Linux From Scratch

Version 10.1-systemd

Publié le 01 mars 2021

Créé par Gerard Beekmans
Rédacteur en chef : Bruce Dubbs
Éditeur : Douglas R. Reno
Éditeur : DJ Lucas

Linux From Scratch: Version 10.1-systemd: Publié le 01 mars 2021

par Créé par Gerard Beekmans, Rédacteur en chef : Bruce Dubbs, Éditeur : Douglas R. Reno, et Éditeur : DJ Lucas

Copyright © 1999-2021 Gerard Beekmans

Copyright © 1999-2021, Gerard Beekmans

Tous droits réservés.

Ce livre est distribué sous la Creative Commons License.

Les instructions d'ordinateur peuvent être extraites du livre sous la MIT License.

Linux® est une marque déposée de Linus Torvalds.

Table des matières

Préface vii
i. Avant-propos vii
ii. Public visé viii
iii. Architectures cibles de LFS viii
iv. Prérequis ix
v. LFS et les standards ix
vi. Raison de la présence des paquets dans le livre xi
vii. Typographie xvi
viii. Structure xvii
ix. Errata xviii
I. Introduction 1
1. Introduction 2
1.1. Comment construire un système LFS 2
1.2. Quoi de neuf depuis la dernière version 2
1.3. Historique des modifications 4
1.4. Ressources 8
1.5. Aide 8
II. Préparation à la construction 11
2. Préparation du système hôte 12
2.1. Introduction 12
2.2. Prérequis du système hôte 12
2.3. Les étapes de la construction de LFS 14
2.4. Créer une nouvelle partition 15
2.5. Créer un système de fichiers sur la partition 17
2.6. Définir la variable $LFS 18
2.7. Monter la nouvelle partition 18
3. Paquets et correctifs 20
3.1. Introduction 20
3.2. Tous les paquets 20
3.3. Correctifs requis 28
4. Dernières préparations 29
4.1. Introduction 29
4.2. Créer un ensemble limité de répertoire dans le système de fichiers LFS 29
4.3. Ajouter l'utilisateur LFS 29
4.4. Configurer l'environnement 30
4.5. À propos des SBU 32
4.6. À propos des suites de tests 33
III. Construction des outils croisés LFS et des outils temporaires 34
Informations préliminaires importantes xxxv
i. Introduction xxxv
ii. Notes techniques sur la chaîne d'outils xxxv
iii. Instructions générales de compilation xxxix
5. Compilation d'une chaîne d'outils croisée 41
5.1. Introduction 41
5.2. Binutils-2.36.1 — Passe 1 42
5.3. GCC-10.2.0 — Passe 1 44
5.4. Linux-5.10.17 API Headers 47
5.5. Glibc-2.33 48

5.6. Libstdc++ de GCC-10.2.0, passe 1 51
6. Compilation croisée des outils temporaires 52
6.1. Introduction 52
6.2. M4-1.4.18 53
6.3. Ncurses-6.2 54
6.4. Bash-5.1 56
6.5. Coreutils-8.32 57
6.6. Diffutils-3.7 58
6.7. File-5.39 59
6.8. Findutils-4.8.0 60
6.9. Gawk-5.1.0 61
6.10. Grep-3.6 62
6.11. Gzip-1.10 63
6.12. Make-4.3 64
6.13. Patch-2.7.6 65
6.14. Sed-4.8 66
6.15. Tar-1.34 67
6.16. Xz-5.2.5 68
6.17. Binutils-2.36.1 — Passe 2 69
6.18. GCC-10.2.0 — Passe 2 70
7. Entrée dans le chroot et construction des outils temporaires supplémentaires 72
7.1. Introduction 72
7.2. Changer de propriétaire 72
7.3. Préparer les systèmes de fichiers virtuels du noyau 72
7.4. Entrer dans l'environnement chroot 73
7.5. Créer les répertoires 74
7.6. Créer les fichiers et les liens symboliques essentiels 75
7.7. Libstdc++ de GCC-10.2.0, Passe 2 79
7.8. Gettext-0.21 80
7.9. Bison-3.7.5 81
7.10. Perl-5.32.1 82
7.11. Python-3.9.2 83
7.12. Texinfo-6.7 84
7.13. Util-linux-2.36.2 85
7.14. Nettoyage et sauvegarde du système temporaire 86
IV. Construction du système LFS 88
8. Installer les logiciels du système de base 89
8.1. Introduction 89
8.2. Gestion de paquets 89
8.3. Man-pages-5.10 93
8.4. Iana-Etc-20210202 94
8.5. Glibc-2.33 95
8.6. Zlib-1.2.11 102
8.7. Bzip2-1.0.8 103
8.8. Xz-5.2.5 105
8.9. Zstd-1.4.8 107
8.10. File-5.39 108
8.11. Readline-8.1 109
8.12. M4-1.4.18 111
8.13. Bc-3.3.0 112

8.14. Flex-2.6.4 113
8.15. Tcl-8.6.11 114
8.16. Expect-5.45.4 116
8.17. DejaGNU-1.6.2 117
8.18. Binutils-2.36.1 118
8.19. GMP-6.2.1 121
8.20. MPFR-4.1.0 123
8.21. MPC-1.2.1 124
8.22. Attr-2.4.48 125
8.23. Acl-2.2.53 126
8.24. Libcap-2.48 127
8.25. Shadow-4.8.1 128
8.26. GCC-10.2.0 132
8.27. Pkg-config-0.29.2 137
8.28. Ncurses-6.2 138
8.29. Sed-4.8 141
8.30. Psmisc-23.4 142
8.31. Gettext-0.21 143
8.32. Bison-3.7.5 145
8.33. Grep-3.6 146
8.34. Bash-5.1 147
8.35. Libtool-2.4.6 149
8.36. GDBM-1.19 150
8.37. Gperf-3.1 151
8.38. Expat-2.2.10 152
8.39. Inetutils-2.0 153
8.40. Perl-5.32.1 155
8.41. XML::Parser-2.46 158
8.42. Intltool-0.51.0 159
8.43. Autoconf-2.71 160
8.44. Automake-1.16.3 161
8.45. Kmod-28 162
8.46. Libelf de Elfutils-0.183 164
8.47. Libffi-3.3 165
8.48. OpenSSL-1.1.1j 166
8.49. Python-3.9.2 168
8.50. Ninja-1.10.2 170
8.51. Meson-0.57.1 171
8.52. Coreutils-8.32 172
8.53. Check-0.15.2 177
8.54. Diffutils-3.7 178
8.55. Gawk-5.1.0 179
8.56. Findutils-4.8.0 180
8.57. Groff-1.22.4 181
8.58. GRUB-2.04 184
8.59. Less-563 186
8.60. Gzip-1.10 187
8.61. IPRoute2-5.10.0 189
8.62. Kbd-2.4.0 191
8.63. Libpipeline-1.5.3 193

8.64. Make-4.3 194
8.65. Patch-2.7.6 195
8.66. Man-DB-2.9.4 196
8.67. Tar-1.34 199
8.68. Texinfo-6.7 200
8.69. Vim-8.2.2433 202
8.70. Systemd-247 205
8.71. D-Bus-1.12.20 211
8.72. Procps-3.3.17 213
8.73. Util-linux-2.36.2 215
8.74. E2fsprogs-1.46.1 220
8.75. À propos des symboles de débogage 223
8.76. Supprimer de nouveau les symboles des fichiers objets 223
8.77. Nettoyer 224
9. Configuration du système 226
9.1. Introduction 226
9.2. Configuration générale du réseau 226
9.3. Manipulation des périphériques et modules 229
9.4. Gérer les périphériques 233
9.5. Configuration de l'horloge système 233
9.6. Configurer la Console Linux 234
9.7. Configuration des paramètres régionaux du système 236
9.8. Créer le fichier /etc/inputrc 237
9.9. Créaction du fichier /etc/shells 239
9.10. Utilisation et configuration de Systemd 239
10. Rendre le système LFS amorçable 243
10.1. Introduction 243
10.2. Créer le fichier /etc/fstab 243
10.3. Linux-5.10.17 245
10.4. Utiliser GRUB pour paramétrer le processus de démarrage 250
11. La fin 253
11.1. Fin 253
11.2. Enregistrez-vous 253
11.3. Redémarrer le système 253
11.4. Et maintenant ? 254
V. Annexes 256
A. Acronymes et Termes 257
B. Remerciements 259
C. Dépendances 262
D. Licences LFS 276
D.1. Creative Commons License 276
D.2. The MIT License 280
Index 281

Préface

Avant-propos

Mon parcours pour apprendre et mieux comprendre Linux a débuté en 1998. Je venais d'installer ma première distribution Linux et je fus rapidement intrigué par l'ensemble du concept et la philosophie sous-jacente de Linux.

Il y a toujours bien des manières d'accomplir une seule tâche. Il en est de même pour les distributions Linux. Un grand nombre existent depuis des années. Certaines existent encore, certaines se sont transformées en quelque chose d'autre, tandis que d'autres encore ont été reléguées à nos souvenirs. Elles font toutes les choses différemment pour s'adapter au besoin de leur public. Vu qu'il existait énormément de manières différentes d'atteindre le même objectif, je me rendis compte que je n'étais plus obligé de me limiter à une organisation en particulier. Avant de découvrir Linux, on supportait simplement les problèmes dans d'autres systèmes d'exploitation puisqu'on n'avait pas le choix. Cela valait ce que ça valait, que cela nous plaise ou non. Avec Linux, le concept du choix a commencé à émerger. Si vous n'aimiez pas quelque chose, vous étiez libres, voire encouragés à le modifier.

J'ai essayé un certain nombre de distributions et n'ai pas pu me décider pour l'une d'entre elles. C'étaient de bons systèmes, chacun à sa façon. Ce n'était plus une question de bonne ou mauvaise qualité. C'était devenu une question de goût personnel. Avec tout ce choix disponible, il devenait clair qu'il n'y aurait pas un seul système qui serait parfait pour moi. Donc je résolus de créer mon propre système Linux qui correspondrait totalement à mes préférences personnelles.

Pour que ce soit vraiment mon propre système je résolus de compiler tout à partir du code source au lieu d'utiliser des paquets de binaires pré-compilés. Ce système Linux « parfait » aurait les forces de plusieurs systèmes sans leurs faiblesses ressenties. De prime abord, l'idée était décourageante. Je restais sceptique à la pensée de pouvoir construire un tel système.

Après avoir rencontré quelques problèmes comme des dépendances circulaires et des erreurs à la compilation, j'ai finalement construit un système Linux entièrement personnalisé. Il était totalement opérationnel et parfaitement utilisable comme n'importe quel autre système Linux du moment. Mais c'était ma propre création. C'était très satisfaisant d'avoir concocté un tel système moi-même. Créer chaque morceau de logiciel par moi-même aurait été la seule option encore plus satisfaisante. C'était là la meilleure alternative.

Alors que je partageais mes objectifs et mes expériences avec d'autres membres de la communauté Linux, il devint manifeste qu'il y avait un intérêt soutenu concernant ces idées. Il devint rapidement clair que de tels systèmes Linux personnalisés satisfaisaient non seulement les exigences des utilisateurs mais servaient aussi d'une opportunité idéale d'apprentissage pour les programmeurs et les administrateurs systèmes, afin d'améliorer leurs compétences (existantes) sous Linux. De cet intérêt est né le projet *Linux From Scratch*..

Ce livre *Linux From Scratch* est le cœur de ce projet. Il fournit la base et les instructions qui vous sont nécessaires pour concevoir et construire votre propre système. Bien que ce livre fournisse un modèle qui aboutira à un système qui fonctionne correctement, vous êtes libres de modifier les instructions pour les adapter à vos envies, ce qui fait partie des finalités importantes du projet après tout. Vous gardez le contrôle ; nous vous donnons simplement un coup de main pour débuter votre propre parcours.

J'espère sincèrement que vous passerez un bon moment en travaillant sur votre propre système *Linux From Scratch* et que vous apprécierez les nombreux bénéfices qu'apporte un système qui est réellement le vôtre.

--
Gerard Beekmans
gerard@linuxfromscratch.org

Public visé

Beaucoup de raisons peuvent vous pousser à vouloir lire ce livre. Une des questions que beaucoup de monde se pose est « pourquoi se fatiguer à construire à la main un système Linux de A à Z alors qu'il suffit de télécharger et installer une distribution existante ? »

Une raison importante de l'existence de ce projet est de vous aider à apprendre comment fonctionne un système Linux de l'intérieur. Construire un système LFS aide à démontrer ce qui fait que Linux fonctionne, et comment les choses interagissent et dépendent les unes des autres. Une des meilleures choses que l'expérience de cet apprentissage peut vous apporter est la capacité à personnaliser un système Linux afin qu'il soit à votre goût et réponde à vos besoins.

