vous lui passez avec les fichiers dans `config/*` avec beaucoup d'autres options aux valeurs par défaut. Si vous exécutez `lb config nouveau`, il ne réinitialisera pas l'option qui a été mise par défaut en fonction de vos options initiales. Ainsi, par exemple, si vous exécutez `lb config nouveau` avec une nouvelle valeur pour `--binary-images`, toutes les options qui ont été mises à leur valeur par défaut pour l'ancien type d'image ne peuvent plus fonctionner avec la nouvelle option. Ces fichiers ne sont pas destinés à être lus ou modifiés. Ils enregistrent des valeurs pour plus d'une centaine d'options, donc personne (y-compris vous) ne pourra voir dans ces options lesquelles vous avez

rellement indiquées. Finalement, si vous lancez `lb config`, puis mettez `live-build` à niveau et que celui-ci renomme une option, `config/*` contiendra toujours des variables nommées en fonction de l'ancienne option et qui ne seront plus valides.

Pour toutes ces raisons, les scripts `auto/*` vous rendront la vie plus facile. Ils sont de simples emballages pour les commandes `lb config`, `lb build` et `lb clean` qui sont conçus pour vous aider à gérer votre configuration. Le script `auto/config` enregistre votre commande `lb config` avec toutes les options désirées, le script `auto/clean` supprime les fichiers contenant les valeurs des variables de configuration et le script `auto/build` crée un `build.log` de chaque construction. Et chaque fois que vous lancez la commande `lb` correspondante, ces fichiers sont exécutés automatiquement. En utilisant ces scripts, votre configuration est plus facile à lire et a une cohérence interne d'une révision à l'autre. En outre, il sera plus facile pour vous d'identifier et corriger les options qui doivent changer lorsque vous mettez à niveau `live-build` après avoir lu la documentation mise à jour.

6.1.2 Utiliser les scripts auto d'exemple

Pour votre commodité, `live-build` est fourni avec des scripts shell d'exemple, pour les copier et les modifier. Lancez une nouvelle configuration par défaut, puis copiez les exemples:

```
$ mkdir mylive && cd mylive && lb config
$ mkdir auto
$ cp /usr/share/doc/live-build/examples/auto/* auto/
```

Modifiez `auto/config` en ajoutant des options comme bon vous semble. Par exemple:

```
#!/bin/sh
lb config noauto "
  --distribution stable "
  --binary-images hdd "
  --mirror-bootstrap http://ftp.ch.debian.org/debian/ "
  --mirror-binary http://ftp.ch.debian.org/debian/ "
"$@"
```

images comme suit:

```
$ mkdir live-images && cd live-images
$ lb config --config https://salsa.debian.org/live-team/live-↔
  images.git::debian
$ cd images/standard
```

345

340 Maintenant, chaque fois que vous utilisez lb config, auto/config rinitialisera la configuration base sur ces options. Lorsque vous souhaitez effectuer des modifications, modifiez les options dans ce fichier au lieu de les passer lb config. Lorsque vous utilisez lb clean, auto/clean va nettoyer les fichiers ainsi que tous les autres produits de construction. Et enfin, lorsque vous utilisez lb build, un journal de la construction est crit par auto/build dans build.log.

Modifiez auto/config et toutes les autres choses dont vous avez besoin dans l'arbre config en fonction de vos besoins. Par exemple, les images precompiles non officielles qui contiennent des paquets non-free sont faites en ajoutant simplement `--archive-areas main contrib non-free`.

Vous pouvez ventuellement dfnir un raccourci dans votre configuration Git en ajoutant la ligne suivante votre `$HOME"/.gitconfig`:

348

```
[url "https://salsa.debian.org/live-team/"]
  insteadOf = lso:
```

This enables you to use lso: anywhere you need to specify the address of a salsa.debian.org git repository. If you also drop the optional .git suffix, starting a new image using this configuration is as easy as:

350

```
$ lb config --config lso:live-images::debian
```

Le clonage de la totalit du dpt live-images copie les configurations utilises pour plusieurs images. Si vous voulez construire une image diffrente lorsque vous avez termin avec la premire, changez de rpertoire et, ventuellement, faites les modifications dont vous avez besoin.

Dans tous les cas, n'oubliez pas qu'il faut chaque fois construire l'image en tant que superutilisateur: lb build

352

341 Remarque: Un paramtre spcial noauto est utilis ici pour limiter un autre appel auto/config, vitant ainsi une rcursion infinie. Assurez-vous que vous ne l'avez pas accidentellement supprim en modifiant le fichier. Aussi, prenez soin de vous assurer quand vous divisez la commande lb config sur plusieurs lignes pour une meilleure lisibilit, comme le montre l'exemple ci-dessus, que vous n'oubliez pas la barre oblique inverse (de sorte que chaque ligne continue la suivante.

342 6.2 Cloner une configuration publie via Git

343 Use the lb config `--config` option to clone a Git repository that contains a live system configuration. If you would like to base your configuration on one maintained by the Debian Live Project, look at <https://salsa.debian.org/live-team/> for the repository named live-images in the category Subgroups and projects. This repository contains the configurations for the live systems **prebuilt images**.

344 Par exemple, pour construire une image standard, utilisez le dpt live-

Personnalisation des contenus

7. Vue d'ensemble de la personnalisation

Ce chapitre donne un aperçu des diverses façons dont vous pouvez personnaliser un système live.

7.1 Configuration pendant la construction vs. l'amorçage

Les options de configuration d'un système live sont divisées en options au moment de la construction, ces options sont appliquées pendant la création et des options au moment du démarrage, qui sont appliquées pendant le démarrage. Les options au moment du démarrage sont divisées en celles qui surviennent au début, appliquées par le paquet live-boot, et celles qui arrivent plus tard, appliquées par live-config. Toute option d'amorçage peut être modifiée par l'utilisateur en l'indiquant l'invite de démarrage. L'image peut également être construite avec les paramètres de démarrage par défaut et alors les utilisateurs peuvent normalement démarrer directement le système live sans indiquer aucune option lorsque toutes les valeurs par défaut sont appropriées. En particulier, l'argument `lb --bootappend-live` se compose de toutes les options de ligne de commande du noyau par défaut pour le système live, comme la persistance, les claviers, ou le fuseau horaire. Voir [Personnalisation des paramètres régionaux et la langue](#), par exemple.

Les options de configuration pendant la construction sont décrites dans la page de manuel pour `lb config`. Les options de configuration pendant l'amorçage sont décrites dans les pages de manuel pour `live-boot` et `live-config`. Bien que les paquets `live-boot` et `live-config` soient installés dans le système live que vous construisez, il est recommandé que vous les installiez également sur votre système de construction pour vous y référer facilement lorsque vous travaillez sur votre configuration. Cela peut être fait sans

danger car aucun des scripts contenus ne sont exécutés moins que le système soit configuré comme un système live.

7.2 Tapes de la construction

Le processus de construction est divisé en tapes, avec des personnalisations différentes appliquées dans l'ordre dans chaque tape. La première tape exécutée est la tape `bootstrap`. C'est la phase initiale de peuplement du répertoire `chroot` avec des paquets pour faire un système Debian de base. Elle est suivie par la tape `chroot`, qui complète la construction du répertoire `chroot`, le peuplant de tous les paquets listés dans la configuration, ainsi que tout autre matériel. La plupart de la personnalisation du contenu se produit à ce stade. La dernière tape de la préparation de l'image live est la tape `binary`, qui construit une image amorçable, en utilisant le contenu du répertoire `chroot` pour construire le système de fichiers racine pour le système Live. Il comprend l'installateur et tout autre matériel supplémentaire sur le support cible en dehors du système de fichiers du système live. Quand l'image live est construite, s'il est activé, le tarball des sources est construit dans la tape source.

Chacune de ces tapes, les commandes sont appliquées dans un ordre particulier. Les commandes sont ordonnées de manière à assurer que les personnalisations puissent être superposées de manière raisonnable. Par exemple, dans la tape `chroot`, les préconfigurations sont appliquées avant que tous les paquets ne soient installés, les paquets sont installés avant que tous les fichiers locaux inclus ne soient copiés et les hooks sont exécutés plus tard, quand tous les matériaux sont en place.

7.3 Supplément lb config avec des fichiers

Bien que `lb config` crée une arborescence de configuration dans le répertoire `config/`, pour accomplir vos objectifs, vous pourriez avoir besoin

de fournir des fichiers supplémentaires dans les sous-répertoires de `config/`. Selon l'endroit où les fichiers sont stockés dans la configuration, ils peuvent être copiés dans le système de fichiers du système live ou dans le système de fichiers de l'image binaire, ou peuvent fournir pendant la construction des configurations du système qui seraient lourdes à passer comme options de la ligne de commande. Vous pouvez inclure des choses telles que des listes personnalisées de paquets, art personnalisé, ou des scripts hook exécuter, soit pendant la construction soit au démarrage, ce qui augmente la flexibilité de `debian-live` avec le code de votre choix.

7.4 Tâches de personnalisation

Les chapitres suivants sont organisés par les types des tâches de personnalisation que les utilisateurs effectuent généralement: **Personnalisation de l'installation de paquets**, **Personnalisation des contenus** et **Personnalisation des paramètres régionaux et la langue** couvrent quelques choses que vous pourriez vouloir faire.

Personnalisation de l'installation de paquets

8. Personnalisation de l'installation de paquets

La personnalisation la plus fondamentale d'un système live est sans doute la sélection des paquets incluse dans l'image. Ce chapitre vous guide tout au long des différentes options de construction pour personnaliser l'installation des paquets avec live-build. Le plus large choix influençant les paquets disponibles pour l'installation dans l'image sont la distribution et les zones d'archive. Afin de vous assurer des vitesses de téléchargement décentes, vous devez choisir un miroir de distribution proche. Vous pouvez également ajouter vos propres dépôts pour les retroportages, paquets expérimentaux ou personnalisés, ou inclure des paquets directement comme fichiers. Vous pouvez définir des listes de paquets, incluant des metapaquets qui installent en même temps de nombreux paquets liés, tels que les paquets de bureau ou une langue particulière. Enfin, un certain nombre d'options donne un certain contrôle sur apt, ou si vous préférez, aptitude, pendant la construction quand les paquets sont installés. Vous pouvez trouver cela très pratique si vous utilisez un proxy, si vous voulez désactiver l'installation des paquets recommandés pour économiser l'espace, ou avez besoin de contrôler quelles versions des paquets sont installés via APT pinning, pour ne nommer que quelques possibilités.

8.1 Sources des paquets

8.1.1 Distribution, zones d'archive et mode

The distribution you choose has the broadest impact on which packages

are available to include in your live image. Specify the codename, which defaults to testing . Any current distribution carried in the archive may be specified by its codename here. (See [Terms](#) for more details.) The `--distribution` option not only influences the source of packages within the archive, but also instructs live-build to enable other sources.

For example, to build against the stable release, with security, updates (enabled per default) and additionally proposed-updates and backports, specify:

```
$ lb config --distribution stable --proposed-updates true --backports true
```

Similarly, for the unstable release, sid , which has neither security nor updates, specify:

```
$ lb config --distribution sid
```

Dans l'archive de distribution, les zones d'archive (archive areas) sont les principales divisions de l'archive. Dans Debian, ce sont main, contrib et non-free. Seule main contient des logiciels qui font partie de la distribution Debian, c'est donc la valeur par défaut. Une ou plusieurs valeurs peuvent être indiquées, par exemple:

```
$ lb config --archive-areas "main contrib non-free"
```

La prise en charge d'experimental est disponible pour certains dérivés de Debian grâce l'option `--mode`. L'option par défaut est debian mais seulement si vous construisez sur un système Debian ou un système inconnu. Si `lb config` est appelé sur un des dérivés pris en charge, il créera par défaut une image de ce dérivé. Si par exemple `lb config` est lancé en mode ubuntu,

les noms de distribution et des zones d'archives pour ce driv spécifique seront grs la place de ceux de Debian. Le mode modifie galement le comportement de live-build en fonction des drivs.

```
$ lb config --mirror-bootstrap http://localhost/debian/ "
--mirror-chroot-security http://localhost/debian-↵
security/
```

Remarque: Les projets pour lesquels ces modes ont t ajouts sont chargs d'aider les utilisateurs de ces options. Le Debian Live Project, en retour, fournit un soutien de dveloppement sur une base du meilleur effort seulement, en fonction des commentaires sur les projets drivs que nous n'avons pas dvelopps ou pris en charge nous-mmes.

Le miroir chroot, indiqu avec `--mirror-chroot`, est par dfaut la valeur de `--mirror-bootstrap`.

8.1.2 Miroirs de distribution

L'archive Debian est rplique sur un grand rseau de miroirs autour du monde pour que les habitants de chaque rgion puissent choisir un miroir proche ayant la meilleure vitesse de tlchargement. Chacune des options `--mirror-*` rgit quel miroir de distribution est utilis dans les diffrentes tapes de la construction. Rappelez-vous dans les **tapes de la construction** que l'tape `bootstrap` a lieu quand le chroot est initialement peupl par `debootstrap` avec un systme minimal, et l'tape `chroot` a lieu quand le chroot utilis pour construire le systme de fichiers du systme live est construit. Ainsi, les commutateurs des miroirs correspondants sont utilis pour ces tapes et plus tard, dans l'tape `binary`, les valeurs `--mirror-binary` et `--mirror-binary-security` sont utilises, remplaant tout miroir utilis dans une tape antrieure.

8.1.4 Miroirs de distribution utilis pendant l'exécution

The `--mirror-binary*` options govern the distribution mirrors placed in the binary image. These may be used to install additional packages while running the live system. The defaults employ `deb.debian.org`, a service that chooses a geographically close mirror based, among other things, on the user's IP family and the availability of the mirrors. This is a suitable choice when you cannot predict which mirror will be best for all of your users. Or you may specify your own values as shown in the example below. An image built from this configuration would only be suitable for users on a network where mirror is reachable.

```
$ lb config --mirror-binary http://mirror/debian/ "
--mirror-binary-security http://mirror/debian-security/ ↵
"
--mirror-binary-backports http://mirror/debian-backports↵
/
```

8.1.3 Miroirs de distribution utilis lors de la construction

Pour dfnir les miroirs de distribution utilis pendant la construction pour pointer vers un miroir local, il suffit de paramtrer `--mirror-bootstrap` et `--mirror-chroot-security` comme suit.