Un autre avantage clé de LFS est qu'il vous permet d'avoir plus de contrôle sur votre système sans avoir à dépendre d'une implémentation créée par quelqu'un d'autre. Avec LFS, vous êtes maintenant au volant et vous êtes capable de décider chaque aspect du système.

LFS vous permet de créer des systèmes Linux très compacts. Lors de l'installation d'une distribution habituelle, vous êtes souvent obligé d'installer de nombreux programmes que vous n'utiliserez ni ne comprendrez probablement jamais. Ces programmes gaspillent des ressources. Vous pourriez répondre qu'avec les disques durs et les processeurs d'aujourd'hui, les ressources ne sont plus un problème. Pourtant, vous êtes parfois contraint par des questions d'espace, ou d'autres limitations. Pensez aux CD, clés USB amorçables et aux systèmes embarqués. Ce sont des domaines où LFS peut être avantageux.

Un autre avantage d'un système Linux personnalisé est un surcroît de sécurité. En compilant le système complet à partir du code source, vous avez la possibilité de tout vérifier et d'appliquer tous les correctifs de sécurité désirés. Il n'est plus nécessaire d'attendre que quelqu'un d'autre vous fournisse les paquets binaires réparant une faille de sécurité. À moins que vous n'examiniez vous-même le correctif et que vous ne l'appliquiez vous-même, vous n'avez aucune garantie que le nouveau paquet binaire ait été compilé correctement et qu'il corrige bien le problème.

Le but de *Linux From Scratch* est de construire les fondations d'un système complet et utilisable. Si vous ne souhaitez pas construire votre propre système à partir de rien, vous pourriez cependant bénéficier des informations contenues dans ce livre.

Il existe trop de bonnes raisons de construire votre système LFS pour pouvoir toutes les lister ici. En fin de compte, l'apprentissage est de loin la raison la plus puissante. En continuant dans votre expérience de LFS, vous trouverez la puissance réelle que donnent l'information et la connaissance.

Architectures cibles de LFS

Les architectures cibles primaires de LFS sont les processeurs AMD et Intel x86 (32 bits) et x86_64 (64 bits). En même temps les instructions de ce livre sont connues pour fonctionner également, avec quelques modifications, sur les processeurs Power PC et ARM. Pour construire un système qui utilise un de ces processeurs, le prérequis principal supplémentaire à ceux des pages suivantes est la présence d'un système comme une LFS précédemment installée, Ubuntu, Red Hat/Fedora, SuSE, ou une autre distribution ciblant l'architecture que vous avez. Remarquez aussi que vous pouvez installer et utiliser un système 32 bits en tant que système hôte sur un système AMD ou Intel 64 bits.

Pour la construction de LFS, le gain obtenu en compilant sur un système 64 bits comparé à un système 32 bits est minimal. Par exemple, dans le test de la construction de LFS-9.1 sur un système basé sur un processeur Core i7-4790, nous avons relevé les statistiques suivantes :

```
Temps de construction de l'architecture     Taille de la construction
32 bit       239,9 minutes  3,6 Go
64 bit       233,2 minutes  4,4 Go
```

Comme vous pouvez le constater, sur le même matériel, la construction 64 bits est seulement 3 % plus rapide et est 22 % plus grosse que la construction en 32 bits. Si vous voulez utiliser LFS pour un serveur LAMP, ou un pare-feu, un CPU 32 bits peut être largement suffisant. Au contraire, plusieurs paquets dans BLFS ont maintenant besoin de plus de 4 Go de RAM pour être construits ou lancés, si bien que si vous voulez utiliser LFS sur un ordinateur de bureau, les auteurs de LFS recommandent de construire un système 64 bits.

La construction 64 bits par défaut qui résulte de LFS est considérée comme un système « pur » 64 bits. C'est-à-dire qu'elle ne prend en charge que les exécutables en 64 bits. La construction d'un système « multi-lib » implique la construction de beaucoup d'applications à deux reprises, une fois pour le système 32 bits et une fois pour le système 64 bits. Ceci n'est pas pris en charge par LFS car cela interférerait avec l'objectif pédagogique visant à fournir les instructions nécessaires à un simple système Linux de base. Certains éditeurs de LFS et BLFS maintiennent un fork de LFS pour le multilib, accessible sur *http://www.linuxfromscratch.org/~thomas/multilib/index.html*. Mais c'est un sujet avancé.

Prérequis

Construire un système LFS n'est pas une tâche facile. Cela requiert un certain niveau de connaissance en administration de système Unix pour résoudre les problèmes et exécuter correctement les commandes listées. En particulier, au strict minimum, vous devriez déjà savoir comment utiliser la ligne de commande (le shell) pour copier et déplacer des fichiers et des répertoires, pour lister le contenu de répertoires et de fichiers, et pour changer de répertoire. Il est aussi attendu que vous disposiez d'une connaissance raisonnable de l'utilisation et de l'installation de logiciels Linux.

Comme le livre LFS attend *au moins* ce simple niveau de connaissance, les différents forums d'aide de LFS seront peu capables de vous fournir une assistance en dessous de ce niveau. Vous finirez par remarquer que vos questions n'auront pas de réponses ou que vous serez renvoyé à la liste des lectures principales avant l'installation.

Avant de construire un système LFS, nous recommandons de lire les guides pratiques suivants :

- Software-Building-HOWTO *http://www.tldp.org/HOWTO/Software-Building-HOWTO.html*

  C'est un guide complet sur la construction et l'installation « générique » de logiciels Unix sous Linux. Bien qu'il ait été écrit il y a longtemps, il offre encore un bon résumé des techniques de base requises pour construire et installer un logiciel.

- Beginner's Guide to Installing from Source (Guide de l’installation à partir des sources pour le débutant) *http://moi.vonos.net/linux/beginners-installing-from-source/*

  Ce guide propose un bon résumé des compétences et des connaissances techniques de base nécessaires à la construction de logiciels à partir du code source.

LFS et les standards

La structure de LFS suit les standards Linux aussi fidèlement que possible. Les principaux standards sont :

- *POSIX.1-2008.*.
- *Filesystem Hierarchy Standard (FHS) version 3.0*
- *Linux Standard Base (LSB) version 5.0 (2015)*

  La LSB comporte quatre standards séparés : le cœur, le bureau, les langages à l'exécution et l'impression. Outre les exigences génériques, il y a aussi des exigences spécifiques à l'architecture. Il y a aussi deux domaines pour l'évaluation : Gtk3 et les graphismes. LFS s'efforce de respecter l'architecture évoquée dans la section précédente.

Note

Beaucoup de gens ne sont pas d'accord avec les exigences de la LSB. L'objectif principal de leur existence est de garantir que les logiciels propriétaires pourront être installés et lancés correctement sur un système conforme. Comme LFS est basée sur le code source, l'utilisateur a un contrôle complet des paquets qu'il désire et beaucoup choisissent de ne pas installer certains paquets qui sont spécifiés dans la LSB.

La création d'un système complet capable de réussir les tests de certificats LSB est possible, mais non sans quelques paquets supplémentaires qui vont au-delà des objectifs de LFS. Ces paquets supplémentaires ont des instructions d'installation dans BLFS.

Paquets fournis par LFS requis pour satisfaire les exigences de la LSB

Cœur de la LSB :	Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux, Zlib
LSB pour les bureaux :	Aucun
LSB pour les langages à l'exécution :	Perl
LSB pour l'impression :	Aucun
LSB pour Gtk3 et LSB pour les graphismes (évaluation):	Aucun

Paquets fournis par BLFS requis pour satisfaire les exigences de la LSB

Cœur de la LSB :	At, Batch (partie d'At), Cpio, Ed, Fcrontab, LSB-Tools, NSPR, NSS, PAM, Pax, Sendmail (ou Postfix ou Exim), time
LSB pour les bureaux :	Alsa, ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Gdk-pixbuf, Glib2, GTK+2, Icon-naming-utils, Libjpeg-turbo, Libpng, Libtiff, Libxml2, MesaLib, Pango, Xdg-utils, Xorg
LSB pour les langages à l'exécution :	Python, Libxml2, Libxslt
LSB pour l'impression :	CUPS, Cups-filters, Ghostscript, SANE
LSB pour Gtk3 et LSB pour les graphismes (évaluation):	GTK+3

Paquets fournis ni par LFS ni par BLFS mais qui sont requis pour satisfaire les exigences de la LSB

Cœur de la LSB :	Aucun
LSB pour les bureaux :	Qt4 (mais Qt5 est disponible)
LSB pour les langages à l'exécution :	Aucun
LSB pour l'impression :	Aucun
LSB pour Gtk3 et LSB pour les graphismes (évaluation):	Aucun

Raison de la présence des paquets dans le livre

Comme indiqué plus haut, le but de LFS est de construire les fondations complètes et utilisables d'un système. Il inclut tous les paquets nécessaires pour être répliqué tout en fournissant une base relativement minimale vous permettant de personnaliser un système plus complet basé sur les choix de l'utilisateur. Cela ne veut pas dire que LFS est le plus petit système possible. Plusieurs paquets importants sont inclus et ne sont pas absolument indispensables. Les listes ci-dessous documentent la raison pour laquelle chaque paquet se trouve dans le livre.

- Acl

  Ce paquet contient des outils d'administration des listes de contrôle d'accès, utilisées pour définir plus finement les droits d'accès de votre choix pour les fichiers et les répertoires.

- Attr

  Ce paquet contient des programmes d'administration des attributs étendus sur les objets d'un système de fichiers.

- Autoconf

  Le paquet Autoconf contient des programmes produisant des scripts shell qui configurent automatiquement le code source à partir du modèle fourni par le développeur. Il est souvent requis pour reconstruire un paquet après une mise à jour des procédures de construction.

- Automake

  Ce paquet contient des programmes pour générer des Makefile à partir d'un modèle. Il est souvent requis pour reconstruire un paquet après une mise à jour des procédures de construction.

- Bash

  Ce paquet satisfait une exigence du cœur de la LSB pour fournir une interface Bourne Shell au système. Il a été choisi parmi d'autres shells du fait de son utilisation répandue et de ses fonctionnalités étendues au-delà des fonctions d'un shell de base.

- Bc

  Ce paquet fournit un langage de traitement numérique à précision arbitraire. Il satisfait une exigence utilisée pour la construction du noyau Linux.

- Binutils

  Ce paquet contient un éditeur de liens, un assembleur et d'autres outils de gestion des fichiers objets. Les programmes de ce paquet sont nécessaires pour compiler la plupart des paquets d'un système LFS et au-delà.

- Bison

  Ce paquet contient la version GNU de yacc (*Yet Another Compiler Compiler*, encore un nouveau compilateur de compilateur) requis pour construire plusieurs autres programmes de LFS.

- Bzip2

  Ce paquet contient des programmes de compression et de décompression de fichiers. Il est nécessaire pour décompresser plusieurs paquets de LFS.

- Check

  Ce paquet contient un harnais de tests pour d'autres programmes.

- Coreutils

  Ce paquet contient un certain nombre de paquets essentiels pour visualiser et manipuler des fichiers et des répertoires. Ces programmes sont nécessaires pour la gestion de fichiers en ligne de commande et ils sont nécessaires pour les procédures d'installation de chaque paquet de LFS.

- D-Bus

  Ce paquet contient des programmes pour implémenter un système de message par bus, qui est un moyen simple pour les applications de communiquer avec les autres.

- DejaGNU

  Ce paquet contient un harnais de tests pour d'autres programmes.

- Diffutils

  Ce paquet contient des programmes qui montrent les différences entre des fichiers ou entre des répertoires. On peut utiliser ces programmes pour créer des correctifs et ils sont aussi utilisés dans de nombreuses procédures de construction de paquets.

- E2fsprogs

  Ce paquet contient les outils de gestion des systèmes de fichiers ext2, ext3 et ext4. Ce sont les systèmes de fichiers les plus courants et les plus largement testés, parmi ceux pris en charges par Linux.

- Expat

  Ce paquet contient une bibliothèque d'analyse XML relativement petite. Il est exigé par le module Perl XML::Parser.

- Expect

  Ce paquet contient un programme pour réaliser des dialogues scriptés avec d'autres programmes interactifs. Il est souvent utilisé pour tester d'autres paquets. Il n'est installé que dans la chaîne d'outils temporaire.

- File

  Ce paquet contient un outil pour déterminer le type d'un ou plusieurs fichiers donnés. Quelques paquets en ont besoin dans leur script de construction.

- Findutils

  Ce paquet contient des programmes pour rechercher des fichiers sur un système de fichiers. Il est utilisé dans les scripts de construction de nombreux paquets.