8.1.5 Dpts additionnels

Vous pouvez ajouter d'autres dpts, largissant votre choix de paquets au-del de ceux disponibles dans votre distribution cible. Cela peut tre, par exemple, pour avoir des paquets `rtports`, personnalis ou expérimentaux. Pour configurer des dpts supplmentaires, crez les fichiers

config/archives/your-repository.list.chroot, et/ou config/archives/your-repository.list.binary. Comme avec les options `--mirror-*`, ces fichiers donnent les dpts utilisés dans l'tape chroot lors de la construction de l'image, et dans l'tape binary, c'est--dire pendant l'exécution du système live.

Par exemple, config/archives/live.list.chroot vous permet d'installer les paquets du dpt des instantans debian live pendant la construction du système live.

```
deb http://debian-live.alioth.debian.org/ sid-snapshots main ↵
contrib non-free
```

Si vous ajoutez la même ligne config/archives/live.list.binary, le dpt sera ajouté au rpertoire /etc/apt/sources.list.d/ de votre système live.

Si ces fichiers existent, ils seront sélectionnés automatiquement.

You should also put the ASCII-armored GPG key used to sign the repository into config/archives/your-repository.key--binary,chroot" files.

Si vous avez besoin d'un APT pinning personnalisé, les préférences APT peuvent être placées dans les fichiers config/archives/your-repository.pref--binary,chroot" et elles seront automatiquement ajoutées à votre système live dans le rpertoire /etc/apt/preferences.d/.

Similarly, if you need custom APT AUTH.CONF(5) authentication configuration, this can be placed in config/archives/your-repository.auth--binary,chroot" files and will be automatically added to your live system's /etc/apt/auth.conf.d/ directory

8.2 Choisir les paquets à installer

Il y a un certain nombre de façons pour choisir quels paquets live-build

s'installeront dans votre image, couvrant toute une variété de besoins. Vous pouvez tout simplement nommer les paquets individuels à installer dans une liste de paquets. Vous pouvez également choisir des métapaquets dans ces listes, ou les sélectionner en utilisant les champs de contrôle de fichiers des paquets. Enfin, vous pouvez placer des paquets dans votre arbre config/ qui est bien adapté aux essais de nouveaux paquets ou expérimentaux avant qu'ils ne soient disponibles sur un dpt.

8.2.1 Listes de paquets

Les listes de paquets sont un excellent moyen d'exprimer quels paquets doivent être installés. La syntaxe de la liste comprend des sections conditionnelles, ce qui les rend faciles à construire et à adapter pour leur utilisation dans des configurations multiples. Les noms des paquets peuvent également être injectés dans la liste en utilisant des assistants de shell pendant la construction.

Remarque: Le comportement de live-build pour indiquer un paquet qui n'existe pas est déterminé par votre choix de l'utilitaire APT. Consultez **Choisir apt ou aptitude** pour plus de détails.

8.2.2 Utilisation des métapaquets

La façon la plus simple de remplir votre liste de paquets consiste à utiliser un métapaquet de tâche maintenu par votre distribution. Par exemple:

```
$ lb config
$ echo task-gnome-desktop > config/package-lists/desktop.list.↵
chroot
```

This supersedes the older predefined list method supported in live-build 2.x. Unlike predefined lists, task metapackages are not specific to the

Live System project. Instead, they are maintained by specialist working groups within the distribution and therefore reflect the consensus of each group about which packages best serve the needs of the intended users. They also cover a much broader range of use cases than the predefined lists they replace.

Tous les mtaquets de tches sont prfixs avec task-, donc un moyen rapide pour dterminer lesquels sont disponibles (mme s'il peut y avoir une poigne de faux positifs dont le nom correspond mais qui ne sont pas des mtaquets) est de faire correspondre le nom du paquet avec:

```
$ apt-cache search --names-only ^task -
```

En plus, vous trouverez d'autres mtaquets des fins diverses. Certains sont des sous-ensembles de paquets de tches plus larges, comme gnome-core, tandis que d'autres sont des pices individuelles spcialises d'un Debian Pure Blend, comme les mtaquets education-*. Pour lister tous les mtaquets dans l'archive, installez le paquet debtags et listez tous les paquets ayant l'tiquette role::metapackage comme suit:

```
$ debtags search role::metapackage
```

8.2.3 Listes de paquets locaux

Que vous listiez des mtaquets, des paquets individuels ou une combinaison des deux, toutes les listes de paquets locaux sont stockes dans config/package-lists/. Comme plus d'une liste peut tre utilise, cela se prte bien une conception modulaire. Par exemple, vous pouvez dcider de consacrer une liste un choix particulier de bureau, l'autre une collection de paquets connexes qui pourraient aussi bien tre utilis au-dessus d'un

bureau diffrent. Cela vous permet d'exprimer avec diffrentes combinaisons d'ensembles de paquets avec un minimum de tracas en utilisant des listes communes entre les diffrents projets d'images live.

Les listes de paquets qui existent dans ce rpertoire ont besoin d'avoir un suffixe .list pour tre traites, puis un suffixe d'tape supplmentaire .chroot ou .binary pour indiquer quelle tape la liste est destine.

The packages in the .list.chroot.install list are present both in the live system and in the installed system.

Remarque: Si vous n'indiquez pas le suffixe de l'tape, la liste sera utilise pour les deux tapes. Normalement, vous voulez indiquer .list.chroot de sorte que les paquets soient seulement installs dans le systme de fichiers live et ne pas avoir une copie supplmentaire des .deb place sur le support.

8.2.4 Listes de paquets locaux pour l'tape binary

Pour faire une liste pour l'tape binary, placez un fichier avec le suffixe .list.binary dans config/package-lists/. Ces paquets ne sont pas installs dans le systme de fichiers live, mais sont inclus sur le support live sous pool/. Vous utiliserez gnralement cette liste avec une des variantes d'installation non-live. Comme mentionn ci-dessus, si vous voulez que cette liste soit la mme que votre liste pour l'tape chroot, utilisez simplement le suffixe .list.

8.2.5 Listes de paquets gnres

Il arrive parfois que la meilleure faon de composer une liste soit de la gnrer avec un script. Toute ligne commençant par un point d'exclamation indique une commande excuter dans le chroot lorsque l'image est construite. Par exemple, on pourrait inclure la ligne ! grep-aptavail -n -sPackage

-FPriority standard —sort dans une liste de paquets qui permet de produire une liste trie des paquets disponibles avec Priority: standard.

En fait, la slection des paquets avec la commande `grep-aptavail` (du paquet `dctrl-tools`) est si utile que `live-build` fournit un script `Packages` titre de `commodit`. Ce script accepte deux arguments: `field` et `pattern`. Ainsi, vous pouvez crer une liste avec le contenu suivant:

```
$ lb config
$ echo '! Packages Priority standard' > config/package-lists/←
  standard.list.chroot
```

```
#if ARCHITECTURES i386 amd64
memtest86+
#endif
```

Vous pouvez galemment tester avec des variables pouvant contenir plus d'une valeur, par exemple pour installer `vrms` si `contrib` ou `non-free` est indiqu via `-archive-areas`:

```
#if ARCHIVEAREAS contrib non-free
vrms
#endif
```

8.2.6 Utiliser des conditions dans les listes de paquets

Toutes les variables de configuration de `live-build` stockes dans `config/*` (sans le prfixe `LB'`) peuvent tre utilises dans des instructions conditionnelles dans les listes de paquets. Gnralement, cela correspond toute option `lb config` en majuscule et avec tirets changs en caractres de soulignement. Mais en pratique, ce ne sont que celles qui influencent la slection des paquets qui font sens, comme `DISTRIBUTION`, `ARCHITECTURES` ou `ARCHIVEAREAS`.

Par exemple, pour installer `ia32-libs` si `-architectures amd64` est indiqu:

```
#if ARCHITECTURES amd64
ia32-libs
#endif
```

L'imbrication des conditions n'est pas prise en charge.

8.2.7 Suppression de paquets lors de l'installation

Il est possible de lister des paquets dans des fichiers utilisant les extensions `.list.chroot`live` et `.list.chroot`install` l'intrieur du rpertoire `config/package-lists`. Si une liste `install` et une liste `live` existent conjointement, les paquets dans la liste `.list.chroot`live` seront supprims par un hook aprs l'installation (si l'utilisateur utilise l'installateur). Les paquets dans la liste `.list.chroot`install` sont prsents la fois dans le systme `live` et dans le systme `install`. Il s'agit d'un paramtrage spcial pour l'installateur et peut se rvler utile si vous avez choisi `-debian-install` `live` dans votre configuration, et souhaitez supprimer des paquets spcifiques aux systmes `live` lors de l'installation.

8.2.8 Summary

The table below shows which configuration files are required to achieve the desired availability of the package.

	X.chroot	X.chroot'- live	X	X.binary
Package is installed in the live system	Yes	Yes	Yes	No
Package is removed after in- stalling the live system	No	Yes	No	N/A
Package can be installed from the live system with- out network	N/A	N/A	Yes *1	Yes

l'installateur de Debian. Cela ne veut pas dire qu'une image live ne pourrait pas tre construite pour prendre en charge plusieurs environnements de bureau ou plusieurs langues et offrir l'utilisateur un choix, mais ce n'est pas le comportement par dfaut de live-build.

Comme aucune disposition n'est faite automatiquement pour les tches de la langue, qui comprennent des lments tels que des polices spcifiques la langue et des paquets de mthodes de saisie, vous devez les indiquer dans votre configuration si vous les voulez. Par exemple, une image de bureau GNOME contenant la prise en charge de l'allemand pourrait inclure les mtpaquets de tches suivants:

```
$ lb config
$ echo "task-gnome-desktop task-laptop" && config/package-lists/my<->
  .list.chroot
$ echo "task-german task-german-desktop task-german-gnome-desktop"<->
  && config/package-lists/my.list.chroot
```

*1: Because the installer needs this package

X = config/package-lists/custom`name`.list

8.2.9 Tches de bureau et de langue

Les tches de bureau et de langue sont des cas particuliers qui ont besoin d'une certaine planification et de configuration supplmentaire. Dans l'installateur Debian, si le support a t prpar pour un environnement de bureau particulier, la tche correspondante sera automatiquement installe. Ainsi, il y a tches internes gnome-desktop, kde-desktop, lxde-desktop et xfce-desktop, dont aucune n'est propose dans le menu tasksel. De mme, il n'y a pas d'lment de menu pour les tches de langue, mais le choix de la langue de l'utilisateur lors de l'installation influence le choix des tches de langue correspondantes.

Lors du dveloppement d'une image de bureau live, l'image s'amorce gnralement directement sur un bureau de travail. Les choix de l'environnement de bureau et la langue par dfaut ont t faits pendant la construction et non pas pendant l'excution comme dans le cas de

8.2.10 Version et type de noyau

Un ou plusieurs types de noyau seront inclus dans votre image par dfaut, en fonction de l'architecture. Vous pouvez choisir diffrents types avec l'option `--linux-flavours`. Chaque type est suffix partir de `linux-image` pour former le nom de chaque mtpaquet qui dpend son tour d'un paquet noyau exact inclure dans votre image.

Ainsi, par dfaut, une image pour l'architecture amd64 comprendra le mtpaquet `linux-image-amd64`, et une image pour l'architecture i386 comprendra le mtpaquet `linux-image-586`.

Lorsque plus d'une version du paquet du noyau est disponible dans vos archives configurees, vous pouvez indiquer un nom du paquet du noyau diffrent avec l'option `--linux-packages`. Par exemple, supposons

que vous construisiez une image pour l'architecture amd64 et ajoutiez l'archive expérimentale pour faire des essais. Pour installer le noyau linux-image-3.18.0-trunk-amd64 vous pouvez configurer cette image comme suit:

```
$ lb config --linux-packages linux-image-3.18.0-trunk
$ echo "deb http://deb.debian.org/debian/ experimental main" & \
  config/archives/experimental.list.chroot
```

8.2.11 Noyaux personnalisés

Vous pouvez créer et inclure vos propres noyaux personnalisés, condition qu'ils soient intégrés dans le système de gestion des paquets Debian. Le système de live-build ne gère pas les noyaux qui ne sont pas construits sous forme de paquets .deb.

La façon correcte et recommandée de déployer vos propres paquets du noyau est de suivre les instructions dans le kernel-handbook. N'oubliez pas de modifier l'ABI et les suffixes de manière appropriée, puis d'inclure une construction complète des paquets linux et linux-latest dans votre dpt.

Si vous optez pour la construction des paquets du noyau sans les meta-paquets correspondants, vous devez indiquer une chaîne `--linux-packages` appropriée tel que discuté dans [Version et type de noyau](#). Comme nous l'expliquons dans [Installation de paquets modifiés ou tiers](#), il est préférable que vous incluiez vos paquets de noyau personnalisés dans votre propre dpt, bien que les alternatives discutées dans cette section fonctionnent bien également.

Donner des conseils sur la façon de personnaliser votre noyau sort du cadre de ce document. Toutefois, vous devez au moins vous assurer que votre configuration répond à ces exigences minimales:

Utilisez un ramdisk initial.

Incluez le système de fichiers union (i.e. généralement OverlayFS).

Incluez tous les autres modules du système de fichiers requis pour votre configuration (habituellement squashfs).

8.3 Installation de paquets modifiés ou tiers

Bien que ce soit contre la philosophie d'un système live, il peut parfois être nécessaire de construire un système live avec des versions modifiées des paquets du dpt Debian. Cela peut être pour modifier ou prendre en charge des fonctionnalités supplémentaires, des langues et la marque, ou même pour supprimer des éléments indésirables dans les paquets existants. De même, les paquets tiers peuvent être utilisés pour ajouter des fonctionnalités sur mesure et/ou propriétaires.

This section does not cover advice regarding building or maintaining modified packages. Joachim Breitner's 'How to fork privately' method from <http://www.joachim-breitner.de/blog/archives/282-How-to-fork-privately.html> may be of interest, however. The creation of bespoke packages is covered in the Debian New Maintainers' Guide at <https://www.debian.org/doc/manuals/maint-guide/> and elsewhere.

Il y a deux façons d'installer des paquets personnalisés modifiés:

packages.chroot

Utiliser un dpt APT personnalisé

Utiliser packages.chroot est plus simple à réaliser et utile pour les personnalisations ponctuelles mais a un certain nombre d'inconvénients, alors qu'utiliser un dpt personnalisé APT est plus fastidieux à mettre en place.

8.3.1 Utiliser packages.chroot pour installer des paquets personnalisés

Pour installer un paquet personnalisé, il suffit de le copier dans le répertoire `config/packages.chroot/`. Les paquets qui sont dans ce répertoire seront automatiquement installés dans le système live pendant la construction du système - vous n'avez pas besoin de les indiquer ailleurs.

Les paquets doivent être nommés de la manière prescrite. Une façon simple de le faire consiste à utiliser `dpkg-name`.

L'utilisation de `packages.chroot` pour l'installation de paquets personnalisés a des inconvénients:

- Il n'est pas possible d'utiliser APT de façon sécurisée.

- Vous devez installer tous les paquets appropriés dans le répertoire `config/packages.chroot/`.

- Il ne se prête pas au stockage de configurations des systèmes live dans le contrôle de révision.

8.3.2 Utiliser un dépôt APT pour installer des paquets personnalisés.

Contrairement à l'utilisation de `packages.chroot`, lorsque vous utilisez un dépôt personnalisé APT, vous devez vous assurer que vous indiquez les paquets ailleurs. Voir [Choisir les paquets à installer](#) pour plus de détails.

S'il peut sembler inutile de créer un dépôt APT pour installer des paquets personnalisés, l'infrastructure peut être facilement réutilisée une date ultérieure pour offrir les mises à jour des paquets modifiés.

The APT repository does not necessarily need to be online, you can use a local repository instead. However, in both cases the repository needs to be signed.

Example:

```
$ gpg --armor --output config/archives/custom_repo.gpg.key$-EXTENSION" --export-options export-minimal --export $-SIGNING-KEY"
$ cat ;; EOF & config/archives/custom_repo.list$-EXTENSION"
deb [signed-by=/etc/apt/trusted.gpg.d/custom_repo.gpg.key$-EXTENSION".asc] $-URI" $-SUITE" $-COMPONENTS"
EOF
$ echo "$-PACKAGESFROMREPOSITORY"" & config/package-lists/
custom_repo.list$-EXTENSION"
```

Where:

- `$-EXTENSION`: the optional stage suffix, see the [summary](#)

- `$-SIGNING-KEY`: the keyID of the signature of the repository

- `$-URI`: the URI to the repository, e.g. `http://deb.debian.org/debian/` or `file://$(pwd)/my-local-repository`

- `$-SUITE`: the suite within the repository, e.g. `my-debian-based-distro`

- `$-COMPONENTS`: the components within the repository, e.g. `main`

- `$-PACKAGESFROMREPOSITORY`: the names of the packages to install (dependencies will automatically be installed as well)

8.3.3 Les paquets personnalisés et APT

live-build utilise apt pour installer tous les paquets dans le système live, il héritera donc des comportements de ce logiciel. Un exemple pertinent est que (en supposant une configuration par défaut), s'il y a un paquet disponible dans deux dépôts différents avec des numéros de version différents, APT choisira d'installer le paquet ayant le numéro de version le plus grand.

Pour cette raison, vous pouvez incrémenter le numéro de version dans les fichiers `debian/changelog` de vos paquets personnalisés pour vous assurer que votre version modifiée sera installée au lieu d'une autre provenant

des dpts officiels Debian. Cela peut aussi tre fait en modifiant les prfrnces d'APT pinning dans le systme live voir **APT pinning** pour plus d'informations.

```
$ lb config --apt-http-proxy http://proxy/
```

8.4 Configuration d'APT pendant la construction

Vous pouvez configurer APT par un certain nombre d'options appliques uniquement pendant la construction. (La configuration d'APT utilise dans le systme live en fonctionnement peut tre configure de faon normale pour un systme live, c'est--dire en incluant les configurations appropries dans config/includes.chroot/.) Pour une liste complte, regardez les options commenant par apt dans la page de manuel de lb.config.

8.4.1 Choisir apt ou aptitude

Vous pouvez choisir d'utiliser soit apt, soit aptitude. Le choix du logiciel utilis dpend de l'argument `--apt` de lb.config. Choisissez la mthode ayant le comportement que vous prfrez pour l'installation de paquets, la diffrence notable tant la manire dont les paquets manquants sont traits.

apt: Avec cette mthode, si un paquet manquant est indiqu, l'installation va chouer. C'est le rglage par dfaut.

aptitude: Avec cette mthode, si un paquet manquant est indiqu, l'installation va russir.

8.4.2 Utilisation d'un proxy avec APT

One commonly required APT configuration is to deal with building an image behind a proxy. You may specify your APT proxy with the `--apt-http-proxy` option as needed, e.g.

8.4.3 Rgler APT pour conomiser de l'espace

Vous pouvez avoir besoin d'conomiser de l'espace sur le support d'image, auquel cas l'une ou l'autre ou les deux options suivantes peuvent tre d'intrt.

Si vous ne voulez pas inclure les index d'APT dans l'image, vous pouvez les omettre avec:

```
$ lb config --apt-indices false
```

Cela n'influencera pas les entres dans `/etc/apt/sources.list`, mais seulement le fait que `/var/lib/apt` contienne les fichiers index ou non. La contrepartie est qu'APT a besoin de ces index afin d'oprer dans le systme live. Par consquent, avant de procder `apt-cache search` ou `apt-get install` par exemple, l'utilisateur doit faire `apt-get update` pour crer ces index.

Si vous trouvez que l'installation des paquets recommends fait trop gonfler votre image, condition d'tre prt faire face aux consequences dcrites ci-dessous, vous pouvez dsactiver l'option par dfaut d'APT avec:

```
$ lb config --apt-recommends false
```

The most important consequence of turning off recommends is that live-boot and live-config themselves recommend some packages that provide important functionality used by most Live configurations.

Two packages which you most probably will want to add again are:

user-setup which live-config recommends is used to create the live user.

sudo which live-config recommends is used to obtain root access in the live-image, which is needed to shutdown the computer.

```
$ lb config --apt-recommends false
$ echo "user-setup sudo" & config/package-lists/recommends.list.<↵
chroot
```

```
$ lb config --apt-options "--yes -oAcquire::Check-Valid-Until=<↵
false"
```

In all but the most exceptional circumstances you need to add back at least some of these recommends to your package lists or else your image will not work as expected, if at all. Look at the recommended packages for each of the live-* packages included in your build and if you are not certain you can omit them, add them back into your package lists.

The more general consequence is that if you don't install recommended packages for any given package, that is, packages that would be found together with this one in all but unusual installations ([APT pinning](#).

8.4.4 Passer des options apt ou aptitude

S'il n'y a pas d'option lb config pour modifier le comportement d'APT de la façon dont vous avez besoin, utilisez `--apt-options` ou `--aptitude-options` pour passer des options par le biais de l'outil APT configur. Consultez les pages de manuel apt et aptitude pour les détails. Notez que les deux options ont des valeurs par défaut que vous aurez besoin de conserver en plus des remplacements que vous pouvez fournir. Ainsi, par exemple, supposons que vous ayez inclus quelque chose de `snapshot.debian.org` des fins de test et que vous vouliez indiquer `Acquire::Check-Valid-Until=false` pour satisfaire APT avec le fichier `Release` obsolète. Vous le feriez comme dans l'exemple suivant, avec l'ajout de la nouvelle option après la valeur par défaut `--yes`:

Veillez lire attentivement les pages de manuel pour bien comprendre ces options et quand les utiliser. Ce n'est qu'un exemple et ne doit pas être interprété comme un conseil pour configurer votre image de cette façon. Par exemple, cette option ne serait pas appropriée pour une version finale d'une image live.

Pour les configurations plus compliquées concernant des options `apt.conf`, vous pourriez vouloir créer un fichier `config/apt/apt.conf`. Consultez aussi les autres options `apt-*` pour obtenir quelques raccourcis pratiques pour les options fréquemment utilisées.

8.4.5 APT pinning

Pour plus de contexte, veuillez d'abord lire la page de manuel `apt-preferences(5)`. APT pinning peut être configuré soit pendant la construction, soit pendant l'exécution. Dans le premier cas, créez `config/archives/*.pref`, `config/archives/*.pref.chroot`, et `config/apt/preferences`. Dans le second cas, créez `config/includes.chroot/etc/apt/preferences`.

Imaginons que vous voulez construire un système live trixie mais qu'il faut installer tous les paquets live qui finissent dans l'image binaire de `sid` pendant la construction. Vous devez ajouter `sid` votre fichier APT sources et fixer tous les paquets live avec une priorité supérieure mais tous les autres paquets avec une priorité inférieure la priorité par défaut de sorte que seuls les paquets que vous voulez soient installés à partir de `sid` pendant la construction et que tous les autres viennent de la distribution du système cible, trixie. Ce qui suit devrait accomplir cela:

```
$ echo "deb http://mirror/debian/ sid main" > config/archives/sid.list.chroot
$ cat << config/archives/sid.pref.chroot >> EOF
Package: live-*
Pin: release n=sid
Pin-Priority: 600

Package: *
Pin: release n=sid
Pin-Priority: 1
EOF
```

518 Une priorité pin négative empêchera l'installation d'un paquet, comme dans le cas où vous ne voudriez pas un paquet qui est recommandé par un autre paquet. Supposons que vous construisiez une image LXDE en utilisant `task-lxde-desktop` dans `config/package-lists/desktop.list.chroot` mais que vous ne vouliez pas que l'utilisateur soit invité à stocker les mots de passe wifi dans le trousseau de clés. Cette liste comprend `lxde-core`, qui recommande `gksu`, qui à son tour recommande `gnome-keyring`. Donc, vous voulez omettre le paquet recommandé `gnome-keyring`. Cela peut être fait en ajoutant ce qui suit à `config/apt/preferences`:

```
Package: gnome-keyring
Pin: version *
Pin-Priority: -1
```

Personnalisation des contenus

9. Personnalisation des contenus

Ce chapitre aborde la personnalisation fine des contenus du système live au-delà du simple choix des paquets inclure. Les inclusions vous permettent d'ajouter ou de remplacer des fichiers arbitraires dans votre image du système live, les hooks vous permettent d'exécuter des commandes arbitraires dans différentes étapes de la construction et au démarrage et la préconfiguration (preseeding) vous permet de configurer les paquets quand ils sont installés en fournissant des réponses aux questions debconf.

9.1 Includes

Bien qu'idéalement un système live comprendrait des fichiers entièrement fournis par des paquets non modifiés, il peut être pratique de fournir ou de modifier certains contenus par le biais de fichiers. Avec les includes, il est possible d'ajouter (ou remplacer) des fichiers arbitraires dans votre image du système live. live-build prévoit deux mécanismes pour leur utilisation:

Chroot local includes: Ils vous permettent d'ajouter ou remplacer des fichiers sur le système de fichiers chroot/Live. Veuillez consulter [Live/chroot local includes](#) pour plus d'informations.

Binary local includes: Ils vous permettent d'ajouter ou de remplacer des fichiers dans l'image binaire. Veuillez consulter [Binary local includes](#) pour plus d'informations.

Veuillez consulter les [Termes](#) pour plus d'informations sur la distinction entre les images Live et binary.

9.1.1 Live/chroot local includes

Les chroot local includes peuvent être utilisés pour ajouter ou remplacer des fichiers dans le système de fichiers chroot/Live afin qu'ils puissent être utilisés dans le système Live. Une utilisation typique est de peupler l'arborescence du répertoire de l'utilisateur (/etc/skel) utilisée par le système live pour créer le répertoire home de l'utilisateur Live. Une autre est de fournir des fichiers de configuration qui peuvent être simplement ajoutés ou remplacer l'image sans traitement, voir [Chroot local hooks](#) si le traitement est nécessaire.

Pour inclure des fichiers, il suffit de les ajouter à votre répertoire config/includes.chroot. Ce répertoire correspond au répertoire racine / du système live. Par exemple, pour ajouter un fichier /var/www/index.html dans le système live, utilisez:

```
$ mkdir -p config/includes.chroot/var/www
$ cp /path/to/my/index.html config/includes.chroot/var/www
```

Votre configuration aura alors le schéma suivant:

```
-- config
[... ]
-- includes.chroot
--   -- var
--     -- www
--       -- index.html
[... ]
```

Les chroot local includes sont installés après l'installation de paquets de sorte que les fichiers installés par les paquets sont remplacés.

9.1.2 Binary local includes

Pour inclure des matriels tels que des documents ou des vidos sur le systme de fichiers des supports, afin qu'ils soient accessibles ds l'insertion du support sans dmarrer le systme live, vous pouvez utiliser binary local includes. Cela fonctionne de faon similaire aux chroot local includes. Par exemple, supposons que les fichiers ~/video'demo.* sont des vidos de dmonstration du systme live dcrits par et lis par une page d'index HTML. Copiez simplement le matriel dans config/includes.-binary/ comme suit:

```
$ cp ~/video'demo.* config/includes.binary/
```

Ces fichiers apparaissent maintenant dans le rpertoire racine du support live.

9.2 Hooks

Les hooks permettent l'excuton des commandes dans les tapes de la construction chroot et binary afin de personnaliser l'image. Selon que vous construisez une image live ou une image normal du systme, vous devez placer vos hooks dans config/hooks/live ou config/hooks/normal respectivement. Ceux-ci sont souvent appels hooks locales parce qu'ils sont excuts dans l'environnement de construction.

Il y a aussi des hooks pendant le dmarrage qui vous permettent d'excuter des commandes une fois que l'image a dj t construite, au cours du processus de dmarrage.

9.2.1 Chroot local hooks

Pour excuter des commandes l'tape chroot, crer un script hook avec le

suffixe .hook.chroot contenant les commandes dans le rpertoire config/hooks/live ou config/hooks/normal. Le hook s'excutera dans le chroot aprs que le reste de votre configuration chroot ait t appliqu, donc n'oubliez pas de vous assurer que votre configuration inclut tous les paquets et les fichiers dont votre hook a besoin pour fonctionner. Consultez les exemples de scripts chroot hook pour diverses tches courantes de personnalisation chroot fournis dans /usr/share/doc/live-build/examples/hooks que vous pouvez copier ou faire un lien symbolique pour les utiliser dans votre propre configuration.

9.2.2 Binary local hooks

Pour excuter des commandes l'tape binaire, crez un script hook avec le suffixe .hook.binary contenant les commandes dans le rpertoire config/hooks/live ou config/hooks/normal. Le hook sera excut aprs toutes les autres commandes binaires, mais avant binary'checksums, la dernire commande binaire. Les commandes de votre hook ne s'excutent pas dans le chroot, afin de prendre soin de ne pas modifier les fichiers en dehors de l'arbre de construction, ou vous pourriez endommager votre systme de construction! Consultez les exemples de scripts binary pour diverses tches courantes de personnalisation binaires fournis dans /usr/share/doc/live-build/examples/hooks que vous pouvez copier ou lier symboliquement pour les utiliser dans votre propre configuration.