- Flex

  Ce paquet contient un outil de génération de programmes qui reconnaît des modèles de texte. C'est la version GNU du programme lex (*lexical analyzer*, analyseur lexical). Il est nécessaire pour construire plusieurs paquets LFS.

- Gawk

  Ce paquet contient des programmes de manipulation de fichiers texte. C'est la version GNU du programme awk (Aho-Weinberg-Kernighan). Il est utilisé dans les scripts de construction de nombreux autres paquets.

- GCC

  Ce paquet est le *Gnu Compiler Collection*. Il contient les compilateurs C et C++ ainsi que d'autres qui ne sont pas construits dans LFS.

- GDBM

  Ce paquet contient la bibliothèque *GNU Database Manager* (gestionnaire de base de données GNU). Il est utilisé par un autre paquet de LFS : Man-DB.

- Gettext

  Ce paquet contient des outils et des bibliothèques pour l'internationalisation et la traduction de nombreux paquets.

- Glibc

  Ce paquet contient la bibliothèque C principale. Les programmes Linux ne peuvent pas s'exécuter sans elle.

- GMP

  Ce paquet contient des bibliothèques mathématiques qui fournissent des fonctions utiles pour de l'arithmétique en précision arbitraire. Il est nécessaire pour construire GCC.

- Gperf

  Ce paquet contient un programme qui génère une fonction de hachage parfaite à partir d'un trousseau. Il est exigé par Eudev.

- Grep

  Ce paquet contient des programmes de recherche au sein de fichiers. Ces programmes sont utilisés par la plupart des scripts de construction des paquets.

- Groff

  Le paquet Groff contient des programmes de formatage de texte. Une des fonctions importantes de ces programmes est le formatage des pages de manuels.

- GRUB

  Ce paquet est le chargeur *Grand Unified Boot*. Ce n'est pas le seul chargeur d'amorçage disponible, mais c'est le plus flexible.

- Gzip

  Ces paquets contiennent des programmes de compression et de décompression de fichiers. Il est nécessaire pour décompresser de nombreux paquets sur LFS et au-delà.

- Iana-etc

  Ce paquet fournit des données pour des services et des protocoles réseau. Il est nécessaire pour activer les bonnes fonctionnalités de réseau.

- Inetutils

  Ce paquet contient des programmes d'administration réseau de base.

- Intltool

  Ce paquet contient des outils pour extraire des chaînes traduisibles de fichiers sources.

- IProute2

  Ce paquet contient des programmes pour du réseau de base ou avancé en IPv4 et IPv6. Il a été choisi parmi les paquets d'outils réseau courants (net-tools) pour ses fonctionnalités IPv6.

- Kbd

  Ce paquet contient des fichiers de tables de touches, des outils claviers pour les claviers non américains et un certain nombre de polices pour console.

- Kmod

  Ce paquet contient des programmes nécessaires pour administrer les modules du noyau Linux.

- Less

  Ce paquet contient un très bon visualiseur de texte qui permet le défilement vers le haut ou vers le bas lors de la visualisation d'un fichier. Il est aussi utilisé par Man-DB pour visualiser des pages de manuels.

- Libcap

Ce paquet implémente les interfaces au niveau utilisateur avec les possibilités POSIX 1003.1e disponibles dans les noyaux Linux.

- Libelf

  Le projet elfutils fournit des bibliothèques et des outils pour les fichiers ELF et le format de données DWARF. La plupart des utilitaires de ce paquet sont disponibles dans d'autres paquets mais la bibliothèque est requise pour construire le noyau Linux avec la configuration par défaut (et la plus efficace).

- Libffi

  Ce paquet implémente une interface de programmation portable et haut-niveau pour diverses conventions d'appel. Certains programmes peuvent ne pas savoir à la compilation les arguments à passer à une fonction. Par exemple, il se peut qu'un interpréteur n'apprenne le nombre et le type des arguments utilisés pour appeler une fonction donnée qu'à l'exécution. Libffi peut être utilisée dans ces programmes pour fournir une passerelle entre l'interpréteur et le code compilé.

- Libpipeline

  Le paquet Libpipeline contient une bibliothèque pour manipuler des files (pipelines) de sous-processus de façon flexible et commode. Il est requis par le paquet Man-DB.

- Libtool

  Ce paquet contient le script générique de prise en charge des bibliothèques de GNU. Il englobe la complexité de l'utilisation des bibliothèques partagées dans une interface cohérente et portable. Il est exigé par les suites de tests d'autres paquets de LFS.

- Noyau Linux

  Ce paquet est le système d'exploitation. C'est Linux dans l'environnement GNU/Linux.

- M4

  Ce paquet contient un traitement de macros textuelles générales utile en tant qu'outil de construction d'autres programmes.

- Make

  Ce paquet contient un programme de gestion de la construction des paquets. Il est requis par presque tous les paquets de LFS.

- Man-DB

  Ce paquet contient des programmes de recherche et de visualisation de pages de manuels. Il a été préféré au paquet man du fait de fonctionnalités d'internationalisation supérieures. Il fournit le programme man.

- Man-pages

  Ce paquet contient le contenu final des pages de manuels de base de Linux.

- Meson

  Ce paquet fournit un outil logiciel pour automatiser la construction de logiciels. Le but principal de Meson est de minimiser le temps que les développeurs passent à configurer leur système de construction.

- MPC

  Ce paquet contient des fonctions pour le calcul de nombres complexes. Il est exigé par GCC.

- MPFR

  Ce paquet contient des fonctions pour des maths à précision multiple. Il est exigé par GCC.

- Ninja

Ce paquet contient un petit système de construction qui met l'accent sur la vitesse. Ses fichiers d'entrées sont prévus pour être générés par un système de construction de plus haut niveau, et il est prévu pour lancer des constructions aussi rapidement que possible.

- Ncurses

  Le paquet Ncurses contient les bibliothèques de gestion des écrans type caractère, indépendant des terminaux. Il est souvent utilisé pour fournir le contrôle du curseur dans un système en menus. Il est exigé par un certain nombre de paquets de LFS.

- Openssl

  Ce paquet fournit les outils de gestion et les bibliothèques liées à la cryptographie. Ils sont utiles pour fournir des fonctions de cryptographies à d'autres paquet, dont le noyau Linux.

- Patch

  Ce paquet contient un programme pour modifier ou créer des fichiers en appliquant un fichier de *correctif* créé en général par le programme diff. Il est requis par la procédure de construction de plusieurs paquets LFS.

- Perl

  Ce paquet est un interpréteur du langage PERL. Il est nécessaire pour l'installation et les suites de tests de plusieurs paquets LFS.

- Pkg-config

  Ce paquet fournit un programme qui renvoie les métadonnées d'une bibliothèque ou d'un binaire installé.

- Procps-NG

  Ce paquet contient des programmes de surveillance des processus. Ces programmes sont utiles pour l'administration système et ils sont aussi utilisés par les scripts de démarrage LFS.

- Psmisc

  Ce paquet contient des programmes d'affichage d'informations sur les processus en cours d'exécution. Ces programmes sont utiles pour l'administration système.

- Python 3

  Ce paquet fournit un langage interprété dont la philosophie de conception valorise la lisibilité du code.

- Readline

  Ce paquet est un ensemble de bibliothèques qui offrent des fonctionnalités d'édition et d'historique de la ligne de commande. Il est utilisé par Bash.

- Sed

  Ce paquet permet de saisir du texte sans ouvrir le fichier dans un éditeur de texte. Il est aussi requis par la plupart des scripts de configuration des paquets LFS.

- Shadow

  Ce paquet contient des programmes de gestion sécurisée des mots de passe.

- Systemd

  Ce paquet fournit un programme d'init et plusieurs possibilités supplémentaires de contrôle du démarrage et du système en tant qu'alternative à SysVinit. Il est utilisé dans de nombreuses distributions commerciales.

- Tar

  Ce paquet fournit des fonctionnalités d'archivage et d'extraction de potentiellement tous les paquets utilisés dans LFS.

- Tcl

  Ce paquet contient le *Tool Command Language* utilisé dans beaucoup de suites de tests des paquets LFS.

- Texinfo

  Ce paquet contient des programmes de lecture, d'écriture et de conversion de pages info. Il est utilisé dans les procédures d'installation de beaucoup de paquets LFS.

- Util-linux

  Ce paquet contient des programmes généraux. Parmi eux, se trouvent des outils de gestion des systèmes de fichiers, de consoles, de partitions et de messages.

- Vim

  Ce paquet contient un éditeur. Il a été choisi pour sa compatibilité avec l'éditeur vi classique et son grand nombre de fonctionnalités puissantes. Un éditeur est un choix très personnel de chaque utilisateur et vous pouvez le remplacer par n'importe quel éditeur si vous le désirez.

- XML::Parser

  Ce paquet est un module Perl qui interagit avec Expat.

- XZ Utils

  Ce paquet contient des programmes de compression et de décompression de fichiers. Il offre la compression la plus haute disponible et il est utile pour la décompression des paquets au format XZ ou LZMA.

- Zlib

  Ce paquet contient des routines de compression et de décompression utilisées par quelques programmes.

- Zstd

  Ce paquet contient des routines de compression et de décompression utilisées par quelques programmes. Il fournit de forts ratios de compression et une large gamme de compromis entre compression et vitesse.

Typographie

Pour faciliter ce qui suit, voici quelques conventions typographiques suivies tout au long de ce livre. Cette section contient quelques exemples du format typographique trouvé dans Linux From Scratch.

```
./configure --prefix=/usr
```

Ce style de texte est conçu pour être tapé exactement de la même façon qu'il est vu sauf si le texte indique le contraire. Il est aussi utilisé dans les sections d'explications pour identifier les commandes référencées.

Dans certains cas, une ligne logique s'étend sur deux lignes physiques voire plus avec un antislash à la fin de la ligne.

```
CC="gcc -B/usr/bin/" ../binutils-2.18/configure \
  --prefix=/tools --disable-nls --disable-werror
```

Remarquez que l'antislash doit être suivi d'un retour chariot immédiat. Tout autre caractère blanc comme des espaces ou des tabulations donnera des résultats incorrects.

```
install-info: unknown option '--dir-file=/mnt/lfs/usr/info/dir'
```

Ce style de texte (texte à largeur fixe) montre une sortie d'écran, généralement le résultat de commandes. Ce format est aussi utilisé pour afficher des noms de fichiers, comme `/etc/ld.so.conf`.

Mise en évidence

Ce style de texte est utilisé dans différents buts dans ce livre. Son but principal est de mettre en évidence les points importants.

http://www.linuxfromscratch.org/

Ce format est utilisé pour les liens, ceux de la communauté LFS et ceux référençant des pages externes. Cela inclut les guides pratiques, les emplacements de téléchargement et des sites web.

```
cat > $LFS/etc/group << "EOF"
root:x:0:
bin:x:1:
......
EOF
```

Ce format est utilisé principalement lors de la création de fichiers de configuration. La première commande indique au système de créer le fichier `$LFS/etc/group` à partir de ce qui est saisi jusqu'à ce que la séquence de fin de fichier (*End Of File*) (EOF) soit rencontrée. Donc, cette section entière est généralement saisie de la même façon.

<TEXTE À REMPLACER>

Ce format est utilisé pour intégrer du texte qui ne devra pas être saisi tel quel et qui ne devra pas être copié/collé.

[TEXTE FACULTATIF]

Ce format est utilisé pour intégrer du texte qui est facultatif.

`passwd(5)`

Ce format est utilisé pour faire référence à une page de manuel (man) spécifique. Le nombre entre parenthèses indique une section spécifique à l'intérieur des manuels. Par exemple, **passwd** a deux pages de manuel. Pour les instructions d'installation de LFS, ces deux pages de manuels seront situées dans `/usr/share/man/man1/passwd.1` et `/usr/share/man/man5/passwd.5`. Quand le livre utilise `passwd(5)`, il fait spécifiquement référence à `/usr/share/man/man5/passwd.5`. **man passwd** affichera la première page de manuel qu'il trouvera et qui correspond à « passwd », `/usr/share/man/man1/passwd.1`. Dans cet exemple, vous devrez exécuter **man 5 passwd** pour lire cette page spécifique. Il devrait être noté que la plupart des pages de man n'ont pas de nom de page dupliqué dans les différentes sections. Du coup, **man *<nom du programme>*** est généralement suffisant.

Structure

Ce livre est divisé en plusieurs parties.

Partie I — Introduction

La première partie donne quelques informations importantes, comme la façon d'installer LFS. Cette section fournit aussi des méta-informations sur le livre.

Partie II — Préparation de la construction

La deuxième partie décrit comment préparer le processus de construction : création d'une partition, téléchargement des paquets et compilation d'outils temporaires.