9.2.3 Hooks pendant le dmarrage

Pour excuter des commandes pendant le dmarrage, vous pouvez fournir live-config hooks comme expliqu dans la section Personnalisation de sa page de manuel. Examinez les hooks de live-config fournis dans /lib/live/config/, en notant les numros de squence. Fournissez ensuite votre propre hook prcd d'un numro de squence approprie, soit comme un chroot

local include dans `config/includes.chroot/lib/live/config/`, soit comme un paquet personnalis tel que discut dans [Installation des paquets modifis ou de tiers](#).

548 9.3 Préconfigurer questions de debconf

549 Les fichiers dans le rpertoire `config/preseed/` avec le suffixe `.cfg` suivis de l'tape (`.chroot` or `.binary`) sont considrs comme des fichiers de prconfiguration debconf et sont installs par live-build en utilisant `debconf-set-selections`.

550 Pour plus d'informations sur debconf, veuillez consulter `debconf(7)` dans le paquet debconf.

Personnalisation des comportements pendant l'exécution

10. Personnalisation des comportements pendant l'exécution

Toute la configuration qui est faite pendant l'exécution est faite par live-config. Voici quelques options parmi les plus courantes de live-config qui peuvent intéresser les utilisateurs. Une liste complète de toutes les possibilités peut être trouvée dans la page de manuel de live-config.

10.1 Personnalisation de l'utilisateur live

Une considération importante est que l'utilisateur live est créé par live-boot au démarrage, non pas par live-build pendant la construction. Cela influence non seulement l'emplacement des documents relatifs à l'utilisateur live sont introduits dans la construction, tel que discuté dans [Live/chroot local includes](#), mais aussi tous les groupes et autorisations associés à l'utilisateur live.

You can specify additional groups that the live user will belong to by using any of the possibilities to configure live-config. For example, to add the live user to the fuse group, you can either add the following file in `config/includes.chroot/etc/live/config.conf.d/10-user-setup.conf`:

```
LIVE`USER`DEFAULT`GROUPS="audio cdrom dip floppy video plugdev ↵
netdev powerdev scanner bluetooth fuse"
```

ou utiliser `live-config.user-default-groups=audio,cdrom,dip,floppy,video,plugdev`

comme paramètre d'amorçage.

Il est également possible de changer le nom de l'utilisateur par défaut `user` et du mot de passe par défaut `live`. Si vous voulez faire cela, vous pouvez le faire facilement comme suit:

Pour modifier le nom de l'utilisateur par défaut, vous pouvez simplement l'indiquer dans votre configuration:

```
$ lb config --bootappend-live "boot=live components username=live ↵
user"
```

Une façon possible de changer le mot de passe par défaut est d'utiliser un hook comme décrit dans [Hooks pendant le démarrage](#). Pour ce faire vous pouvez utiliser le hook `passwd` de `/usr/share/doc/live-config/examples/hooks`, ajouter un préfixe correct (par exemple `2000-passwd`) et l'ajouter dans `config/includes.chroot/lib/live/config/`

10.2 Personnalisation des paramètres régionaux et de la langue

Au démarrage du système live, la langue est implicite dans deux tapes:

- la génération des paramètres régionaux
- le réglage de la disposition du clavier

Les paramètres régionaux par défaut pendant la construction d'un système Live sont `locales=en`US.UTF-8`. Pour définir les paramètres régionaux qui doivent être générés, utilisez le paramètre `locales` dans l'option `--bootappend-live` de `lb config`, par exemple

```
$ lb config --bootappend-live "boot=live components locales=de`CH. ↵
UTF-8"
```


Plusieurs paramtres rgionaux peuvent tre indiqus dans une liste spare par des virgules.

Ce paramtre, ainsi que les paramtres de configuration du clavier indiqus ci-dessous, peut galement tre utilis sur la ligne de commande du noyau. On peut indiquer des paramtres rgionaux avec `language=country` (dans ce cas, le codage par dfaut est utilis) ou l'expression complte `language=country.encoding`. Une liste des paramtres rgionaux et le codage pour chacun peuvent tre trouvs dans `/usr/share/i18n/SUPPORTED`.

La configuration du clavier pour la console et pour X est faite par `live-config` en utilisant le paquet `console-setup`. Pour les configurer, utilisez les paramtres de dmarrage `keyboard-layouts`, `keyboard-variants`, `keyboard-options` et `keyboard-model` avec l'option `--bootappend-live`. On peut trouver les options valides dans `/usr/share/X11/xkb/rules/base.lst`. Pour trouver les dispositions et les variantes correspondantes une langue, essayez de rechercher le nom anglais de la nation o la langue est parle, par exemple:

```
$ egrep -i '(!—german.*switzerland)' /usr/share/X11/xkb/rules/↵
base.lst
! model
! layout
!   ch                German (Switzerland)
! variant
!   legacy            ch: German (Switzerland, legacy)
!   de`nodeadkeys     ch: German (Switzerland, eliminate dead keys)
!   de`sundeadkeys    ch: German (Switzerland, Sun dead keys)
!   de`mac             ch: German (Switzerland, Macintosh)
! option
```

Chaque variante prsente une description de la disposition applique.

Souvent, seule la disposition doit tre configure. Par exemple, pour obtenir les fichiers des paramtres rgionaux de l'allemand et la disposition du clavier suisse allemand dans X, utilisez:

```
$ lb config --bootappend-live "boot=live components locales=de`CH.↵
UTF-8 keyboard-layouts=ch"
```

Toutefois, pour les cas d'utilisation trs spcifiques, on peut inclure d'autres paramtres. Par exemple, pour mettre en place un systme franais avec une disposition French-Dvorak (Bpo) avec un clavier USB TypeMatrix EZ-Reach 2030, utilisez:

```
$ lb config --bootappend-live "
boot=live components locales=fr`FR.UTF-8 keyboard-layouts=fr↵
keyboard-variants=bepo keyboard-model=tm2030usb"
```

Plusieurs valeurs peuvent tre indiqus dans des listes spares par des virgules pour chacune des options `keyboard-*`, l'exception de `keyboard-model` qui accepte une seule valeur. Veuillez consulter la page de manuel `keyboard(5)` pour plus de dtails et des exemples des variables `XKBMODEL`, `XKBLAYOUT`, `XKBVARIANT` et `XKBOPTIONS`. Si plusieurs valeurs `keyboard-variants` sont donnes, elles seront jumeles une avec les valeurs `keyboard-layouts` (voir `setxkbmap(1)` option `-variant`). On peut utiliser des valeurs vides; par exemple pour rgler deux dispositions, une par dfaut US QWERTY et l'autre US Dvorak, utilisez:

```
$ lb config --bootappend-live "
boot=live components keyboard-layouts=us,us keyboard-↵
variants=,dvorak"
```

10.3 Persistance

Le paradigme d'un Live CD est d'tre un systme pr-install qui amorce sur

un support en lecture seule, comme un cdrom, où les données et les modifications ne survivent pas aux redémarrages du matériel hôte qui l'exécute.

Un système live est une généralisation de ce paradigme et greffe ainsi d'autres supports en plus de CDs. Malgré tout, dans son comportement par défaut, il doit être considéré en lecture seule et toutes les évolutions pendant l'exécution du système sont perdues à l'arrêt.

La persistance est un nom commun pour les différents types de solutions pour sauvegarder, après un redémarrage, certaines ou toutes les données, de cette évolution pendant l'exécution du système. Pour comprendre comment cela fonctionne, il peut être utile de savoir que même si le système est démarré et exécuté à partir d'un support en lecture seule, les modifications des fichiers et répertoires sont écrites sur des supports inscriptibles, typiquement un disque ram (tmpfs) et les données des disques RAM ne survivent pas à un redémarrage.

Les données stockées sur ce disque virtuel doivent être enregistrées sur un support inscriptible persistant comme un support de stockage local, un partage réseau ou même une session d'un CD/DVD multisession (r)inscriptible. Tous ces supports sont pris en charge dans les systèmes live de différentes manières, et tous, à partir du dernier, nécessitent un paramètre d'amorçage spécial préciser au moment du démarrage: `persistence`.

Si le paramètre de démarrage `persistence` est réglé (et `nopersistence` n'est pas utilisé), les supports de stockage locaux (par exemple les disques durs, les clés USB) seront examinés pour trouver des volumes persistants pendant le démarrage. Il est possible de limiter les types de volumes persistants à utiliser en indiquant certains paramètres de démarrage décrits dans la page de manuel `live-boot(7)`. Un volume persistant est un des éléments suivants:

une partition, identifiée par son nom GPT.

un système de fichiers, identifié par son étiquette de système de fichiers.

un fichier image situé sur la racine d'un système de fichiers en lecture seule (même une partition NTFS d'un système d'exploitation étranger), identifié par son nom de fichier.

L'étiquette du volume pour les overlays doit être `persistence` mais elle sera ignorée moins de contenir dans sa racine un fichier nommé `persistence.conf` qui est utilisé pour personnaliser entièrement la persistance du volume, c'est-à-dire indiquer les répertoires que vous voulez sauvegarder dans votre volume de persistance après un redémarrage. Voir [Le fichier `persistence.conf`](#) pour plus de détails.

Voici quelques exemples montrant comment préparer un volume à utiliser pour la persistance. Cela peut être, par exemple, une partition ext4 sur un disque dur ou sur une clé USB créée avec:

```
# mkfs.ext4 -L persistence /dev/sdb1
```

Voir aussi [Utilisation de l'espace disponible sur une clé USB](#).

Si vous avez déjà une partition sur votre périphérique, vous pouvez simplement modifier l'étiquette avec l'un des exemples suivants:

```
# tune2fs -L persistence /dev/sdb1 # for ext2,3,4 filesystems
```

Voici un exemple montrant comment créer un fichier image avec un système de fichiers ext4 pour être utilisé pour la persistance:

```
$ dd if=/dev/null of=persistence bs=1 count=0 seek=1G # for a 1GB ←
    sized image file
$ /sbin/mkfs.ext4 -F persistence
```

Une fois que le fichier image est créé, titre d'exemple, pour rendre `/usr`

persistant mais seulement enregistrer les modifications que vous apportez ce rpertoire et non pas tout le contenu de /usr, vous pouvez utiliser l'option union. Si le fichier image se trouve dans votre rpertoire personnel, copiez-le la racine du systme de fichiers de votre disque dur et montez-le dans /mnt comme suit:

```
# cp persistence /
# mount -t ext4 /persistence /mnt
```

Ensuite, crez le fichier persistence.conf ajoutant du contenu et dmontez le fichier image.

```
# echo "/usr union" && /mnt/persistence.conf
# umount /mnt
```

Maintenant, redmarrez dans votre support live avec le paramtre de dmar-
rage persistence.

10.3.1 Le fichier persistence.conf

Un volume ayant l'tiquette persistence doit tre configur avec un fichier persistence.conf pour crer des rpertoires persistants arbitraires. Ce fichier, situ sur le systme de fichiers racine du volume, contrle quels rpertoires il rend persistants, et de quelle manire.

La faon de configurer les montages overlays est dcrite en dtail dans la page de manuel persistence.conf(5), mais un simple exemple devrait suffire pour la plupart des utilisations. Imaginons que nous voulions rendre notre rpertoire personnel et APT cache persistants dans un systme de fichiers ext4 sur la partition /dev/sdb1:

```
# mkfs.ext4 -L persistence /dev/sdb1
# mount -t ext4 /dev/sdb1 /mnt
# echo "/home" && /mnt/persistence.conf
# echo "/var/cache/apt" && /mnt/persistence.conf
# umount /mnt
```

Puis nous redmarrons. Lors du premier dmarrage, les contenus du /home et /var/cache/apt seront copis dans le volume persistant. partir de ce moment, tous les changements dans ces rpertoires seront stocks dans le volume persistant. Veuillez remarquer que les chemins rpertoris dans le fichier persistence.conf ne peuvent pas contenir d'espaces ou d'lments spciaux . et ... En outre, ni /lib, /lib/live (ou un de leurs sous-rpertoires), ni / ne peuvent tre rendus persistants en utilisant des montages personnalis. Comme solution cette limitation, vous pouvez ajouter / union votre fichier persistence.conf pour obtenir une persistance complte.

10.3.2 Utilisation de plusieurs dispositifs de persistance

Il existe diffrentes mthodes d'utilisation de multiples dispositifs de persistance pour les diffrents cas d'utilisation. Par exemple, utiliser plusieurs dispositifs la fois ou en slectionner un seul, entre plusieurs, des fins trs spcifiques.

Plusieurs volumes overlays diffrents (avec leurs propres fichiers persistence.conf) peuvent tre utilis au mme temps, mais si plusieurs volumes rendent le mme rpertoire persistant, un seul d'entre eux sera utilis. Si les deux sont imbriqu (un est un sous-rpertoire de l'autre) le premier sera mont avant le second de sorte qu'aucun ne sera cach par l'autre. Monter des lments personnalis imbriqu est problmatique s'ils sont numrs dans le mme fichier persistence.conf. Voir la page de manuel persistence.conf(5) pour savoir comment grer ce cas si vous en avez vraiment besoin (remarque: ce n'est gnralement pas le cas).

Un cas d'utilisation possible: Si vous souhaitez stocker les donnes de

l'utilisateur, c'est--dire /home et les donnees du superutilisateur, c'est--dire /root dans des partitions differentes, creer deux partitions avec l'etiquette persistence et ajouter un fichier persistence.conf dans chacun comme a # echo /home & persistence.conf pour la premiere partition qui permettra de sauver les fichiers de l'utilisateur et # echo /root & persistence.conf pour la seconde partition qui permettra de stocker les fichiers du superutilisateur. Enfin, utiliser le parametre d'amorage persistence.

facile de manipuler des partitions chiffrees avec live-boot. Pour utiliser luks , qui est le type de chiffrement pris en charge, vous devez installer cryptsetup tant sur la machine avec laquelle vous crez la partition chiffre et aussi dans le systme live avec lequel vous allez utiliser la partition persistante chiffre.

Pour installer cryptsetup sur votre machine:

```
# apt-get install cryptsetup
```

Pour installer cryptsetup dans votre systme live, ajouter vos listes de paquets:

```
$ lb config
$ echo "cryptsetup cryptsetup-initramfs" & config/package-lists/↵
  encryption.list.chroot
```

Une fois que vous avez votre systme live avec cryptsetup, vous avez essentiellement besoin de creer une nouvelle partition, la chiffrer et dmarrer avec les parametres persistence et persistence-encryption=luks. Nous aurions pu dj anticiper cette tape et ajoute ces parametres de dmarrage selon la procedure habituelle:

```
$ lb config --bootappend-live "boot=live components persistence ↵
  persistence-encryption=luks"
```

Si un utilisateur a besoin de stockages persistants multiples du mme type pour differents endroits ou essais, tel que private et work, le parametre de dmarrage persistence-label utilis en conjonction avec le parametre de dmarrage persistence permettra de multiples mais uniques supports persistants. Dans le cas o un utilisateur voudrait utiliser une partition persistante tiquete private, pour des donnees personnelles comme les marque-pages du navigateur ou d'autres types, il utiliserait les parametres de dmarrage: persistence persistence-label=private. Et pour stocker des donnees lies au travail, comme des documents, des projets de recherche ou d'autres types, il utiliserait les parametres de dmarrage: persistence persistence-label=work.

Il est important de se rappeler que chacun de ces volumes, private et work, a galement besoin d'un fichier persistence.conf dans sa racine. La page de manuel live-boot contient plus d'informations sur la faon d'utiliser ces etiquettes avec des noms anciens.

10.3.3 Utilisation de la persistance avec chiffrement

Utiliser la persistance signifie que certaines donnees sensibles peuvent tre exposes au risque. Surtout si les donnees persistantes sont stockes sur un dispositif portable tel qu'une cl USB ou un disque dur externe. C'est quand le chiffrement est plus pratique. Mme si la procedure compte peut paratre complique en raison du nombre d'etapes suivre, il est vraiment

Allons dans les dtails pour tous ceux qui ne connaissent pas bien le chiffrement. Dans l'exemple suivant, nous allons utiliser une partition sur une cl usb qui correspond au dispositif /dev/sdc2. S'il vous plat tre averti que vous devez dterminer quelle partition est celui que vous allez utiliser dans votre cas particulier.

La première étape est de brancher votre clé USB et de déterminer quel dispositif il est. La méthode recommandée pour lister les dispositifs dans live-manual est d'utiliser `ls -l /dev/disk/by-id`. Après cela, créer une nouvelle partition et la chiffrer avec un mot de passe comme suit :

```
# cryptsetup --verify -passphrase luksFormat /dev/sdc2
```

Ensuite, ouvrir la partition luks dans le mappeur de dispositifs virtuel. Utilisez n'importe quel nom que vous aimez. Nous utilisons `live` ici titre d'exemple :

```
# cryptsetup luksOpen /dev/sdc2 live
```

La prochaine étape est de remplir le dispositif avec des zéros avant de créer le système de fichiers :

```
# dd if=/dev/zero of=/dev/mapper/live
```

Maintenant, nous sommes prêts à créer le système de fichiers. Notez que nous ajoutons l'étiquette `persistence` de sorte que le dispositif est monté en tant que magasin de persistance au moment du démarrage.

```
# mkfs.ext4 -L persistence /dev/mapper/live
```

Pour continuer avec notre configuration, nous avons besoin de monter le dispositif, par exemple dans `/mnt`.

```
# mount /dev/mapper/live /mnt
```

Créer le fichier `persistence.conf` à la racine de la partition. Ceci est, comme expliqué précédemment, strictement nécessaire. Voir [Le fichier persistence.conf](#).

```
# echo "/ union" > /mnt/persistence.conf
```

Puis, démontez le point de montage :

```
# umount /mnt
```

Et éventuellement, bien qu'il pourrait être un bon moyen de sécuriser les données que nous venons d'ajouter à la partition, nous pouvons fermer le dispositif :

```
# cryptsetup luksClose live
```

Résumons la procédure. Jusqu'ici nous avons créé un système capable d'utiliser le chiffrement, qui peut être copié sur une clé USB comme expliqué dans [Copie d'une image ISO hybride sur une clé USB](#). Nous avons également créé une partition chiffrée, qui peut être située dans la même clé USB pour la porter autour et nous avons configuré la partition chiffrée pour être utilisée comme magasin de persistance. Alors maintenant, nous avons seulement besoin de démarrer le système live. Au moment du démarrage, `live-boot` nous demandera le mot de passe pour monter la partition chiffrée et l'utiliser pour la persistance.

Personnalisation de l'image binaire

```
include menu.cfg
default vesamenu.c32
prompt 0
timeout 50
```

11. Personnalisation de l'image binaire

11.1 Chargeurs d'amorage

live-build utilise syslinux et certains de ses drivs (selon le type d'image) comme chargeurs d'amorage par dfaut. Vous pouvez facilement les personnaliser pour rpondre vos besoins.

Pour utiliser un thme complet, copiez /usr/share/live/build/bootloaders dans config/bootloaders et modifiez les fichiers l. Si vous ne voulez pas modifier toutes les configurations du chargeur d'amorage prises en charge, il suffit de fournir une copie locale personnalise d'un des chargeurs, par exemple copiez la configuration d'isolinux dans config/bootloaders/isolinux, selon votre cas d'utilisation.

Lors de la modification d'un des thmes par dfaut, si vous souhaitez utiliser une image de fond personnalise qui sera affiche avec le menu de dmarrage, ajouter une image splash.png de 640x480 pixels. Ensuite, supprimer le fichier splash.svg.

Il y a beaucoup de possibilits quand il s'agit de faire des changements. Par exemple, les drivs de syslinux sont configurs par dfaut avec un timeout de 0 (zro), ce qui signifie qu'ils se mettront en pause indfiniment leur cran de dmarrage jusqu' ce que vous pressiez une touche.

Pour modifier le dlai de dmarrage d'une image iso-hybrid, vous pouvez modifier un fichier isolinux.cfg en prcisant le timeout dans les units de 1/10 secondes. Un isolinux.cfg modifi pour dmarrer cinq secondes plus tard serait semblable ceci:

11.2 Mtadonnes ISO

En crant une image binaire ISO9660, vous pouvez utiliser les options suivantes pour ajouter diffrentes mtadonnes textuelles pour votre image. Cela peut vous aider facilement identifier la version ou la configuration d'une image sans la dmarrer.

LB'ISO'APPLICATION/–iso-application NAME: Cela devrait dcire l'application qui sera sur l'image. Le nombre maximum de caractres pour ce champ est 128.

LB'ISO'PREPARER/–iso-preparer NAME: Cela devrait dcire le prparateur de l'image, gnralement avec certains dtails de contact. Le dfaut de cette option est la version de live-build que vous utilisez, ce qui peut faciliter le dbogage plus tard. Le nombre maximum de caractres pour ce champ est 128.

LB'ISO'PUBLISHER/–iso-publisher NAME: Ce doit dcire l'diteur de l'image, gnralement avec certains dtails de contact. Le nombre maximum de caractres pour ce champ est 128.

LB'ISO'VOLUME/–iso-volume NAME: Cela devrait indiquer l'ID de volume de l'image. Il est utilis comme une tiquette visible par l'utilisateur sur certaines plateformes comme Windows et Apple Mac OS. Le nombre maximum de caractres pour ce champ est 32.

Personnalisation de l'installateur Debian

12. Personnalisation du contenu pour l'installateur Debian

Les images des systèmes live peuvent être intégrées avec l'installateur Debian. Il y a un certain nombre de types d'installation différents, variant en fonction de ce qui est inclus et de la façon dont fonctionne l'installateur.

Veuillez noter l'utilisation prudente des lettres majuscules pour désigner l'Installateur Debian dans cette section - lorsqu'il est utilisé comme cela, nous faisons explicitement référence à l'installateur officiel pour le système Debian. On le voit souvent abrégé en d-i.

12.1 Types d'installateur Debian

Les trois principaux types de programme d'installation sont:

Installateur Debian Normal : C'est une image du système live avec un noyau et initrd spars qui (lorsqu'ils sont sélectionnés à partir du chargeur d'amorage approprié) se lancent dans une instance d'installateur Debian standard, exactement comme si vous aviez téléchargé une image de CD de Debian et l'aviez démarré. Les images contenant un système live et un installateur indépendant sont souvent appelées images combinées.

Sur ces images, Debian est installé par l'extraction et l'installation de paquets .deb à l'aide de debootstrap, à partir des supports locaux ou du réseau, résultant en un système Debian par défaut installé sur le disque dur.

Tout ce processus peut être préconfiguré et personnalisé d'un certain nom-

bre de façons. Consultez les pages correspondantes dans le manuel de l'Installateur Debian pour plus d'informations. Une fois que vous avez un fichier de préconfiguration qui fonctionne, live-build peut automatiquement l'ajouter à l'image et l'activer pour vous.

Installateur Debian Live : C'est une image du système live avec un noyau et initrd spars qui (lorsqu'ils sont sélectionnés à partir du chargeur d'amorage approprié) se lancent dans une instance de l'installateur Debian.

L'installation continue de manière identique à l'installation normale décrite ci-dessus, mais à l'étape de l'installation des paquets, au lieu d'utiliser debootstrap pour aller chercher et installer des paquets, l'image du système de fichiers live est copiée vers la cible. Ce résultat est obtenu avec un udeb spécial appelé live-installer.

Après cette étape, l'installateur Debian continue normalement, en installant et configurant des éléments tels que les chargeurs d'amorage et les utilisateurs locaux, etc.

Remarque: Pour prendre en charge les deux options `installateur normal` et `live` dans le chargeur d'amorage du même support live, vous devez désactiver `live-installer` en utilisant la préconfiguration `live-installer/enable=false`.

Installateur Debian de bureau : Indépendamment du type d'installateur Debian inclus, d-i peut être lancé à partir du bureau en cliquant sur une icône, ce qui est plus facile à utiliser dans certaines situations. Pour pouvoir en faire usage, le paquet `debian-installer-launcher` doit être inclus.

Notez que, par défaut, `live-build` n'inclut pas les images de l'installateur Debian dans les images, il doit être spécifiquement activé avec `lb config`. De même, veuillez noter que pour que l'installateur de bureau fonctionne, le noyau du système live doit correspondre au noyau que d-i utilise pour l'architecture indiquée. Par exemple:

```
$ lb config --debian-installer live
$ echo debian-installer-launcher && config/package-lists/my.list <->
chroot
```

12.2 Personnalisation de l'installateur Debian par préconfiguration

As described in the Debian Installer Manual, Appendix B at <https://www.debian.org/releases/stable/amd64/apb.en.html>, Preseeding provides a way to set answers to questions asked during the installation process, without having to manually enter the answers while the installation is running. This makes it possible to fully automate most types of installation and even offers some features not available during normal installations. This kind of customization is best accomplished with live-build by placing the configuration in a preseed.cfg file included in config/includes.installer/. For example, to preseed setting the locale to en`US:

```
$ echo "d-i debian-installer/locale string en`US" "
&& config/includes.installer/preseed.cfg"
```

12.3 Personnalisation de contenu pour l'Installateur Debian

des fins expérimentales ou de débogage, vous pouvez inclure des paquets udeb d-i construits localement. Placez-les dans config/packages.binary/ pour les inclure dans l'image. Plusieurs fichiers supplémentaires ou de remplacement et plusieurs répertoires peuvent aussi être inclus dans l'initrd de l'installateur, d'une manière similaire **Live/chroot local includes** en plaçant le contenu dans config/includes.installer/.

Projet

Contribuer au projet

13. Contribuer au projet

Lorsque vous soumettez une contribution, veuillez indiquer clairement le copyright et inclure toute mention légale relative à la licence. Notez que pour être acceptée, la contribution doit être déposée sous la même licence que le reste du document, c'est-à-dire la GPL version 3 ou ultérieure.

Contributions to the project, such as translations and patches, are greatly welcome. Anyone can send merge requests. The projects are hosted on Salsa: <https://salsa.debian.org/live-team> follow Salsa's documentation for instructions on how to contribute.

Même si toutes les livraisons pourraient être revues, nous vous demandons d'utiliser votre bon sens et de faire bonnes livraisons avec de bons messages de livraison.

Veuillez écrire les commentaires de livraison à l'aide de phrases complètes, en commençant par une majuscule et en terminant par un point, et avec la forme 'Fixing/Adding/Removing/Correcting/Translating/...'.

Écrivez de bons messages de livraison. La première ligne doit être un résumé précis du contenu du commit qui sera inclus dans le changelog. Si vous avez besoin de faire quelques explications supplémentaires, écrivez-les au-dessous en laissant une ligne vide après la première, puis une autre ligne vide après chaque paragraphe. Les lignes des paragraphes ne doivent pas dépasser 80 caractères.

Faites des livraisons de façon atomique, c'est-à-dire, ne mélangez pas des choses sans liens entre elles dans la même livraison. Faites un commit différent pour chaque modification que vous apportez.

13.1 Traduction des pages de manuel

Vous pouvez également contribuer au projet en travaillant à la traduction des pages man pour les différents paquets live-* que le projet maintient. La procédure est différente suivant que vous démarriez une nouvelle traduction ou que vous continuiez à travailler sur une traduction dj existante :

Continuer avec une traduction dj existante

Si vous voulez maintenir la traduction d'une langue dj existante, vous devez effectuer vos modifications dans le ou les fichier(s) manpages/po/\$-LANGUAGE"/*.po puis, lancer la commande make rebuild depuis le répertoire manpages/. Ceci mettra à jour les pages de manuel dans manpages/\$-LANGUAGE"/*

Commencer une nouvelle traduction à partir de zéro

Pour ajouter une nouvelle traduction de l'une des pages de manuel du projet, il vous faut suivre une procédure similaire. Elle peut être résumée comme suit :

Ouvrez le ou les fichiers(s) manpages/pot/ dans votre éditeur favori, comme poedit, et sauvegardez-le en tant que fichier dans manpages/po/\$-LANGUAGE"/. (Vous devrez créer votre répertoire \$-LANGUAGE"/).

Lancez make rebuild depuis l'intérieur du répertoire manpages/ pour créer les fichiers manpages/\$-LANGUAGE"/ contenant les pages de manuel.

Souvenez-vous que vous devrez ajouter tous les répertoires et les fichiers, puis faire le commit, et finalement, pousser sur le serveur git.

Signaler des bogues

14. Signaler des bogues

Les systèmes live sont loin d’être parfaits, mais nous voulons les rendre aussi parfaits que possible avec votre aide. N’hésitez pas signaler un bogue. Il est préférable de remplir un rapport deux fois plus que jamais. Toutefois, ce chapitre contient des recommandations pour présenter de bons rapports de bogues.