Partie III — Construction des outils croisés LFS et des outils temporaires

La troisième partie fournit des instructions pour construire les outils requis pour la construction du système LFS final.

Partie IV — Construction du système LFS

La quatrième partie guide le lecteur tout au long de la construction du système LFS : compilation et installation de tous les paquets un par un, mise en place des scripts de démarrage et installation du noyau. Le système Linux basique résultant est la fondation à partir de laquelle d'autres logiciels peuvent être construits pour étendre le système de la façon désirée. À la fin du livre se trouve une référence facile à utiliser et listant tous les programmes, bibliothèques et fichiers importants qui ont été installés.

Partie V — Annexes

La cinquième partie fournit des informations sur le livre lui-même, en particulier les acronymes et les termes utilisés, les remerciements, les dépendances des paquets, une liste des scripts de démarrage de LFS, la licence pour redistribuer ce livre et un catalogue complet des paquets, des programmes, des bibliothèques et des scripts.

Errata

Les logiciels utilisés pour créer un système LFS sont constamment mis à jour et améliorés. Des messages d'avertissement pour la sécurité et des corrections de bogues pourraient survenir après la sortie du livre LFS. Pour vérifier si les versions du paquet ou les instructions de cette version de LFS ont besoin de modifications pour corriger des vulnérabilités en termes de sécurité ou toute autre correction de bogue, merci de visiter *http://www.linuxfromscratch.org/lfs/errata/systemd/* avant de commencer votre construction. Vous devez noter toutes les modifications et les appliquer à la section correspondante du livre pendant votre progression lors de la construction du système LFS.

Partie I. Introduction

Chapitre 1. Introduction

1.1. Comment construire un système LFS

Le système LFS sera construit en utilisant une distribution Linux déjà installée (telle que Debian, OpenMandriva, Fedora ou openSUSE). Ce système Linux existant (l'hôte) sera utilisé comme point de départ pour fournir certains programmes nécessaires, ce qui inclut un compilateur, un éditeur de liens et un shell, pour construire le nouveau système. Sélectionnez l'option « développement » (*development*) lors de l'installation de la distribution pour disposer de ces outils.

Au lieu d'installer une distribution séparée complète sur votre machine, vous pouvez aussi utiliser le LiveCD d'une distribution commerciale.

Le Chapitre 2 de ce livre décrit comment créer une nouvelle partition native Linux et un système de fichiers. C'est l'endroit où le nouveau système LFS sera compilé et installé. Le Chapitre 3 explique quels paquets et correctifs ont besoin d'être téléchargés pour construire un système LFS et comment les stocker sur le nouveau système de fichiers. Le Chapitre 4 traite de la configuration pour un environnement de travail approprié. Merci de lire le Chapitre 4 avec attention, car il explique plusieurs problèmes importants que vous devez connaître avant de commencer à travailler sur le Chapitre 5 et les chapitres suivants.

Le Chapitre 5 explique l'installation d'une chaîne de construction initiale (binutils, gcc et glibc) avec une technique de compilation croisée qui isole les nouveaux outils du système hôte.

Le Chapitre 6 montre comment vous pouvez compiler les outils de base avec la chaîne de construction croisée tout juste construite.

Le Chapitre 7 entre ensuite dans un environnement « chroot » et utilise les outils précédemment construits pour construire les outils supplémentaires nécessaires à la construction et à la vérification du système final.

Cet effort consistant à isoler le nouveau système de la distribution hôte peut sembler excessif. Une explication technique complète est fournie dans Notes techniques sur la chaîne d'outils.

Dans le Chapitre 8, le système LFS complet est construit. Un autre avantage fournit par l'environnement chroot est qu'il vous permet de continuer à utiliser le système hôte durant la construction de LFS. Vous pouvez continuer à utiliser votre ordinateur normalement en attendant la fin de la construction d'un paquet.

Pour terminer l'installation, les scripts de démarrage sont configurés dans le Chapitre 9, le noyau et le chargeur de démarrage sont configurés dans le Chapitre 10. Le Chapitre 11 contient des informations sur la suite de l'expérience LFS après ce livre. Après avoir suivi les étapes de ce livre, l'ordinateur sera prêt à redémarrer dans le nouveau système LFS.

Ceci expose rapidement le processus. Des informations détaillées sur chaque étape sont traitées dans les chapitres suivants avec les descriptions des paquets. Les éléments qui peuvent sembler compliqués seront clarifiés et les pièces du puzzle s'emboîteront les unes après les autres tout au long de votre aventure LFS.

1.2. Quoi de neuf depuis la dernière version

Dans cette version de LFS, il y a eu une réorganisation majeure du livre pour utiliser des techniques qui évitent de changer le système hôte et qui fournissent une méthode de construction plus directe.

Vous trouverez ci-dessous la liste des mises à jour de paquets opérées depuis la version précédente du livre.

Mis à jour vers :

-
- Autoconf-2.71

- Automake-1.16.3
- Bash 5.1
- Bc 3.3.0
- Binutils-2.36.1
- Bison-3.7.5
- E2fsprogs-1.46.1
- Expat-2.2.10
- Findutils-4.8.0
- GDBM-1.19
- Glibc-2.33
- GMP-6.2.1
- Grep-3.6
- IANA-Etc-20210202
- Inetutils-2.0
- IPRoute2-5.10.0
- Kbd-2.4.0
- Kmod-28
- Less-563
- Libcap-2.48
- Libelf-0.183 (de elfutils)
- Linux-5.10.17
- Man-DB-2.9.4
- Man-pages-5.10
- Meson-0.57.1
- MPC-1.2.1
- Ninja-1.10.2
- Openssl-1.1.1j
- Perl-5.32.1
- Procps-ng-3.3.17
- Psmisc-23.4
- Python-3.9.2
- Systemd-247
- Tar-1.34
- Tcl-8.6.11
- Tzdata-2021a
- Util-Linux-2.36.2
- Vim-8.2.2433
- Zstd-1.4.8

Ajoutés :

-

- systemd-247-upstream_fixes-1.patch

Supprimés :

-
- bash-5.0-upstream_fixes-1.patch

1.3. Historique des modifications

Il s'agit de la version 10.1-systemd du livre Linux From Scratch, datant du 01 mars 2021. Si ce livre est daté de plus de six mois, une version plus récente et améliorée est probablement déjà disponible. Pour en avoir le cœur net, merci de vérifier la présence d'une nouvelle version sur l'un des miroirs via *http://www.linuxfromscratch.org/mirrors.html*.

Ci-dessous se trouve une liste des modifications apportées depuis la version précédente du livre.

Entrées dans l'historique des modifications :

- 01-03-2021
  - [bdubbs] — Publication de LFS-10.1.
- 20-02-2021
  - [bdubbs] — Mise à jour vers meson-0.57.1. Corrige *#4821*
  - [bdubbs] — Mise à jour vers Python-3.9.2. Corrige *#4820*
  - [pierre] — Correction de grub pour éviter la génération d'un énorme blob binaire avec la compression lzma. Rapporté sur la liste de diffusion lfs-support par Frans de Boer.
- 18-02-2021
  - [bdubbs] - Correction d'un bogue dans glibc qui pose problème avec les applications en chroot. Corrige *#4817*
  - [bdubbs] — Mise à jour vers bc-3.3.0. Corrige *#4819*
  - [bdubbs] — Mise à jour vers openssl-1.1.1j. Corrige *#4816*
  - [bdubbs] — Mise à jour vers meson-0.57.0. Corrige *#4815*
  - [bdubbs] — Mise à jour vers linux-5.10.17. Corrige *#4807*
- 14-02-2021
  - [renodr] — Ajout d'une explication des commandes pour pour « pwait ».
  - [bdubbs] — Mise à jour vers util-linux-2.36.2. Corrige *#4812*
  - [bdubbs] — Mise à jour vers tar-1.34. Corrige *#4813*
  - [bdubbs] — Mise à jour vers psmisc-23.4. Corrige *#4805*
  - [bdubbs] — Mise à jour vers procps-ng-3.3.17. Corrige *#4811*
  - [bdubbs] — Mise à jour vers man-db-2.9.4. Corrige *#4808*
  - [bdubbs] — Mise à jour vers eudev-3.2.10. Corrige *#4810*
  - [bdubbs] — Mise à jour vers elfutils-0.183. Corrige *#4806*
  - [bdubbs] — Mise à jour vers e2fsprogs-1.46.1. Corrige *#4809*
  - [bdubbs] — Mise à jour vers bc-3.2.7. Corrige *#4814*
- 01-02-2021
  - [bdubbs] — Mise à jour vers iana-etc-20201022. Corrige *#4722*
  - [bdubbs] — Mise à jour vers inetutils-2.0. Corrige *#4803*

- [bdubbs] — Mise à jour vers libcap-2.48. Corrige *#4802*
- [bdubbs] — Mise à jour vers binutils-2.36.1. Corrige *#4804*
- [bdubbs] — Mise à jour vers linux-5.10.13. Corrige *#4801*

- 02-02-2021
  - [pierre] — Déplacement de tcl, expect et dejagnu juste avant binutils dans le chapitre 8 : cela permet de les construire avec la glibc du chapitre 8.
  - [pierre] — Pour pouvoir construire avec des versions plus récentes de la glibc que la distribution hôte, construction d'un exécutable « file » pour l'hôte avant de le compiler avec le compilateur croisé.
  - [pierre] — Remplacement de libctf reliée (qui peut se lier de manière incorrecte à la libz de la distribution hôte) par celle qu a été liée pendant le « make » de binutils-pass2. Merci à Xi Ruoyao pour le correctif.
  - [bdubbs] — Mise à jour vers bc-3.2.6. Corrige *#4800*
  - [bdubbs] — Mise à jour vers glibc-2.33. Corrige *#4799*
  - [bdubbs] — Changement mineur du script de démarrage de mountfs. Corrige *#4798*
- 01-02-2021
  - [bdubbs] — Mise à jour vers vim-8.2.2433. Corrige *#4500*.
  - [bdubbs] — Mise à jour vers iana-etc-20210115. Corrige *#4722*
  - [bdubbs] — Mise à jour vers bc-3.2.5. Corrige *#4797*
  - [bdubbs] — Mise à jour vers e2fsprogs-1.46.0. Corrige *#4796*
  - [bdubbs] — Mise à jour vers autoconf-2.71. Corrige *#4795*
  - [bdubbs] — Mise à jour vers tzdata-2021a. Corrige *#4793*
  - [bdubbs] — Mise à jour vers perl-5.32.1. Corrige *#4792*
  - [bdubbs] — Mise à jour vers libcap-2.47. Corrige *#4791*
  - [bdubbs] — Mise à jour vers bison-3.7.5. Corrige *#4790*
  - [bdubbs] — Mise à jour vers binutils-2.36. Corrige *#4789*
  - [bdubbs] — Mise à jour vers linux-5.10.12. Corrige *#4786*
- 16-01-2020
  - [renodr] — Mise à jour vers findutils-4.8.0. Corrige *#4785*
  - [renodr] — Mise à jour vers kmod-28. Corrige *#4784*
  - [renodr] — Mise à jour vers tar-1.33. Corrige *#4783*
  - [renodr] — Mise à jour vers linux-5.10.7. Corrige *#4782*
  - [renodr] — Mise à jour vers meson-0.56.2. Corrige *#4781*
  - [renodr] — Mise à jour vers tcl-8.6.11. Corrige *#4780*
- 13-01-2021
  - [xry111] — Correction du chemin vers le socket uuidd de util-linux.
  - [xry111] — Suppression de diverses références obsolètes à /var/run, on utilise /run à la place.
- 01-01-2021
  - [bdubbs] — Mise à jour vers vim-8.2.2253. Corrige *#4500*.
  - [bdubbs] — Mise à jour vers iana-etc-20201229. Corrige *#4722*
  - [bdubbs] — Mise à jour vers gdbm-1.19. Corrige *#4779*

- [bdubbs] — Mise à jour vers tzdata-2020f. Corrige *#4777*
- [bdubbs] — Mise à jour vers man-pages-5.10. Corrige *#4776*
- [bdubbs] — Mise à jour vers iproute2-5.10.0. Corrige *#4775*
- [bdubbs] — Mise à jour vers zstd-1.4.8. Corrige *#4774*
- [bdubbs] — Mise à jour vers kbd-2.4.0. Corrige *#4773*
- [bdubbs] — Mise à jour vers linux-5.10.4. Corrige *#4772*