Pour les impatientes:

First check whether the bugs has been reported already. You can see the full list of bugs that are assigned to the live-team at <https://bugs.debian.org/cgi-bin/pkgreport.cgi?maint=debian-live%40lists.debian.org>.

Before submitting a bug report always try to reproduce the bug with the most recent versions of the packages of live-build, live-boot, live-config and live-tools that you’re using.

Essayez de donner des informations aussi précises que possible sur le bogue. Cela comprend (au moins) la version de live-build, live-boot, live-config et live-tools, de la distribution utilisée et du système live que vous construisez.

14.1 Problèmes connus

Currently known issues are listed in the BTS at <https://bugs.debian.org/cgi-bin/pkgreport.cgi?maint=debian-live%40lists.debian.org>.

Note: Since Debian testing and Debian unstable distributions are moving targets, when you specify either of them as the target system distribution, a successful build may not always be possible.

Si cela vous pose trop de difficulté, ne construisez pas un système basé sur testing ou unstable, mais utilisez plutôt stable. live-build utilise toujours la version stable par défaut.

It is out of the scope of this manual to train you to correctly identify and fix problems in packages of the development distributions, however, you can always try the following: If a build fails when the target distribution is testing, try unstable. If unstable does work, revert to testing and pin the newer version of the failing package from unstable (see [APT pinning](#) for details).

14.2 Effectuer une recherche

Avant de présenter le bogue, veuillez rechercher sur le web le message d’erreur ou un symptôme particulier que vous obtenez. Comme il est hautement improbable que vous soyez la seule personne faisant l’expérience d’un problème particulier, il y a toujours une chance qu’il ait été discuté ailleurs, et qu’une solution possible, un correctif, ou une solution de contournement ait été proposée.

Vous devez prêter une attention particulière la liste de diffusion du système live, ainsi qu’à la page d’accueil, car elles sont susceptibles de contenir des informations jour. Si ces informations existent, incluez toujours les références au sein de vos rapports de bogues.

En outre, vous devriez vérifier les listes de bogues en cours de live-build, live-boot, live-config et live-tools pour voir si quelque chose de semblable n’a pas déjà été signalé.

14.3 Reconstruire à partir de zéro

Afin de vous assurer qu’un bogue en particulier n’est pas causé par un

système mal construit, veuillez toujours reconstruire l'ensemble du système live à partir de zéro pour voir si le bogue est reproductible.

14.4 Utiliser des paquets mis à jour

Using outdated packages can cause significant problems when trying to reproduce (and ultimately fix) your problem. Make sure your build system is up-to-date and any packages included in your image are up-to-date as well. If possible, try to reproduce the bug with the newest code from source, see [Installation](#) for details.

14.5 Recueillir l'information

Veuillez fournir assez d'informations avec votre rapport. Incluez au moins la version exacte de live-build où le bogue est rencontré et les mesures pour le reproduire. Veuillez utiliser votre bon sens et incluez d'autres renseignements pertinents, si vous pensez que cela pourrait aider résoudre le problème.

Pour tirer le meilleur parti de votre rapport de bogue, nous avons au moins besoin des informations suivantes:

L'architecture du système hôte

Distribution du système hôte

Version de live-build sur le système hôte

Version de debootstrap sur le système hôte

L'architecture du système live

Rpartition du système live

Version de live-boot sur le système live

Version de live-config sur le système live

Version de live-tools sur le système live

Vous pouvez générer un journal du processus de construction en utilisant la commande `tee`. Nous recommandons de faire cela automatiquement avec un script `auto/build` (voir [Gestion d'une configuration](#) pour les détails).

```
# lb build 2i&1 -- tee build.log
```

Au démarrage, live-boot et live-config stockent un journal dans `/var/log/-live/boot.log`. Vérifiez-les pour des messages d'erreur.

Par ailleurs, pour cartier d'autres erreurs, c'est toujours une bonne idée de faire un tar de votre répertoire `config/` et de le télécharger quelque part (ne pas l'envoyer en pièce jointe à la liste de diffusion), de sorte que nous puissions essayer de reproduire les erreurs que vous rencontrez. Si cela est difficile (en raison par exemple de la taille) vous pouvez utiliser la sortie de `lb config --dump` qui produit un résumé de votre arbre de config (c'est-à-dire les listes des fichiers dans les sous-répertoires de `config/` mais ne les inclut pas).

N'oubliez pas d'envoyer tous les journaux produits avec les paramètres régionaux anglais. Par exemple, exécutez vos commandes live-build précédées par `LC_ALL=C` ou `LC_ALL=en_US`.

14.6 Isoler le cas qui choue, si possible

Si possible, isolez le cas qui choue au plus petit changement possible. Il n'est pas toujours facile de faire cela, donc si vous ne pouvez pas le gérer pour votre rapport, ne vous inquiétez pas. Toutefois, si vous planifiez bien votre cycle de développement, en utilisant de petits ensembles de changements par itération, vous pourriez être capable d'isoler le problème en construisant une configuration simple de base qui correspond étroitement à

configuration réelle avec seulement le changement de casse ajout. S'il est difficile de trier vos modifications qui cassent, il est possible que vous incluiez trop dans chaque ensemble de modifications et vous devriez développer en petits incréments.

14.7 Utiliser le paquet adquat pour rapporter un bogue

En général, vous devez signaler les erreurs de construction contre le paquet live-build, les erreurs lors du démarrage contre live-boot, et les erreurs d'exécution contre live-config. Si vous n'êtes pas sûr du paquet approprié ou si vous avez besoin d'aide avant de soumettre un rapport de bogue, veuillez signaler le bogue contre le pseudo-paquet debian-live. Nous le réattribuerons s'il y a lieu.

Toutefois, nous apprécierions que vous essayiez de le réduire en fonction de l'endroit où le bogue apparaît.

14.7.1 Pendant la construction durant l'amorçage

live-build amorçait d'abord un système Debian de base avec debootstrap. Si un bogue apparaît ici, vérifiez si l'erreur est liée à un paquet Debian spécifique (plus probable), ou si elle est liée à l'outil d'amorçage lui-même.

Dans les deux cas, ce n'est pas un bogue dans le système live, mais plutôt dans Debian lui-même que probablement nous ne pouvons pas le résoudre directement. Veuillez signaler un bogue sur l'outil d'amorçage ou du paquet de faille.

14.7.2 Pendant la construction durant l'installation de paquets

live-build installe des paquets supplémentaires de l'archive Debian et en fonction de la distribution Debian utilisée et l'état quotidien de l'archive,

il peut chouer. Si un bogue apparaît ici, vérifiez si l'erreur est également reproductible sur un système normal.

Si c'est le cas, ce n'est pas un bogue dans le système live, mais plutôt dans Debian – veuillez envoyer le rapport sur le paquet de faille. L'exécution de debootstrap dépend du système de construction ou l'exécution de live-bootstrap – debug vous donnera plus d'informations.

Aussi, si vous utilisez un miroir local et/ou un proxy et vous rencontrez un problème, veuillez toujours le reproduire en amorçant d'abord sur un miroir officiel.

14.7.3 Pendant le démarrage

Si votre image ne démarre pas, veuillez le signaler à la liste de diffusion avec les informations demandées dans **Recueillir l'information**. N'oubliez pas de mentionner, comment/quand l'image a chuté, soit en virtualisation ou sur du matériel réel. Si vous utilisez une technologie de virtualisation de quelque sorte, veuillez toujours tester sur du matériel réel avant de signaler un bogue. Fournir une copie d'écran de l'échec est également très utile.

14.7.4 Pendant l'exécution

If a package was successfully installed, but fails while actually running the Live system, this is probably a bug in live-config.

14.8 Où rapporter les bogues

Le Debian Live Project conserve la trace de tous les bogues dans le système de suivi des bogues (BTS). Pour plus d'informations sur la façon d'utiliser le système, veuillez consulter <https://bugs.debian.org/>. Vous pouvez également

soumettre les bogues en utilisant la commande `reportbug` du paquet du mme nom.

748

Veillez noter que les bogues trouvs dans les distributions drives de Debian (comme Ubuntu et autres) ne doivent pas tre rapports au BTS de Debian, sauf s'ils peuvent tre galement reproduits sur un systme Debian en utilisant les paquets Debian officiels.

Style de code

15. Style du code

Ce chapitre documente le style du code utilis dans les systmes live.

15.1 Compatibilit

Avoid bashisms, the codebase must be POSIX compliant and thus universally compatible.

Furthermore it must comply with the version of the POSIX specification chosen by the current Debian Policy.

Vous pouvez vrifier vos scripts avec ‘sh -n’ et ‘checkbashisms’.

Assurez-vous que tout le code fonctionne avec ‘set-e ‘.

15.2 Indentation

Utilisez toujours des tabulations au lieu des espaces.

Keep case branch terminators (;;) aligned with the content of the branch, rather than the branch entry.

Bien:

```
case "$1" in
    foo)
        foobar
        ;;
    bar)
        foobar
        ;;
```

esac

15.3 Adaptateur

Generally, lines should be 80 chars at maximum.

Placement of keywords like then and do should be chosen with good judgement with respect to clutter and readability. For small bits of code in particular it should be preferred to have them on the same line as the prior keyword they relate to (if; for; etc). Only place on the next line where it makes good sense to do so; typically this might only be to comply with maximum line length restrictions. One situation where they should always be placed on the next line is where what they follow is broken up onto multiple lines, and thus it being on a new line creates clear separation between that and the body of code following it. I.e. :

Preferred:

```
if foo; then
    bar
fi

for FOO in $ITEMS; do
    bar
done

if [ "$MYLOCATIONVARIABLE" = "something" ] && [ -e "$MYOUTPUTFILE" ]
then
    MYOTHERVARIABLE="$(some`bin $-FOOBAR" — awk -F' ' — `
    print $1 "`)"
fi

if [ "$MYFOO" = "something" ] && [ -e "path/$FILE1" ] —
[ "$MYBAR" = "something`else" ] && [ $-ALLOW = "true" ]
then
    foobar
```

```
fi
```

Awful:

776

Less ideal:

777

```
if [ "$MYLOCATIONVARIABLE" = "something" ] && [ -e "$MYOUTPUTFILE" ]; then
    MYOTHERVARIABLE="$(some`bin $-FOOBAR" — awk -F' ' — ↵
    print $1 "')"
fi
```

```
Foo ()
-
    bar
"
```

Horrible:

```
if [ "$MYLOCATIONVARIABLE" = "something" ] && [ -e "$MYOUTPUTFILE" ] — [ "$MYLOCATIONVARIABLE" = "something" ↵
-else" ] && [ -e "$MYOUTPUTFILE2" ]; then
    MYOTHERVARIABLE="$(some`bin $-FOOBAR" — awk -F' ' — ↵
    print $1 "')"
fi
```

15.4 Variables

Les variables sont toujours en lettres majuscules.

Config variables used in live-build should start with an LB` prefix.

Local function variables should be restricted to local scope.

Les variables en relation avec un paramtre de dmarrage dans live-config commencent par LIVE`.

Toutes les autres variables dans live-config commencent par le prfixe `.

Utilisez des accolades autour des variables; crivez par exemple `$-FOO` au lieu de `$FOO`.Always protect variables with quotes to respect potential whitespaces (except where necessary to achieve correct word splitting): write `$-FOO` not `$-FOO`.

Pour des raisons de cohrence, utilisez toujours les guillemets lors de l'attribution des valeurs aux variables:

Mal:

```
FOO=bar
```

Prefer placing the opening brace of a function on a new line (for consistency with established style), and keep the braces aligned with the function name:

Bien:

```
Foo ()
-
    bar
"
```

Bad (inconsistent with existing style):

```
Foo () -
    bar
"
```


Bien:

789

790

```
FOO="bar"
```

791 If multiple variables are used, prefer quoting the full expression:

792 Typically bad:

793

```
if [ -f "$FOO"/foo/"$BAR"/bar ]; then
    foobar
fi
```

794 Bien:

795

```
if [ -f "$FOO"/foo/"$BAR"/bar ]; then
    foobar
fi
```

796 15.5 Autres

797 Prefer `—` (without the surround quotes) as a separator in calls to `sed`,
e.g. `sed -e 's—'` (without `"`).

798 Don't use the `test` command for comparisons or tests, use `[` and `]`
(without `"`); e.g. `if [-x /bin/foo]; ...` and not `if test -x /bin/foo; ...`

799 Use `case` wherever it makes code more readable than conditional checks
(`if foo; ...` and tests without the actual `if` keyword, e.g. `[-e $FILE]`
——`exit 0`).

800 Use `Foo`bar` style names for functions, i.e. a capital first letter, then
all lowercase, with sensible use of underscores for better readability.

Exemples

Exemples

16. Exemples

Ce chapitre présente des exemples de constructions pour des cas d'utilisation spécifiques avec des systèmes live. Si vous débutez dans la construction de vos propres images des systèmes live, nous vous recommandons de regarder d'abord les trois tutoriels la suite car chacun d'entre eux enseigne de nouvelles techniques qui vous aideront à utiliser et comprendre les exemples restants.

16.1 Utiliser les exemples

Pour utiliser ces exemples, vous avez besoin d'un système pour les construire qui doit répondre aux exigences numérotées dans [Exigences](#) et sur lequel live-build est installé comme décrit dans [Installation de live-build](#).

Notez que, pour des raisons de concision, dans ces exemples, nous n'indiquons pas de miroir local à utiliser pour la construction. Vous pouvez accélérer considérablement les téléchargements si vous utilisez un miroir local. Vous pouvez indiquer ces options lorsque vous utilisez `lb config`, tel que décrit dans [Miroirs de distribution utilisés pendant la construction](#), ou pour plus de commodité, fixez par défaut votre système de construction dans `/etc/live/build.conf`. Il suffit de créer ce fichier et de définir les variables `LB_MIRROR*` correspondantes à votre miroir préféré. Tous les autres miroirs utilisés dans la construction seront choisis par défaut à partir de ces valeurs. Par exemple:

```
LB_MIRROR_BOOTSTRAP="http://mirror/debian/"
LB_MIRROR_CHROOT_SECURITY="http://mirror/debian-security/"
LB_MIRROR_CHROOT_BACKPORTS="http://mirror/debian-updates/"
```

16.2 Tutoriel 1: Une image par défaut

Cas d'utilisation: Créer une image simple d'abord, en apprenant les bases de live-build.