- 15-12-2020
  - [bdubbs] — Mise à jour vers libcap-2.46. Corrige *#4771*
  - [bdubbs] — Mise à jour vers bc-3.2.4. Corrige *#4770*
  - [bdubbs] — Mise à jour vers autoconf-2.70. Corrige *#4769*
  - [bdubbs] — Mise à jour vers openssl-1.1.1i. Corrige *#4768*
  - [bdubbs] — Mise à jour vers Python3-3.9.1. Corrige *#4767*
  - [bdubbs] — Mise à jour vers linux-5.9.14. Corrige *#4766*
  - [bdubbs] — Mise à jour vers bash-5.1 et readline-8.1. Corrige *#4764*
- 07-12-2020
  - [ken] — Correction de meson-0.56.0 pour qu'il crée correctement les fichiers pkgconfig en rapport aux bibliothèques statiques (pour sysprof dans BLFS). Corrige *#4675*.
- 03-12-2020
  - [renodr] — Mise à jour vers linux-5.9.12. Corrige *#4673*
  - [renodr] — Mise à jour vers bc-3.2.3. Corrige *#4672*
  - [renodr] — Mise à jour vers systemd-247. Corrige *#4745*
- 01-12-2020
  - [bdubbs] — Mise à jour vers ninja-1.10.2. Corrige *#4760*
  - [bdubbs] — Mise à jour vers bc-3.2.1. Corrige *#4759*
  - [bdubbs] — Mise à jour vers linux-5.9.11. Corrige *#4758*
  - [bdubbs] — Mise à jour vers automake-1.16.3. Corrige *#4757*
  - [bdubbs] — Mise à jour vers util-linux-2.36.1. Corrige *#4755*
  - [bdubbs] — Mise à jour vers iana-etc-20201124. Corrige *#4722*
  - [bdubbs] — Mise à jour vers vim-8.2.2070. Corrige *#4500*.
- 15-11-2020
  - [bdubbs] — Mise à jour vers gmp-6.2.1. Corrige *#4754*
  - [bdubbs] — Mise à jour vers iana-etc-20201110. Corrige *#4722*
  - [bdubbs] — Mise à jour vers man-pages-5.09. Corrige *#4750*
  - [bdubbs] — Mise à jour vers linux-5.9.8. Corrige *#4749*
  - [bdubbs] — Mise à jour vers libcap-2.45. Corrige *#4751*
  - [bdubbs] — Mise à jour vers grep-3.6. Corrige *#4752*
  - [bdubbs] — Mise à jour vers bison-3.7.4. Corrige *#4753*
- 01-11-2020

- [bdubbs] — Mise à jour vers elfutils-0.182. Corrige *#4748*
- [bdubbs] — Mise à jour vers meson-0.56.0. Corrige *#4747*
- [bdubbs] — Mise à jour vers linux-5.9.2. Corrige *#4746*
- [bdubbs] — Mise à jour vers mpc-1.2.1. Corrige *#4744*
- [bdubbs] — Mise à jour vers less-563. Corrige *#4743*
- [bdubbs] — Mise à jour vers tzdata-2020d. Corrige *#4742*

- 20-10-2020
  - [bdubbs] — Mise à jour vers iproute2-5.9.0. Corrige *#4739*
  - [bdubbs] — Mise à jour vers tzdata-2020c. Corrige *#4741*
  - [bdubbs] — Mise à jour vers linux-5.9.1. Corrige *#4740*
- 15-10-2020
  - [bdubbs] — Mise à jour vers iana-etc-20201012. Corrige *#4722*
  - [bdubbs] — Mise à jour vers vim-8.2.1845. Corrige *#4500*.
  - [bdubbs] — Mise à jour vers bison-3.7.3. Corrige *#4738*
  - [bdubbs] — Mise à jour vers tzdata-2020b. Corrige *#4737*
  - [bdubbs] — Mise à jour vers libcap-2.44. Corrige *#4735*
  - [bdubbs] — Mise à jour vers Python-3.9. Corrige *#4736*
  - [bdubbs] — Mise à jour vers expat-2.2.10. Corrige *#4734*
  - [bdubbs] — Mise à jour vers bc-3.1.6. Corrige *#4732*
  - [bdubbs] — Mise à jour vers linux-5.9. Corrige *#4733*
- 01-10-2020
  - [bdubbs] — Mise à jour vers grep-3.5. Corrige *#4731*
  - [bdubbs] — Mise à jour vers Python-3.8.6. Corrige *#4730*
  - [bdubbs] — Mise à jour vers openssl-1.1.1h. Corrige *#4729*
  - [bdubbs] — Mise à jour vers binutils-2.35.1. Corrige *#4728*
  - [bdubbs] — Mise à jour vers linux-5.8.12. Corrige *#4727*
- 27-09-2020
  - [bdubbs] — Ajout d'autoconf-2.69-consolidated_fixes-1.patch qui permet aux tests d'autoconf de passer.
- 15-09-2020
  - [bdubbs] — Mise à jour vers linux-5.8.9. Corrige *#4723*
  - [bdubbs] — Mise à jour vers bison-3.7.2. Corrige *#4724*
  - [bdubbs] — Mise à jour vers elfutils-0.181. Corrige *#4725*
  - [bdubbs] — Mise à jour vers meson-0.55.3. Corrige *#4726*
- 02-09-2020
  - [bdubbs] — Mise à jour vers linux-5.8.5. Corrige *#4721*
  - [bdubbs] — Mise à jour vers ninja-1.10.1. Corrige *#4719*
  - [bdubbs] — Mise à jour vers mpc-1.2.0. Corrige *#4717*
  - [bdubbs] — Mise à jour vers meson-0.55.1. Corrige *#4716*

- [bdubbs] — Mise à jour vers libcap-2.43. Corrige *#4715*

- 01-09-2020
  - [bdubbs] — Publication de LFS-10.0.

1.4. Ressources

1.4.1. FAQ

Si vous rencontrez des erreurs lors de la construction du système LFS, si vous avez des questions ou si vous pensez qu'il y a une erreur de typographie dans ce livre, merci de commencer par consulter la FAQ (Foire aux Questions) sur *http://fr.linuxfromscratch.org/faq*.

1.4.2. Listes de diffusion

Le serveur `linuxfromscratch.org` gère quelques listes de diffusion utilisées pour le développement du projet LFS. Ces listes incluent, entre autres, les listes de développement et de support. Si la FAQ ne résout pas votre problème, la prochaine étape serait de chercher dans les listes de discussion sur *http://www.linuxfromscratch.org/search.html*.

Pour connaître les listes disponibles, les conditions d'abonnement, l'emplacement des archives et quelques autres informations, allez sur *http://fr.linuxfromscratch.org/aide*.

1.4.3. IRC

Plusieurs membres de la communauté LFS offrent une assistance sur IRC. Avant d'utiliser ce mode de communication, assurez-vous que la réponse à votre question ne se trouve pas déjà dans la FAQ LFS ou dans les archives des listes de diffusion. Vous trouverez le réseau IRC à l'adresse `irc.freenode.net`. Le canal d'entre-aide en français se nomme #lfs-fr.

1.4.4. Sites miroirs

Le projet LFS a un bon nombre de miroirs configurés tout autour du monde pour faciliter l'accès au site web ainsi que le téléchargement des paquets requis. Merci de visiter le site web de LFS sur *http://www.linuxfromscratch.org/mirrors.html* pour obtenir une liste des miroirs à jour.

1.4.5. Contacts

Merci d'envoyer toutes vos questions et commentaires sur les listes de diffusion LFS (voir ci-dessus).

1.5. Aide

Si vous rencontrez une erreur ou si vous vous posez une question en travaillant avec ce livre, merci de vérifier la FAQ sur *http://fr.linuxfromscratch.org/faq#generalfaq*. Les questions y ont souvent déjà une réponse. Si votre question n'a pas sa réponse sur cette page, essayez de trouver la source du problème. L'astuce suivante vous donnera quelques conseils pour cela : *http://fr.linuxfromscratch.org/view/astuces/errors.txt*.

Si votre problème n'est pas listé dans la FAQ, recherchez dans les listes de discussion sur *http://www.linuxfromscratch.org/search.html*.

Nous avons aussi une formidable communauté LFS, volontaire pour offrir une assistance via les listes de discussion et IRC (voir la section Section 1.4, « Ressources » de ce livre). Néanmoins, nous recevons beaucoup de demandes d'aide chaque jour et un grand nombre d'entre elles ont une réponse dans la FAQ et dans les listes de discussions. Pour que nous puissions vous offrir la meilleure assistance possible, vous devez faire quelques recherches de

votre côté. Ceci nous permet de nous concentrer sur les besoins inhabituels. Si vos recherches ne vous apportent aucune solution, merci d'inclure toutes les informations adéquates (mentionnées ci-dessous) dans votre demande d'assistance.

1.5.1. Éléments à mentionner

À part une brève explication du problème, voici les éléments essentiels à inclure dans votre demande d'aide :

- La version du livre utilisée (ici 10.1-systemd)
- La distribution hôte (et sa version) que vous utilisez pour créer LFS
- La sortie du script Prérequis du système hôte
- Le paquet ou la section où le problème a été rencontré
- Le message d'erreur exact ou le symptôme reçu
- Notez si vous avez dévié du livre

Note

Dévier du livre ne signifie *pas* que nous n'allons pas vous aider. Après tout, LFS est basé sur les préférences de l'utilisateur. Nous préciser les modifications effectuées sur la procédure établie nous aide à évaluer et à déterminer les causes probables de votre problème.

1.5.2. Problèmes avec le script configure

Si quelque chose se passe mal lors de l'exécution du script **configure**, regardez le fichier `config.log` Ce fichier pourrait contenir les erreurs rencontrées lors de l'exécution de **configure** qui n'ont pas été affichées à l'écran. Incluez les lignes *intéressantes* si vous avez besoin d'aide.

1.5.3. Problèmes de compilation

L'affichage écran et le contenu de différents fichiers sont utiles pour déterminer la cause des problèmes de compilation. L'affichage de l'écran du script **configure** et du **make** peuvent être utiles. Il n'est pas nécessaire d'inclure la sortie complète mais incluez suffisamment d'informations intéressantes. Ci-dessous se trouve un exemple de type d'informations à inclure à partir de l'affichage écran de **make** :

```
gcc -DALIASPATH=\"/mnt/lfs/usr/share/locale:.\"
-DLOCALEDIR=\"/mnt/lfs/usr/share/locale\"
-DLIBDIR=\"/mnt/lfs/usr/lib\"
-DINCLUDEDIR=\"/mnt/lfs/usr/include\" -DHAVE_CONFIG_H -I. -I.
-g -O2 -c getopt1.c
gcc -g -O2 -static -o make ar.o arscan.o commands.o dir.o
expand.o file.o function.o getopt.o implicit.o job.o main.o
misc.o read.o remake.o rule.o signame.o variable.o vpath.o
default.o remote-stub.o version.o opt1.o
-lutil job.o: In function `load_too_high':
/lfs/tmp/make-3.79.1/job.c:1565: undefined reference
to `getloadavg'
collect2: ld returned 1 exit status
make[2]: *** [make] Error 1
make[2]: Leaving directory `/lfs/tmp/make-3.79.1'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/lfs/tmp/make-3.79.1'
make: *** [all-recursive-am] Error 2
```

Dans ce cas, beaucoup de personnes n’incluraient que la section du bas :

```
make [2]: *** [make] Error 1
```

Cette information n'est pas suffisante pour diagnostiquer correctement le problème, car elle note seulement que quelque chose s'est mal passé, pas *ce qui* s'est mal passé. C'est la section entière, comme dans l'exemple au dessus, qui devrait être copiée, car la commande exécutée et tout message d'erreur associé sont inclus.

Un excellent article sur les demandes d'aide sur Internet est disponible en ligne sur *http://www.gnurou.org/writing/smartquestionsfr*. Lisez et suivez les astuces de ce document pour accroître vos chances d'obtenir l'aide dont vous avez besoin.

Partie II. Préparation à la construction

Chapitre 2. Préparation du système hôte

2.1. Introduction

Dans ce chapitre, on vérifie, puis installe si besoin, les outils du système hôte nécessaires à la construction de LFS. Ensuite on prépare la partition qui contiendra le système LFS. Nous créerons la partition elle-même, lui ajouterons un système de fichiers et nous la monterons.

2.2. Prérequis du système hôte

Votre système hôte doit contenir les logiciels suivants dans leur version minimum indiquée. Cela ne devrait pas poser de problème sur la plupart des distributions Linux modernes. Notez également que certaines distributions placeront les en-têtes des logiciels dans un paquet distinct, ayant souvent la forme « <nom-du-paquet>-devel » ou « <nom-du-paquet>-dev ». Assurez-vous qu'ils sont installés si votre distribution les fournit.

Il se peut que les versions antérieures des paquets logiciels listés fonctionnent, mais elles n'ont pas été testées.

- **Bash-3.2** (/bin/sh devrait être un lien symbolique ou physique vers bash)
- **Binutils-2.25** (les versions supérieures à 2.36.1 ne sont pas recommandées, car elles n'ont pas été testées)
- **Bison-2.7** (/usr/bin/yacc devrait être un lien vers bison ou un petit script qui exécute bison)
- **Bzip2-1.0.4**
- **Coreutils-6.9**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk devrait être un lien vers gawk)
- **GCC-6.2**, y compris le compilateur C++ **g++** (les versions supérieures à 10.2.0 ne sont pas recommandées, car elles n'ont pas été testées)
- **Glibc-2.11** (les versions supérieures à 2.33 ne sont pas recommandées car elles n'ont pas été testées)
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Noyau Linux-3.2**

  Cette version du noyau est requise, car nous spécifions cette version-là lors de la construction de glibc au chapitre 6, suivant ainsi une recommandation des développeurs. Elle est aussi exigée par Udev.

  Si le noyau hôte est plus ancien que le 3.2, vous devrez remplacer le noyau par une version plus à jour. Vous pouvez employer deux méthodes pour cela. Vous pouvez d'abord voir si votre distribution Linux fournit un paquet pour le noyau 3.2 ou supérieur. Si tel est le cas, vous pouvez l'installer. Si votre distribution n'offre pas de paquet acceptable pour le noyau, ou si vous préférez ne pas l'installer, vous pouvez compiler un noyau vous-même. Les instructions pour la compilation du noyau et la configuration du chargeur d'amorçage (en supposant que le système hôte utilise GRUB) se trouvent au Chapitre 10.
- **M4-1.4.10**
- **Make-4.0**
- **Patch-2.5.4**
- **Perl-5.8.8**
- **Python-3.4**
- **Sed-4.1.5**
- **Tar-1.22**
- **Texinfo-4.7**
- **Xz-5.0.0**

Important

Remarquez que les liens symboliques mentionnés ci-dessus sont nécessaires pour construire un système LFS en utilisant les instructions contenues à l'intérieur de ce livre. Il se peut que les liens symboliques pointent vers d'autres logiciels (comme dash, mawk, etc), mais ils n'ont pas été testés, ne sont pas pris en charge par l'équipe de développement LFS et il se peut qu'ils demandent soit des déviations par rapport aux instructions, soit des correctifs supplémentaires de certains paquets.

Pour voir si votre système hôte a toutes les versions nécessaires, exécutez ceci :

```
cat > version-check.sh << "EOF"
#!/bin/bash
# Script simple pour afficher les numéros de version des outils de développement
export LC_ALL=C
bash --version | head -n1 | cut -d" " -f2-4
MYSH=$(readlink -f /bin/sh)
echo "/bin/sh -> $MYSH"
echo $MYSH | grep -q bash || echo "ERREUR : /bin/sh ne pointe pas vers bash"
unset MYSH

echo -n "Binutils : "; ld --version | head -n1 | cut -d" " -f3-
bison --version | head -n1

if [ -h /usr/bin/yacc ]; then
  echo "/usr/bin/yacc -> `readlink -f /usr/bin/yacc`";
elif [ -x /usr/bin/yacc ]; then
  echo yacc est `/usr/bin/yacc --version | head -n1`
else
  echo "yacc introuvable"
fi

bzip2 --version 2>&1 < /dev/null | head -n1 | cut -d" " -f1,6-
echo -n "Coreutils : "; chown --version | head -n1 | cut -d")" -f2
diff --version | head -n1
find --version | head -n1
gawk --version | head -n1

if [ -h /usr/bin/awk ]; then
  echo "/usr/bin/awk -> `readlink -f /usr/bin/awk`";
elif [ -x /usr/bin/awk ]; then
  echo awk est `/usr/bin/awk --version | head -n1`
else
  echo "awk introuvable"
fi
```

```
gcc --version | head -n1
g++ --version | head -n1
ldd --version | head -n1 | cut -d" " -f2-  # version de glibc
grep --version | head -n1
gzip --version | head -n1
cat /proc/version
m4 --version | head -n1
make --version | head -n1
patch --version | head -n1
echo Perl `perl -V:version`
python3 --version
sed --version | head -n1
tar --version | head -n1
makeinfo --version | head -n1  # version de texinfo
xz --version | head -n1

echo 'int main(){}' > dummy.c && g++ -o dummy dummy.c
if [ -x dummy ]
  then echo "g++ compilation OK";
  else echo "g++ compilation échouée"; fi
rm -f dummy.c dummy
EOF

bash version-check.sh
```

2.3. Les étapes de la construction de LFS

LFS est conçu pour être construit en une session. C'est-à-dire que les instructions supposent que le système ne sera pas éteint pendant la construction. Cela ne signifie pas que le système doit être construit d'une traite. Le problème est que certaines procédures doivent être relancées après un redémarrage si vous continuez LFS à différents endroits.

2.3.1. Chapitres 1–4

Ces chapitres se déroulent sur le système hôte. Si vous redémarrez, soyez vigilants :

- Les procédures effectuées en tant que `root` après la Section 2.4 ont besoin que la variable d'environnement LFS soit définie *POUR L'UTILISATEUR ROOT*.

2.3.2. Chapitres 5–6

- La partition /mnt/lfs doit être montée.
- Ces deux chapitres *doivent* être effectués en tant qu'utilisateur `lfs`. Vous devez exécuter **su - lfs** avant d'effectuer quoi que ce soit dans ces chapitres. Si vous ne le faites pas, vous risquez d'installer des paquets sur l'hôte et éventuellement de le rendre inutilisable.
- Les procédures de Instructions générales de compilation sont critiques. Si vous avez le moindre doute sur l'installation d'un paquet, assurez-vous d'avoir supprimé toute archive décompressée, extrayez de nouveau les fichiers du paquet et lancez toutes les instructions de cette section.

2.3.3. Chapitres 7–10

- La partition /mnt/lfs doit être montée.

- Quelques opérations, de « Changer de propriétaire » à « Entrer dans l'environnement chroot » doivent être effectuées en tant qu'utilisateur `root`, avec la variable d'environnement configurée pour l'utilisateur `root`.
- En entrant dans l'environnement chroot, la variable d'environnement LFS doit être définie pour l'utilisateur `root`. La variable LFS n'est plus utilisée ensuite.
- Les systèmes de fichiers virtuels doivent être montés. Ceci peut se faire avant ou après être entré dans l'environnement chroot en changeant de terminal dans le système hôte et, en `root`, en lançant les commandes de la Section 7.3.2, « Monter et peupler /dev » et de la Section 7.3.3, « Monter les systèmes de fichiers virtuels du noyau ».

2.4. Créer une nouvelle partition

Comme la plupart des autres systèmes d'exploitation, LFS est habituellement installé dans une partition dédiée. L'approche recommandée pour la construction d'un système LFS est d'utiliser une partition vide disponible ou, si vous avez assez d'espace non partitionné, d'en créer une.

Un système minimal requiert une partition d'environ 10 Go (gigaoctets). C'est suffisant pour conserver toutes les archives tar des sources et pour compiler tous les paquets. Néanmoins, si le système LFS a pour but d'être un système Linux primaire, des logiciels supplémentaires seront probablement installés et réclameront une place supplémentaire. Une partition de 30 Go est raisonnable pour offrir le nécessaire. Le système LFS lui-même ne prendra pas tout cet espace. Une grande partie de cet espace est requis pour fournir un espace temporaire libre suffisant. Compiler des paquets peut demander beaucoup d'espace disque qui sera récupéré après l'installation du paquet.

Parce qu'il n'y a pas toujours assez de mémoire (RAM) disponible pour les processus de compilation, une bonne idée est d'utiliser une petite partition comme espace d'échange `swap`. Cet espace est utilisé par le noyau pour stocker des données rarement utilisées et pour laisser plus de place disponible aux processus actifs. La partition de `swap` pour un système LFS peut être la même que celle utilisée par le système hôte, donc il n'est pas nécessaire de créer une autre partition si votre système hôte a déjà cette configuration.

Lancez un programme de partitionnement de disques tel que **cfdisk** ou **fdisk** avec une option en ligne de commande nommant le disque dur sur lequel la nouvelle partition sera créée — par exemple `/dev/sda` pour le disque primaire. Créez une partition Linux native et, si nécessaire, une partition de `swap`. Merci de vous référer aux pages de manuel de `cfdisk(8)` ou de `fdisk(8)` si vous ne savez pas encore utiliser le programme.

Note

Pour les utilisateurs expérimentés, d'autres méthodes de partitionnement sont possibles. Le nouveau système LFS peut se situer sur un *RAID* logiciel ou sur un volume *LVM* logique. Par contre, certaines options exigent un *initramfs*, ce qui relève d'un sujet avancé. Ces méthodes de partitionnement ne sont pas recommandées pour les utilisateurs de LFS pour la première fois.

Rappelez-vous de la désignation de la nouvelle partition (par exemple `sda5`). Ce livre y fera référence en tant que la partition LFS. Rappelez-vous aussi de la désignation de la partition `swap`. Ces noms seront nécessaires après pour le fichier `/etc/fstab`.

2.4.1. Autres problématiques du partitionnement

Des demandes de conseils sont souvent postées sur les listes de diffusion LFS. C'est un sujet très subjectif. Par défaut, la plupart des distributions utilisent le disque en entier, sauf une petite partie réservée à la partition d'échange. Ce n'est pas optimal avec LFS, pour plusieurs raisons. Cela réduit la flexibilité, rend plus difficile le partage de données par plusieurs distributions ou constructions de LFS, allonge le temps de sauvegarde et cela peut occuper de l'espace disque avec une allocation inefficace des structures du système de fichiers.

2.4.1.1. La partition racine

Une partition racine LFS (à ne pas confondre avec le répertoire `/root`), de vingt gigaoctets est un bon compromis pour la plupart des systèmes. Cela fournit assez de place pour construire LFS et la plupart de BLFS, tout en étant assez petit pour que plusieurs partitions puissent être créées facilement à des fins expérimentales.

2.4.1.2. La partition d'échange

La plupart des distributions créent automatiquement une partition d'échange. En général, la taille recommandée d'une partition d'échange est à peu près deux fois supérieure à la taille de la RAM physique, cependant c'est rarement nécessaire. Si vous avez un espace de disque limité, laissez la partition d'échange à deux gigaoctets et surveillez l'utilisation de la mémoire d'échange sur le disque.

Si vous voulez utiliser l'hibernation (veille sur disque) de Linux, cela écrit le contenu de la RAM vers la partition d'échange avant d'éteindre la machine. Dans ce cas la partition d'échange devrait être au moins aussi grande que la RAM installée sur le système.

L'utilisation de la mémoire d'échange n'est jamais une bonne chose. Avec un disque mécanique, vous pouvez dire si un système utilise la mémoire d'échange simplement en écoutant l'activité du disque et en observant la façon dont le système réagit aux commandes. Pour un disque SSD vous ne pourrez pas l'entendre utiliser l'espace d'échange mais vous pouvez savoir combien d'espace d'échange est utilisé avec les programmes **top** ou **free**. Vous devriez éviter d'utiliser une partition d'échange sur un SSD si possible. Votre première réaction lorsque la mémoire d'échange est utilisée devrait être de vérifier si une commande n'est pas déraisonnable, comme essayer d'éditer un fichier de cinq gigaoctets. Si l'utilisation de la mémoire d'échange devient un phénomène habituel, la meilleure solution est d'ajouter de la RAM à votre système.

2.4.1.3. La partition Bios de Grub

Si le *disque de démarrage* est partitionné avec une table de partition GUID (GPT), alors une petite partition de l'ordre d'1 Mo doit être créée si elle n'existe pas déjà. Cette partition n'est pas formatée, mais doit être disponible pour que GRUB l'utilise pendant l'installation du chargeur de démarrage. Cette partition sera normalement intitulée « *BIOS Boot* » si vous utilisez **fdisk** ou aura le code *EF02* avec **gdisk**.

Note

La partition Bios de Grub doit être présente sur le disque que le BIOS utilise pour démarrer le système. Ce n'est pas nécessairement le même disque que celui sur lequel la partition racine de LFS est installée. Les disques d'un système peuvent utiliser des types de tables de partitions différents. Le besoin de cette partition ne dépend que du type de table de partitions du disque de démarrage.

2.4.1.4. Partitions de commodité

Plusieurs autres partitions ne sont pas nécessaires mais vous devriez les étudier lorsque vous aménagez un disque dur. La liste suivante n'est pas exhaustive mais peut être perçue comme un guide.