Dans ce tutoriel, nous construirons une image ISO hybride par défaut contenant uniquement des paquets de base (pas de Xorg) et quelques paquets de prise en charge live, en guise de premier exercice dans l'utilisation de live-build.

Vous ne pouvez pas faire plus simple que cela:

```
$ mkdir tutorial1 ; cd tutorial1 ; lb config
```

Examinez le contenu du répertoire `config/` si vous le souhaitez. Vous verrez ici une arborescence de configuration, partiellement personnalisée ou, dans ce cas, utilisée immédiatement pour construire une image par défaut.

Maintenant, en tant que superutilisateur, construisez l'image en enregistrant un journal avec `tee`.

```
# lb build 2i&1 — tee build.log
```

Assuming all goes well, after a while, the current directory will contain `live-image-amd64.hybrid.iso`. This ISO hybrid image can be booted directly in a virtual machine as described in [Testing an ISO image with Qemu](#) and [Testing an ISO image with VirtualBox](#), or else imaged onto optical media or a USB flash device as described in [Burning an ISO image to a physical medium](#) and [Copying an ISO hybrid image to a USB stick](#), respectively.

16.3 Tutoriel 2: Un utilitaire d'un navigateur Web

Cas d'utilisation: Créer l'image d'un utilitaire de navigation Web, en apprenant à appliquer des personnalisations.

Dans ce tutoriel, nous allons créer une image utilisable comme un utilitaire de navigation web, ce qui servira d'introduction à la personnalisation d'images live.

```
$ mkdir tutorial2
$ cd tutorial2
$ lb config
$ echo "task-lxde-desktop firefox-esr" >> config/package-lists/my.list.chroot
```

Notre choix de LXDE pour cet exemple reflète notre volonté de fournir un environnement de bureau minimal, puisque le but de l'image est l'utilisation unique que nous avons en tête, le navigateur web. On pourrait aller encore plus loin et offrir une configuration par défaut pour le navigateur web dans `config/includes.chroot/etc/iceweasel/profile/`, ou des paquets de prise en charge supplémentaires pour visualiser différents types de contenu web, mais nous laissons cela en exercice pour le lecteur.

Construisez l'image, encore une fois en tant que superutilisateur, et gardez un journal comme dans [Tutoriel 1](#):

```
# lb build 2i&1 --tee build.log
```

Encore une fois, vérifiez que l'image est OK et faites un test, comme dans [Tutoriel 1](#):

16.4 Tutoriel 3: Une image personnalisée

Cas d'utilisation: Créer un projet pour construire une image personnalisée,

contenant vos logiciels préférés à emporter avec vous sur une clé USB où que vous alliez, et permettant dans des révisions successives selon les changements de vos besoins et de vos préférences.

Puisque nous allons changer notre image personnalisée pendant un certain nombre de révisions, et que nous voulons suivre ces changements, essayer des choses expérimentalement et éventuellement les annuler si les choses ne fonctionnent pas, nous garderons notre configuration dans le populaire système de contrôle de version git. Nous allons également utiliser les meilleures pratiques d'autoconfiguration via auto scripts tel que décrit dans [Gestion d'une configuration](#).

16.4.1 Première révision

```
$ mkdir -p tutorial3/auto
$ cp /usr/share/doc/live-build/examples/auto/* tutorial3/auto/
$ cd tutorial3
```

ditez `auto/config` comme suit:

```
#!/bin/sh

lb config noauto "
--distribution stable "
"$@"
```

Exécutez `lb config` pour générer l'arbre de configuration, en utilisant le script `auto/config` qu'on a créé:

```
$ lb config
```

Remplissez maintenant votre liste de paquets locaux:

```
$ echo "task-lxde-desktop spice-vdagent hexchat" && config/package-  
lists/my.list.chroot
```

First, `-distribution stable` ensures that `stable` is used instead of the default `-testing`. Second, we have added `spice-vdagent` for easier testing the image in `qemu`. And finally, we have added an initial favourite package: `hexchat`.

Maintenant, construisez l'image:

```
# lb build
```

Notez que contrairement aux deux premiers tutoriels, nous n'avons plus besoin de taper `2&1 —tee build.log` parce que cela est maintenant inclus dans `auto/build`.

Une fois que vous avez testé l'image (comme dans [Tutoriel 1](#)) et vous êtes satisfait de son fonctionnement, il est temps d'initialiser notre dépôt git, en n'ajoutant que les scripts `auto` que nous avons juste créés, puis de faire le premier commit:

```
$ git init  
$ cp /usr/share/doc/live-build/examples/gitignore .gitignore  
$ git add .gitignore auto config  
$ git commit -m "Initial import."
```

16.4.2 Deuxième révision

In this revision, we're going to clean up from the first build, replace the `smplayer` package with `vlc` package, rebuild, test and commit.

La commande `lb clean` va nettoyer tous les fichiers générés par la construction précédente l'exception du cache, ce qui évite d'avoir à recharger les paquets. Cela garantit que la commande `lb build` suivante exécutera à nouveau toutes les étapes pour régénérer les fichiers de notre nouvelle configuration.

```
# lb clean
```

Now install the `vlc` package before the `lxde` package chooses between `smplayer`, `vlc` and `mplayer-gui` in our local package list in `config/package-lists/my.list.chroot`:

```
$ echo "vlc task-lxde-desktop spice-vdagent hexchat" && config/  
package-lists/my.list.chroot
```

Construisez à nouveau:

```
# lb build
```

Testez et, quand vous êtes satisfait, faites le commit de la prochaine révision:

```
$ git commit -a -m "Replacing smplayer with vlc."
```

Bien sûr, des changements plus compliqués de la configuration sont possibles, comme l'ajout de fichiers dans les sous-répertoires de `config/`. Quand vous livrez de nouvelles révisions, prenez soin de ne pas modifier la main ou d'envoyer dans le dépôt les fichiers de niveau supérieur dans `config` contenant les variables `LB*`, car ce sont aussi des produits de la création et ils sont toujours nettoyés par `lb clean` et recréés avec `lb config` via leurs scripts `auto` respectifs.

Nous sommes arrivés à la fin de notre série de tutoriels. Alors que de nombreux types de personnalisations sont possibles, même en n'utilisant que les fonctionnalités explorées dans ces exemples simples, une variété presque infinie d'images différentes peut être créée. Les autres exemples de cette section couvrent plusieurs autres cas d'utilisation tirés des expériences recueillies chez les utilisateurs des systèmes live.

16.5 Un client kiosque VNC

Cas d'utilisation : Créer une image avec live-build pour démarrer directement sur un serveur VNC.

Créez un répertoire de construction et créez une configuration à l'intérieur, désactivez `recommends` pour faire un système minimal. Puis créez deux listes initiales de paquets : la première genre par un script fourni par live-build nommé `Packages` (voir [Générer des listes de paquets](#)), et la seconde incluant `xorg`, `gdm3`, `metacity` et `xvnc4viewer`.

```
$ mkdir vnc-kiosk-client
$ cd vnc-kiosk-client
$ lb config --apt-recommends false
$ echo '! Packages Priority standard' > config/package-lists/standard.list.chroot
$ echo "xorg gdm3 metacity xtightvncviewer" > config/package-lists/my.list.chroot
```

Comme il est expliqué dans [Régler APT pour économiser de l'espace](#), il se peut que vous deviez ajouter quelques paquets recommandés pour faire fonctionner votre image correctement.

Une façon facile d'énumérer les paquets recommandés est d'utiliser `apt-cache`. Par exemple :

```
$ apt-cache depends live-config live-boot
```

Dans cet exemple, nous avons découvert que nous devons inclure plusieurs paquets recommandés par `live-config` et `live-boot` : `user-setup` pour l'autologin et `sudo` un logiciel essentiel pour arrêter le système. En outre, il pourrait être utile d'ajouter `live-tools` pour copier l'image dans la RAM et `eject` pour jeter le support live. Alors :

```
$ echo "live-tools user-setup sudo eject" > config/package-lists/recommends.list.chroot
```

Après cela, créez le répertoire `/etc/skel` dans `config/includes.chroot` avec un fichier `.xsession` personnalisé pour l'utilisateur par défaut qui va lancer `metacity` et `xvncviewer`, en reliant le port 5901 sur un serveur 192.168.1.2 :

```
$ mkdir -p config/includes.chroot/etc/skel
$ cat > config/includes.chroot/etc/skel/.xsession ;; EOF
#!/bin/sh

/usr/bin/metacity &
/usr/bin/xvncviewer 192.168.1.2:1

exit
EOF
```

Construire l'image :

```
# lb build
```

Amusez-vous bien !

16.6 A minimal image for a 512MB USB key

Use case: Create a default image with some components removed in order to fit on a 512MB USB key with a little space left over to use as you see fit.

When optimizing an image to fit a certain media size, you need to understand the tradeoffs you are making between size and functionality. In this example, we trim only so much as to make room for additional material within a 512MB media size, but without doing anything to destroy the integrity of the packages contained within, such as the purging of locale data via the `localepurge` package, or other such intrusive optimizations. Of particular note, we use `--debootstrap-options` to create a minimal system from scratch and `--binary-image hdd` to create an image that can be copied to a USB key.

```
$ lb config --binary-image hdd --apt-indices false --apt-recommends false --debootstrap-options "--variant=minbase" --firmware-chroot false --memtest none
```

Pour faire que l'image fonctionne correctement, nous devons ajouter nouveau au moins deux paquets recommandés qui sont laissés de côté par l'option `--apt-recommends false`. Voir [Régler APT pour économiser de l'espace](#)

```
$ echo "user-setup sudo" && config/package-lists/recommends.list.chroot
```

Additionally, you'll want to have network access, so another two recommended packages need to be re-added:

```
$ echo "ifupdown isc-dhcp-client" && config/package-lists/recommends.list.chroot
```

Maintenant, construisez l'image de la manière habituelle:

```
# lb build 2i&1 --tee build.log
```

On the author's system at the time of writing this, the above configuration produced a 298MiB image. This compares favourably with the 380MiB image produced by the default configuration in [Tutorial 1](#), when `--binary-image hdd` is added.

Laisser tomber les index d'APT avec `--apt-indices false` permet aussi d'économiser une bonne quantité d'espace, le compromis tant que vous devez faire `apt-get update` avant d'utiliser `apt` dans le système live. Abandonner les paquets recommandés avec `--apt-recommends false` économise de l'espace supplémentaire, au détriment de certains paquets dont vous vous attendriez autrement qu'ils soient là. `--debootstrap-options --variant=minbase` construit un système minimal dès le début. L'utilisation de `--firmware-chroot false` n'inclut pas automatiquement les paquets de micrologiciels. Finalement, `--memtest none` prévient l'installation d'un testeur de mémoire.

Note: A minimal system can also be achieved using hooks, like for example the `stripped.hook.chroot` hook found in `/usr/share/doc/live-build/examples/hooks`. It may shave off additional small amounts of space and produce an image of 277MiB. However, it does so by removal of documentation and other files from packages installed on the system. This violates the integrity of those packages and that, as the comment header warns, may have unforeseen consequences. That is why using a minimal `debootstrap` is the recommended way of achieving this goal.

16.7 Un bureau GNOME localisé avec un installateur

Cas d'utilisation: Créer une image de bureau GNOME, localisée pour la

Suisse et incluant un installateur.

884 We want to make an iso-hybrid image using our preferred desktop, in
this case GNOME, containing all of the same packages that would be
installed by the standard Debian installer for GNOME.

885 Notre premier probl me est la dcouverte des noms des t ches appropri es.
Actuellement, live-build ne peut pas aider   faire cela. Alors que nous
pourrions tre chanceux et trouver ces noms par essais et erreurs, il existe
un outil, grep-dctrl, qui peut tre utilis  pour dcouvrir les descriptions des
t ches dans tasksel-data. Pour la pr paration, assurez-vous d’avoir ces deux
outils:

886

```
# apt-get install dctrl-tools tasksel-data
```

887 Maintenant, nous pouvons rechercher les t ches appropri es, d’abord
avec:

888

```
$ grep -dctrl -FTest-lang de /usr/share/tasksel/descs/debian-tasks.↵
desc -sTask
Task: german
```

889 Par cette commande, nous dcouvrons que la t che est appele, assez claire-
ment, german. Maintenant, pour trouver les t ches lies:

890

```
$ grep -dctrl -FEnhances german /usr/share/tasksel/descs/debian-↵
tasks.desc -sTask
Task: german-desktop
Task: german-kde-desktop
```

891 Pendant le dmarrage, nous allons g ner la locale de’CH.UTF-8 et slec-
tionner la disposition de clavier ch . Maintenant, nous allons mettre les
morceaux ensemble. En nous rappelant gr ce **Utilisation des m paquets**

que les m paquets sont pr fix s task-, nous pr cisons ces param tres de la
langue pendant l’amorag , puis nous ajoutons les paquets de priorit  stan-
dard et tous nos m paquets dcouv rts   notre liste de paquets comme
suit:

892

```
$ mkdir live-gnome-ch
$ cd live-gnome-ch
$ lb config “
--bootappend-live "boot=live components locales=de'CH.UTF-8 ↵
keyboard-layouts=ch" “
--debian-installer live
$ echo '! Packages Priority standard'   config/package-lists/↵
standard.list.chroot
$ echo task-gnome-desktop task-german task-german-desktop    ↵
config/package-lists/desktop.list.chroot
$ echo debian-installer-launcher    config/package-lists/installer↵
.list.chroot
```

Note that we have included the debian-installer-launcher package to 893
launch the installer from the live desktop.

Appendix

Guide de style

17. Guide de style

17.1 Lignes directrices pour les auteurs

Cette section traite des considérations générales à prendre en compte lors de la rédaction de la documentation technique pour live-manual. Elles sont divisées en caractéristiques linguistiques et en procédures recommandées.

Remarque: Les auteurs doivent lire d'abord **Contribuer ce document**

17.1.1 Caractéristiques linguistiques

Utilisez un anglais simple

Gardez l'esprit qu'un pourcentage élevé de vos lecteurs ne sont pas de langue maternelle anglaise. Donc, en règle générale, essayez d'utiliser des phrases significatives courtes, suivies d'un point.

Cela ne signifie pas que vous devez utiliser un style naïf et simpliste. Il s'agit d'une suggestion pour essayer d'éviter, autant que possible, phrases subordonnées complexes qui rendent le texte difficile à comprendre pour les locuteurs non natifs de l'anglais.

Varité de l'anglais

Les variétés les plus répandues de l'anglais sont la britannique et l'américain, donc il est très probable que la plupart des auteurs utilisent l'un ou l'autre. Dans un environnement collaboratif, la variété idéale serait l'anglais international, mais il est très difficile, pour ne pas dire impossible, de se pronon-

cer sur quelle variété parmi toutes celles qui existent doit être la meilleure à utiliser.

Nous croyons que les différentes variétés peuvent se mélanger sans créer incompréhensions, mais en termes généraux, vous devriez essayer d'être cohérent et avant de décider entre britannique, américain ou toute autre variété anglaise votre discrétion, s'il vous plaît jeter un œil à la façon dont d'autres gens écrivent et essayer de les imiter.

Soyez équilibré

Ne soyez pas partial. Évitez d'inclure des références à des idéologies totalement étrangères à live-manual. L'écriture technique doit être aussi neutre que possible. Il est dans la nature même de l'écriture scientifique.

Soyez politiquement correct

Essayez d'éviter un langage sexiste autant que possible. Si vous avez besoin de faire référence à la troisième personne du singulier, de préférence utilisez they plutôt que he ou she ou inventions maladroites telles que s/he, s(he), etc.

Soyez concis

Allez droit au but et ne pas errer sans but. Donner autant d'informations que nécessaire, mais ne donnez pas plus d'informations que nécessaire, c'est-à-dire, ne pas expliquer les détails inutiles. Vos lecteurs sont intelligents. Présumez une certaine connaissance préalable de leur part.

Minimiser le travail de traduction

Gardez l'esprit que ce que vous écrivez devra être traduit dans plusieurs autres langues. Cela implique qu'un certain nombre de gens devront faire un travail supplémentaire si vous ajoutez des informations inutiles ou redondantes.

Soyez cohérent

Comme sugg  r   pr  c  demment, il est presque impossible de normaliser un document collaboratif dans un ensemble parfaitement unifi  . Cependant, tous les efforts de votre c  t   pour cri  re d  une man  re coh  rente avec le reste des auteurs seront appr  cis  s.

Soyez coh  sive

Utilisez autant de dispositifs de formation de texte que n  cessaire pour rendre votre texte coh  sive et sans ambigu  t  s. (Les dispositifs de formation de texte sont des marqueurs linguistiques tels que les connecteurs).

Soyez descriptif

Il est pr  f  rable de d  crire le point en une ou plusieurs paragraphes que simplement en utilisant un certain nombre de phrases dans un style typique changelog. D  crivez-le! Vos lecteurs appr  cieront a  .

Dictionnaire

Cherchez le sens des mots dans un dictionnaire ou une encyclop  die si vous ne savez pas comment exprimer certaines notions en anglais. Mais gardez l  esprit qu  un dictionnaire peut   tre votre meilleur ami ou peut se transformer en votre pire ennemi si vous ne savez pas comment l  utiliser correctement.

L  anglais poss  de le plus grand vocabulaire qui existe (avec plus d  un million de mots). Beaucoup de ces mots sont des emprunts d  autres langues. Lorsque l  on regarde le sens des mots dans un dictionnaire bilingue, la tendance d  un locuteur non natif de l  anglais est de choisir celui qui sonne plus similaire dans leur langue maternelle. Cela se transforme souvent en un discours excessivement formelle qui ne semble pas tout   fait naturel en anglais.

En r  gle g  n  rale, si un concept peut   tre exprim   en utilisant diff  rents synonymes, c  est un bon conseil choisir le premier mot propos   par le dictionnaire. En cas de doute, le choix des mots d  origine germanique (Habituellement mots monosyllabiques) est souvent la bonne chose   faire.

Il faut savoir que ces deux techniques peuvent produire un discours plut  t informel, mais au moins votre choix des mots sera d  utilisation grand et g  n  ralement accept  .

L  utilisation d  un dictionnaire de collocations est recommand  e. Ils sont extr  mement utiles quand il s  agit de savoir quels mots surviennent g  n  ralement ensemble.

Encore une fois, c  est une bonne pratique apprendre    partir du travail des autres. Gr  ce    un moteur de recherche pour v  rifier comment les autres auteurs utilisent certaines expressions peut aider beaucoup.

Faux amis, expressions idiomatiques et autres expressions

Attention aux faux amis. Peu importe comment vous tes comptent dans une langue trang  re, vous ne pouvez pas vous emp  cher de tomber de temps en temps dans le pige de ce qu  on appelle les faux amis, des mots qui se ressemblent dans les deux langues, mais dont les significations ou les utilisations pourrait   tre compl  tement diff  rent.

Essayez d  viter les expressions idiomatiques autant que possible. Les expressions idiomatiques sont des expressions qui peuvent transmettre un sens compl  tement diff  rent de ce que leurs mots individuels semblent signifier. Parfois, les expressions idiomatiques peuvent   tre difficiles   comprendre, m  me pour les locuteurs natifs de l  anglais!

vitez l  argot, les abr  viations, les contractions...

M  me si vous tes encourag  s   utiliser un langage simple, l  anglais de tous les jours, la r  daction technique appartient au registre formel de la langue.

Essayez d  viter l  argot, abr  viations inhabituels qui sont difficiles   comprendre et surtout les contractions qui tentent d  imiter le langage parl  . Sans oublier typique expressions d  irc et favorables   la famille.

17.1.2 Procdures

Tester avant d'crire

Il est important que les auteurs testent leurs exemples avant de les ajouter live-manual pour s'assurer que tout fonctionne comme dcrit. Tester sur un chroot propre ou une machine virtuelle peut tre un bon point de dpart. Par ailleurs, il serait idal si les tests ont ensuite t effectus sur des machines diffrentes avec un matriel diffrent pour reprer d'eventuels problmes qui pourraient survenir.

Exemples

En fournissant un exemple essayer d'tre aussi prcis que possible. Un exemple est, aprs tout, juste un exemple.

Il est souvent prfrable d'utiliser une ligne qui ne s'applique qu' un cas particulier que l'utilisation d'abstractions qui peuvent confondre vos lecteurs. Dans ce cas, vous pouvez fournir une brve explication des effets de l'exemple propos.

Il peut y avoir des exceptions lorsque l'exemple suggre d'utiliser certaines commandes potentiellement dangereuses qui, si elles sont mal utilises, peuvent causer des pertes de donnees ou d'autres effets indsirables similaires. Dans ce cas, vous devez fournir une explication approfondie des effets secondaires possibles.

Liens externes

Les liens externes ne doivent tre utilis lors que l'information sur ces sites est cruciale quand il s'agit de comprendre un point particulier. Mme si, essayez d'utiliser des liens externes aussi peu que possible. Les liens internet sont susceptibles de changer de temps en temps rsultant en des liens briss et en laissant vos arguments dans un tat incomplet.

D'ailleurs, les gens qui lisent le manuel hors ligne n'auront pas la chance de suivre ces liens.

vitez les marques et les choses qui violent la licence sous laquelle le manuel est publi

Essayez d'viter les marques autant que possible. Gardez l'esprit que d'autres projets peuvent utiliser la documentation que vous crivez. Donc, vous compliquez les choses pour eux si vous ajoutez certaines matires spcifiques.

live-manual est sous la licence GNU GPL. Cela a un certain nombre de consequences qui s'appliquent la distribution de la matire (de toute nature, y compris les graphiques ou logos protges) qui est publie avec le manuel.

Ecrire un premier projet, rviser, modifier, amliorer, refaire si ncessaire

- Remue-mnages!. Vous devez organiser vos ides d'abord dans une squence logique des vnements.

- Une fois que vous avez en quelque sorte organis ces ides dans votre esprit crire un premier projet.

- Rviser la grammaire, la syntaxe et l'orthographe. Gardez l'esprit que les noms propres des versions, tels que trixie ou sid , ne doivent pas tre capitaliss lorsqu'ils sont utilis comme noms de code. Pour vrifier l'orthographe, on peut excuter la cible spell. C'est dire, make spell

- Amliorer vos phrases et refaire n'importe quelle partie si ncessaire.

Chapitres

Utilisez le systme de numrotation classique des chapitres et sous-titres. Par exemple 1, 1.1, 1.1.1, 1.1.2 ... 1.2, 1.2.1, 1.2.2 ... 2, 2.1 ... et cetera. Voir marqueurs ci-dessous.

Si vous avez d'numrer une srie d'tapes ou phases dans votre description, vous pouvez galement utiliser des nombres ordinaux: premier, deuxime, troisme ... ou d'abord, puis, aprs cela, enfin ... Sinon, vous pouvez utiliser les lments puces.

Balisage

And last but not least, live-manual uses [SiSU](#) to process the text files and produce a multiple format output. It is recommended to take a look at [SiSU's manual](#) to get familiar with its markup, or else type:

```
$ sisu --help markup
```

Voici quelques exemples de balisage qui peuvent ventuellement vous aider:

- Pour accent/texte en gras:

```
*-foo"* or !-foo"! 
```

il produit: foo or foo . Utiliser pour mettre l'accent sur certains mots-clés.

- Pour italique:

```
/-foo"/ 
```

il produit: foo. Utiliser par exemple, pour les noms des paquets Debian.

- Pour monospace:

```
#-foo"# 
```

il produit: foo. Utiliser par exemple, pour les noms des commandes. Et aussi pour souligner certains mots clés ou des choses comme les chemins.

- Pour les blocs de code:

```
code-
$ foo
# bar
"code
```

il produit:

```
$ foo
# bar
```

Utilisez code- pour ouvrir et "code pour fermer les balises. Il est important de se rappeler de laisser un espace au dbut de chaque ligne de code.

17.2 Lignes directrices pour les traducteurs

Cette section traite des considérations générales à prendre en compte lors de la traduction du contenu du live-manual.

Comme recommandation générale, les traducteurs doivent avoir lu et compris les règles de traduction qui s'appliquent à leurs langues spécifiques. Habituellement, les groupes de traduction et listes de diffusion fournissent des informations sur la façon de produire un travail de traduction qui respecte les normes de qualité de Debian.

Remarque: Les traducteurs doivent aussi lire [Contribuer ce document](#). En particulier, la section [Traduction](#)

17.2.1 Conseils de traduction

Commentaires

Le rôle du traducteur est de transmettre le plus fidèlement possible le sens des mots, des phrases, des paragraphes et des textes comme crit par les auteurs originaux dans leur langue cible.

Donc, ils devraient s'abstenir d'ajouter des commentaires personnels ou des morceaux d'informations supplémentaires de leur propre chef. S'ils veulent ajouter un commentaire pour d'autres traducteurs travaillant sur les mêmes documents, ils peuvent le laisser dans l'espace réservé pour cela. Autrement dit, l'entête des chaînes dans les fichiers po précède d'un signe #. Nombreux logiciels de traduction graphiques peuvent gérer automatiquement ces types de commentaires.

NDT, Note Du Traducteur

Il est parfaitement acceptable cependant, inclure un mot ou une expression entre parenthèses dans le texte traduit si, et seulement si, cela fait le sens d'un mot ou d'une expression difficile plus clair pour le lecteur. À l'intérieur des parenthèses, le traducteur doit mettre en évidence que l'addition a été leur en utilisant l'abréviation NDT ou Note Du Traducteur.

Phrases impersonnelles

Les documents écrits en anglais font une utilisation intensive de la forme impersonnelle you. Dans d'autres langues qui ne partagent pas cette caractéristique, ce pourrait donner la fausse impression que les textes originaux traitent directement le lecteur quand en réalité ils ne le font pas. Les traducteurs doivent être conscients de ce fait et traduire dans leur langue le plus fidèlement que possible.

Faux amis

Le piège des faux amis explique avant s'applique surtout aux traducteurs.

Vérifiez le sens des faux amis suspects en cas de doute.

Balisage

Les traducteurs qui travaillent d'abord avec les fichiers pot et plus tard avec les fichiers po trouveront de nombreuses caractéristiques de balisage dans les chaînes. Ils peuvent traduire le texte de toute façon, tant qu'il est traduisible, mais il est extrêmement important qu'ils utilisent exactement le même balisage que la version originale anglaise.

Blocs de code

Même si les blocs de code sont généralement intraduisibles, les inclure dans la traduction est la seule façon d'obtenir une traduction complète 100%. Et même si cela signifie plus de travail au début car cela peut exiger l'intervention des traducteurs si le code change, il est le meilleur moyen, long terme, d'identifier ce qui a déjà été traduit et ce qui n'a pas, lors de la vérification de l'intégrité des fichiers .po.

Sauts de ligne

Les textes traduits doivent avoir les mêmes sauts de lignes exactes que les textes originaux. Veuillez appuyer sur Entrée ou tapez si elles apparaissent dans les fichiers originaux. Ces nouvelles lignes apparaissent souvent, par exemple, dans les blocs de code.

Ne vous trompez pas, cela ne signifie pas que le texte traduit doit avoir la même longueur que la version anglaise. C'est presque impossible.

Chaînes intraduisibles

Les traducteurs ne doivent jamais traduire:

- Les noms de code des versions (qui doivent être écrits en minuscules)
- Les noms des logiciels
- Les commandes fournies titre d'exemples
- Métadonnées (souvent entre deux points :metadata:)

999 - Liens
1000 - Les chemins

SiSU Metadata, document information

Titre: Debian Live Manual

Auteur: Debian Live Project jdebian-live@lists.debian.org

Droits relatifs à la ressource: Copyright: Copyright (C) 2006-2015 Live Systems Project,
Copyright (C) 2016-2025 The Debian Live team

License: This program is free software: you can redistribute it and/or modify it under the
terms of the GNU General Public License as published by the Free Software Foundation,
either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this
program. If not, see <http://www.gnu.org/licenses/>.

The complete text of the GNU General Public License can be found in `/usr/share/-
common-licenses/GPL-3` file.

diteur: Debian Live Project jdebian-live@lists.debian.org

Date: 2025-02-26

Version Information

Fichier source: live-manual.ssm.sst

Filetype: SiSU text 2.0, Unicode text, UTF-8 text, with very long lines (745)

Source Digest: SHA2-256(live-manual.ssm.sst)=81298066d42099e6b145c4307ea37368-
0b036a41e1c1017850fbd4fa84f395ee

Generated

Dernière production du document (metaverse): 2025-02-26 23:59:08 +0000

Gnr par: SiSU 7.3.0 of 2023w44/1 (2023-10-30)

Version de Ruby: ruby 3.3.7 (2025-01-15 revision be31f993d7) [x86_64-linux-gnu]