- /boot – Fortement recommandée. Utilisez cette partition pour conserver les noyaux et d'autres informations de démarrage. Pour minimiser les problèmes de démarrage avec les gros disques, faites-en la première partition physique sur votre premier disque dur. Une taille de partition de 200 mégaoctets est parfaitement adaptée.
- /home – Fortement recommandée. Partage votre répertoire home et vos paramètres utilisateur entre plusieurs distributions ou constructions de LFS. La taille est en général très importante et dépend de l'espace disque disponible.
- /usr – On utilise généralement une partition /usr séparée si on fournit un serveur pour un client léger ou une station de travail sans disque. Elle n'est normalement pas nécessaire pour LFS. Une taille de dix gigaoctets gérera la plupart des installations.

- /opt – Ce répertoire est surtout utile pour BLFS où vous pouvez installer plusieurs versions de gros paquets tels que Gnome ou KDE sans mettre les fichiers dans la hiérarchie /usr. Si vous l'utilisez, 5 à 10 gigaoctets sont généralement adaptés.
- /tmp – Un répertoire /tmp séparé est rare, mais utile si vous configurez un client léger. Cette partition, si vous l'utilisez, ne nécessitera en général pas plus de deux gigaoctets.
- /usr/src – Cette partition est très utile pour fournir un endroit où conserver les fichiers des sources de BLFS et les partager entre des constructions LFS. Vous pouvez aussi l'utiliser comme lieu de construction des paquets BLFS. Une partition raisonnablement grande de 30-50 gigaoctets permet d'avoir beaucoup de place.

Vous devez spécifier toute partition que vous voulez voir montée automatiquement au démarrage dans `/etc/fstab`. Les détails sur la façon de spécifier les partitions seront donnés à la Section 10.2, « Créer le fichier /etc/fstab ».

2.5. Créer un système de fichiers sur la partition

Maintenant qu'une partition vierge est prête, le système de fichiers peut être créé. LFS peut utiliser n'importe quel système de fichiers reconnu par le noyau Linux, mais les types les plus classiques sont ext3 et ext4. Le choix d'un système de fichiers peut être complexe et il dépend des caractéristiques des fichiers et de la taille de la partition. Par exemple :

ext2
: convient aux petites partitions rarement renouvelées telles que /boot.

ext3
: mise à jour de l'ext2 comprenant un journal aidant à récupérer l'état de la partition en cas d'arrêt brutal. On l'utilise en général dans une perspective généraliste.

ext4
: est la dernière version des systèmes de fichiers de la famille ext. Il offre de nouvelles possibilités, notamment l'horodatage à la nanoseconde, la création et l'utilisation de très gros fichiers (16 To), et des améliorations de vitesse.

D'autres systèmes de fichiers comme FAT32, NTFS, ReiserFS, JFS et XFS servent à des fins plus spécifiques. Vous pouvez trouver plus d'informations sur ces systèmes de fichiers sur *https://fr.wikipedia.org/wiki/Liste_des_syst%C3%A8mes_de_fichiers*.

LFS suppose que le système de fichiers racine (/) est de type ext4. Pour créer un système de fichiers `ext4` sur la partition LFS, lancez ce qui suit :

```
mkfs -v -t ext4 /dev/<xxx>
```

Remplacez *<xxx>* par le nom de la partition LFS.

Si vous utilisez une partition de `swap` existante, il n'est pas nécessaire de la formater. Si vous avez créé une nouvelle partition `swap`, elle devra être initialisée, pour pouvoir être utilisée, en exécutant la commande :

```
mkswap /dev/<yyy>
```

Remplacez *<yyy>* par le nom de la partition de `swap`.

2.6. Définir la variable $LFS

Tout au long de ce livre, la variable d'environnement `LFS` sera utilisée à plusieurs reprises. Vous devriez vous assurer de toujours définir cette variable pendant le processus de construction de votre LFS. Elle devrait contenir le nom du répertoire où vous construirez votre système LFS — nous utiliserons `/mnt/lfs` comme exemple mais le choix du répertoire vous appartient. Si vous construisez LFS sur une partition à part, ce répertoire sera le point de montage de la partition. Choisissez un répertoire et définissez la variable avec la commande suivante :

```
export LFS=/mnt/lfs
```

Le fait d'avoir défini cette variable est un avantage dans des commandes comme **mkdir -v $LFS/tools** qu'on peut taper littéralement. Le shell remplacera automatiquement « $LFS » par « /mnt/lfs » (ou le nom défini dans la variable) quand il traitera la ligne de commande.

Attention

N'oubliez pas de vérifier que `LFS` est définie à chaque fois que vous quittez et revenez dans l'environnement de travail (par exemple, en faisant un **su** en `root` ou autres utilisateurs). Vérifiez que la variable `LFS` est définie correctement avec :

```
echo $LFS
```

Assurez-vous que la sortie affiche le chemin vers l'endroit où vous construisez votre système LFS, qui est `/mnt/lfs` si vous avez suivi l'exemple fourni. Si la sortie ne va pas, utilisez la commande donnée ci-dessus dans cette page pour mettre dans `$LFS` le bon nom de répertoire.

Note

Une manière de vous assurer que la variable `LFS` est toujours définie est d'éditer le fichier `.bash_profile` à la fois dans votre répertoire personnel et dans `/root/.bash_profile` et d'y entrer la commande export ci-dessus. De plus, le shell indiqué dans le fichier `/etc/passwd` de tous les utilisateurs ayant besoin de la variable `LFS` doit être bash afin de s'assurer que le fichier `/root/.bash_profile` est inclus dans le processus de connexion.

Une autre chose à prendre en compte est la méthode que vous utilisez pour vous connecter au système hôte. Si vous vous connectez via un gestionnaire d'affichage graphique, le fichier `.bash_profile` de l'utilisateur n'est normalement pas utilisé lorsque le gestionnaire lance un terminal virtuel. Dans ce cas, ajoutez la commande d'export au fichier `.bashrc` à la fois pour l'utilisateur et pour `root`. En plus, certaines distributions ont des instructions qui empêchent le chargement de `.bashrc` dans une invocation non interactive de bash. Assurez-vous d'ajouter la commande d'export avant le test pour l'utilisation non interactive si c'est le cas.

2.7. Monter la nouvelle partition

Maintenant qu'un système de fichiers a été créé, la partition doit être accessible. Pour cela, la partition a besoin d'être montée sur un point de montage choisi. Pour ce livre, il est supposé que le système de fichiers est monté sur le répertoire spécifié par la variable d'environnement `LFS` comme décrit dans la section précédente.

Créez le point de montage et montez le système de fichiers LFS en lançant :

```
mkdir -pv $LFS
mount -v -t ext4 /dev/<xxx> $LFS
```

Remplacez *<xxx>* par le nom de la partition LFS.

Si vous utilisez plusieurs partitions pour LFS (par exemple une pour `/` et une autre pour `/usr`), montez-les en utilisant :

```
mkdir -pv $LFS
mount -v -t ext4 /dev/<xxx> $LFS
mkdir -v $LFS/usr
mount -v -t ext4 /dev/<yyy> $LFS/usr
```

Remplacez *<xxx>* et *<yyy>* par les noms de partition appropriés.

Assurez-vous que cette nouvelle partition n'est pas montée avec des droits trop restrictifs (tels que les options `nosuid` ou `nodev`). Lancez la commande **mount** sans aucun paramètre pour voir les options configurées pour la partition LFS montée. Si les options `nosuid` ou `nodev` sont configurées, la partition devra être remontée.

Avertissement

Les instructions ci-dessus supposent que vous ne redémarrerez pas votre ordinateur tout le long du processus LFS. Si vous éteignez votre système, vous devrez soit remonter la partition LFS à chaque redémarrage de la construction ou modifier votre le fichier /etc/fstab de votre système hôte pour la remonter automatiquement au démarrage. Par exemple :

```
/dev/<xxx>  /mnt/lfs ext4   defaults      1     1
```

Si vous utilisez des partitions facultatives supplémentaires, assurez-vous de les ajouter aussi.

Si vous utilisez une partition de `swap`, assurez-vous qu'elle est activée en lançant la commande **swapon** :

```
/sbin/swapon -v /dev/<zzz>
```

Remplacez *<zzz>* par le nom de la partition de `swap`.

Maintenant qu'il existe un endroit établi pour travailler, il est temps de télécharger les paquets.

Chapitre 3. Paquets et correctifs

3.1. Introduction

Ce chapitre inclut une liste de paquets devant être téléchargés pour construire un système Linux basique. Les numéros de versions affichés correspondent aux versions des logiciels qui, selon nous, fonctionnent à coup sûr. Ce livre est basé sur leur utilisation. Nous vous recommandons fortement de ne pas utiliser de versions supérieures, car les commandes de construction pour une version pourraient ne pas fonctionner avec une version plus récente. Les versions plus récentes pourraient aussi avoir des problèmes nécessitant des contournements. Ces derniers seront développés et stabilisés dans la version de développement du livre.

Il se peut que les emplacements de téléchargement ne soient pas toujours accessibles. Si un emplacement de téléchargement a changé depuis la publication de ce livre, Google (*http://www.google.com/*) offre un moteur de recherche utile pour la plupart des paquets. Si cette recherche est infructueuse, essayez un des autres moyens de téléchargement disponible sur *http://www.linuxfromscratch.org/lfs/packages.html#packages*.

Les paquets et les correctifs téléchargés doivent être stockés quelque part où ils seront facilement disponibles pendant toute la construction. Un répertoire fonctionnel est aussi requis pour déballer les sources et pour les construire. Vous pouvez utiliser le répertoire `$LFS/sources` à la fois comme emplacement de stockage pour les archives tar et les correctifs, mais aussi comme répertoire fonctionnel. En utilisant ce répertoire, les éléments requis seront situés sur la partition LFS et seront disponibles à toutes les étapes du processus de construction.

Pour créer ce répertoire, lancez, en tant qu'utilisateur `root`, avant de commencer la session de téléchargement :

```
mkdir -v $LFS/sources
```

Donnez le droit d'écriture et le droit sticky sur ce répertoire. « Sticky » signifie que même si de nombreux utilisateurs peuvent écrire sur un répertoire, seul le propriétaire du fichier peut supprimer ce fichier à l'intérieur du répertoire sticky. La commande suivante activera les droits d'écriture et sticky :

```
chmod -v a+wt $LFS/sources
```

Il y a plusieurs moyen d'obtenir tous les paquets nécessaires et les correctifs requis pour construire LFS :

- Vous pouvez télécharger les fichiers individuellement comme décrit dans les deux prochaines sections.
- Pour les versions stables du livre, une archive de tous les fichiers nécessaires est disponible au téléchargement sur l'un des miroirs de LFS listés sur *http://www.linuxfromscratch.org/mirrors.html#files*.
- Vous pouvez télécharger les fichiers avec **wget** et un wget-list comme décrit ci-dessous.

Pour télécharger tous les paquets et les correctifs en utilisant *wget-list* comme entrée pour **wget**, utilisez :

```
wget --input-file=wget-list --continue --directory-prefix=$LFS/sources
```

En outre, à partir de LFS-7.0, un fichier séparé, *md5sums*, peut être utilisé pour vérifier que tous les paquets sont disponibles avant de continuer. Mettez ce fichier dans `$LFS/sources` et lancez :

```
pushd $LFS/sources
  md5sum -c md5sums
popd
```

Vous pouvez utiliser cette vérification après avoir récupéré les fichiers nécessaires avec n'importe quelle méthode proposée plus haut.

3.2. Tous les paquets

Téléchargez ou obtenez d'une autre façon les paquets suivants :

• Acl (2.2.53) — 513 Ko :
Page d'accueil : *https://savannah.nongnu.org/projects/acl*
Téléchargement : *http://download.savannah.gnu.org/releases/acl/acl-2.2.53.tar.gz*
Somme de contrôle MD5 : `007aabf1dbb550bcddde52a244cd1070`

• Attr (2.4.48) — 457 Ko :
Page d'accueil : *https://savannah.nongnu.org/projects/attr*
Téléchargement : *http://download.savannah.gnu.org/releases/attr/attr-2.4.48.tar.gz*
Somme de contrôle MD5 : `bc1e5cb5c96d99b24886f1f527d3bb3d`

• Autoconf (2.71) — 1,263 Ko:
Page d'accueil : *http://www.gnu.org/software/autoconf/*
Téléchargement : *http://ftp.gnu.org/gnu/autoconf/autoconf-2.71.tar.xz*
Somme de contrôle MD5 : `12cfa1687ffa2606337efe1a64416106`

• Automake (1.16.3) — 1,554 Ko:
Page d'accueil : *http://www.gnu.org/software/automake/*
Téléchargement : *http://ftp.gnu.org/gnu/automake/automake-1.16.3.tar.xz*
Somme de contrôle MD5 : `c27f608a4e1f302ec7ce42f1251c184e`

• Bash (5.1) — 10,214 Ko:
Page d'accueil : *http://www.gnu.org/software/bash/*
Téléchargement : *http://ftp.gnu.org/gnu/bash/bash-5.1.tar.gz*
Somme de contrôle MD5 : `bb91a17fd6c9032c26d0b2b78b50aff5`

• Bc (3.3.0) — 225 Ko:
Page d'accueil : *https://git.yzena.com/gavin/bc*
Téléchargement : *https://github.com/gavinhoward/bc/releases/download/3.3.0/bc-3.3.0.tar.xz*
Somme de contrôle MD5 : `452ae2d467b1d7212bb7896c0c689825`

• Binutils (2.36.1) — 22,239 Ko:
Page d'accueil : *http://www.gnu.org/software/binutils/*
Téléchargement : *http://ftp.gnu.org/gnu/binutils/binutils-2.36.1.tar.xz*
Somme de contrôle MD5 : `628d490d976d8957279bbbff06cf29d4`

• Bison (3.7.5) — 2,561 Ko:
Page d'accueil : *http://www.gnu.org/software/bison/*
Téléchargement : *http://ftp.gnu.org/gnu/bison/bison-3.7.5.tar.xz*
Somme de contrôle MD5 : `9b762dc24a6723f86d14d957d3deeb90`

• Bzip2 (1.0.8) — 792 Ko:
Téléchargement : *https://www.sourceware.org/pub/bzip2/bzip2-1.0.8.tar.gz*
Somme de contrôle MD5 : `67e051268d0c475ea773822f7500d0e5`

• Check (0.15.2) — 760 Ko :
Page d'accueil : *https://libcheck.github.io/check*
Téléchargement : *https://github.com/libcheck/check/releases/download/0.15.2/check-0.15.2.tar.gz*
Somme de contrôle MD5 : `50fcafcecde5a380415b12e9c574e0b2`

• Coreutils (8.32) — 5,418 Ko:
Page d'accueil : *http://www.gnu.org/software/coreutils/*
Téléchargement : *http://ftp.gnu.org/gnu/coreutils/coreutils-8.32.tar.xz*
Somme de contrôle MD5 : `022042695b7d5bcf1a93559a9735e668`

• D-Bus (1.12.20) — 2,048 Ko:
Page d'accueil : *https://www.freedesktop.org/wiki/Software/dbus*
Téléchargement : *https://dbus.freedesktop.org/releases/dbus/dbus-1.12.20.tar.gz*
Somme de contrôle MD5 : `dfe8a71f412e0b53be26ed4fbfdc91c4`

• DejaGNU (1.6.2) — 514 Ko:
Page d'accueil : *http://www.gnu.org/software/dejagnu/*
Téléchargement : *http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.6.2.tar.gz*
Somme de contrôle MD5 : `e1b07516533f351b3aba3423fafeffd6`

• Diffutils (3.7) — 1,415 Ko:
Page d'accueil : *http://www.gnu.org/software/diffutils/*
Téléchargement : *http://ftp.gnu.org/gnu/diffutils/diffutils-3.7.tar.xz*
Somme de contrôle MD5 : `4824adc0e95dbbf11dfbdfaad6a1e461`

• E2fsprogs (1.46.1) — 9,269 Ko:
Page d'accueil : *http://e2fsprogs.sourceforge.net/*
Téléchargement : *https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.46.1/e2fsprogs-1.46.1.tar.gz*
Somme de contrôle MD5 : `8c52585522b7ca6bdae2bdecba27b3a4`

• Elfutils (0.183) — 8,896 Ko:
Page d'accueil : *https://sourceware.org/ftp/elfutils/*
Téléchargement : *https://sourceware.org/ftp/elfutils/0.183/elfutils-0.183.tar.bz2*
Somme de contrôle MD5 : `6f58aa1b9af1a5681b1cbf63e0da2d67`

• Expat (2.2.10) — 416 Ko :
Page d'accueil : *https://libexpat.github.io/*
Téléchargement : *https://prdownloads.sourceforge.net/expat/expat-2.2.10.tar.xz*
Somme de contrôle MD5 : `e0fe49a6b3480827c9455e4cfc799133`

• Expect (5.45.4) — 618 Ko:
Page d'accueil : *https://core.tcl.tk/expect/*
Téléchargement : *https://prdownloads.sourceforge.net/expect/expect5.45.4.tar.gz*
Somme de contrôle MD5 : `00fce8de158422f5ccd2666512329bd2`

• File (5.39) — 932 Ko:
Page d'accueil : *https://www.darwinsys.com/file/*
Téléchargement : *http://ftp.astron.com/pub/file/file-5.39.tar.gz*
Somme de contrôle MD5 : `1c450306053622803a25647d88f80f25`

Note
Il se peut que le fichier (5.39) ne soit plus disponible à l'emplacement indiqué. Les administrateurs du site de l'emplacement principal de téléchargement suppriment régulièrement les anciennes versions lorsque de nouvelles sortent. Vous pouvez trouver un autre emplacement pour le téléchargement qui peut conserver la bonne version disponible sur *http://www.linuxfromscratch.org/lfs/download.html#ftp*.

• Findutils (4.8.0) — 1,940 Ko:
Page d'accueil : *http://www.gnu.org/software/findutils/*
Téléchargement : *http://ftp.gnu.org/gnu/findutils/findutils-4.8.0.tar.xz*
Somme de contrôle MD5 : `eeefe2e6380931a77dfa6d9350b43186`

• Flex (2.6.4) — 1,386 Ko:
Page d'accueil : *https://github.com/westes/flex*
Téléchargement : *https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz*
Somme de contrôle MD5 : `2882e3179748cc9f9c23ec593d6adc8d`

• Gawk (5.1.0) — 3,081 Ko:
Page d'accueil : *http://www.gnu.org/software/gawk/*
Téléchargement : *http://ftp.gnu.org/gnu/gawk/gawk-5.1.0.tar.xz*
Somme de contrôle MD5 : `8470c34eeecc41c1aa0c5d89e630df50`

• GCC (10.2.0) — 73,247 Ko:
Page d'accueil : *https://gcc.gnu.org/*
Téléchargement : *http://ftp.gnu.org/gnu/gcc/gcc-10.2.0/gcc-10.2.0.tar.xz*
Somme de contrôle MD5 : `e9fd9b1789155ad09bcf3ae747596b50`

• GDBM (1.19) — 946 Ko :
Page d'accueil : *http://www.gnu.org/software/gdbm/*
Téléchargement : *http://ftp.gnu.org/gnu/gdbm/gdbm-1.19.tar.gz*
Somme de contrôle MD5 : `aeb29c6a90350a4c959cd1df38cd0a7e`

• Gettext (0.21) — 9,487 Ko:
Page d'accueil : *http://www.gnu.org/software/gettext/*
Téléchargement : *http://ftp.gnu.org/gnu/gettext/gettext-0.21.tar.xz*
Somme de contrôle MD5 : `40996bbaf7d1356d3c22e33a8b255b31`

• Glibc (2.33) — 16,663 Ko :
Page d'accueil : *http://www.gnu.org/software/libc/*
Téléchargement : *http://ftp.gnu.org/gnu/glibc/glibc-2.33.tar.xz*
Somme de contrôle MD5 : `390bbd889c7e8e8a7041564cb6b27cca`

• GMP (6.2.1) — 1,980 Ko:
Page d'accueil : *http://www.gnu.org/software/gmp/*
Téléchargement : *http://ftp.gnu.org/gnu/gmp/gmp-6.2.1.tar.xz*
Somme de contrôle MD5 : `0b82665c4a92fd2ade7440c13fcaa42b`

• Gperf (3.1) — 1,188 Ko :
Page d'accueil : *http://www.gnu.org/software/gperf/*
Téléchargement : *http://ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz*
Somme de contrôle MD5 : `9e251c0a618ad0824b51117d5d9db87e`

• Grep (3.6) — 1,553 Ko:
Page d'accueil : *http://www.gnu.org/software/grep/*
Téléchargement : *http://ftp.gnu.org/gnu/grep/grep-3.6.tar.xz*
Somme de contrôle MD5 : `f47fe27049510b2249dba7f862ac1b51`

• Groff (1.22.4) — 4,044 Ko:
Page d'accueil : *http://www.gnu.org/software/groff/*
Téléchargement : *http://ftp.gnu.org/gnu/groff/groff-1.22.4.tar.gz*
Somme de contrôle MD5 : `08fb04335e2f5e73f23ea4c3adbf0c5f`

• GRUB (2.04) — 6,245 Ko:
Page d'accueil : *http://www.gnu.org/software/grub/*
Téléchargement : *https://ftp.gnu.org/gnu/grub/grub-2.04.tar.xz*
Somme de contrôle MD5 : `5aaca6713b47ca2456d8324a58755ac7`

• Gzip (1.10) — 757 Ko:
Page d'accueil : *http://www.gnu.org/software/gzip/*
Téléchargement : *http://ftp.gnu.org/gnu/gzip/gzip-1.10.tar.xz*
Somme de contrôle MD5 : `691b1221694c3394f1c537df4eee39d3`

• Iana-Etc (20210202) — 578 Ko:
Page d'accueil : *https://www.iana.org/protocols*
Téléchargement : *https://github.com/Mic92/iana-etc/releases/download/20210202/iana-etc-20210202.tar.gz*
Somme de contrôle MD5 : `1c193a4d6ca36274570d1505140a7bee`

• Inetutils (2.0) — 1,462 Ko:
Page d'accueil : *http://www.gnu.org/software/inetutils/*
Téléchargement : *http://ftp.gnu.org/gnu/inetutils/inetutils-2.0.tar.xz*
Somme de contrôle MD5 : `5e1018502cd131ed8e42339f6b5c98aa`

• Intltool (0.51.0) — 159 Ko :
Page d'accueil : *https://freedesktop.org/wiki/Software/intltool*
Téléchargement : *https://launchpad.net/intltool/trunk/0.51.0/+download/intltool-0.51.0.tar.gz*
Somme de contrôle MD5 : `12e517cac2b57a0121cda351570f1e63`

• IPRoute2 (5.10.0) — 781 Ko:
Page d'accueil : *https://www.kernel.org/pub/linux/utils/net/iproute2/*
Téléchargement : *https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-5.10.0.tar.xz*
Somme de contrôle MD5 : `19ffea480a21e600453776b7225f3319`

• Kbd (2.4.0) — 1,095 Ko:
Page d'accueil : *http://ftp.altlinux.org/pub/people/legion/kbd*
Téléchargement : *https://www.kernel.org/pub/linux/utils/kbd/kbd-2.4.0.tar.xz*
Somme de contrôle MD5 : `3cac5be0096fcf7b32dcbd3c53831380`

• Kmod (28) — 540 Ko:
Téléchargement : *https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-28.tar.xz*
Somme de contrôle MD5 : `0a2b887b1b3dfb8c0b3f41f598203e56`

• Less (563) — 328 Ko:
Page d'accueil : *http://www.greenwoodsoftware.com/less/*
Téléchargement : *http://www.greenwoodsoftware.com/less/less-563.tar.gz*
Somme de contrôle MD5 : `1ee44fa71447a845f6eef5b3f38d2781`

• Libcap (2.48) — 130 Ko :
Page d'accueil : *https://sites.google.com/site/fullycapable/*
Téléchargement : *https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.48.tar.xz*
Somme de contrôle MD5 : `ca71693a9abe4e0ad9cc33a755ee47e0`

• Libffi (3.3) — 1,275 Ko:
Page d'accueil : *https://sourceware.org/libffi/*
Téléchargement : *https://sourceware.org/pub/libffi/libffi-3.3.tar.gz*
Somme de contrôle MD5 : `6313289e32f1d38a9df4770b014a2ca7`

• Libpipeline (1.5.3) — 972 Ko :
Page d'accueil : *http://libpipeline.nongnu.org/*
Téléchargement : *http://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.3.tar.gz*
Somme de contrôle MD5 : `dad443d0911cf9f0f1bd90a334bc9004`

• Libtool (2.4.6) — 951 Ko:
Page d'accueil : *http://www.gnu.org/software/libtool/*
Téléchargement : *http://ftp.gnu.org/gnu/libtool/libtool-2.4.6.tar.xz*
Somme de contrôle MD5 : `1bfb9b923f2c1339b4d2ce1807064aa5`

• Linux (5.10.17) — 113,548 Ko:
Page d'accueil : *https://www.kernel.org/*
Téléchargement : *https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.10.17.tar.xz*
Somme de contrôle MD5 : `4908707ed841923d8d1814130d5c380f`

Note

Le noyau Linux est régulièrement mis à jour, souvent suite à la découverte de nouvelles failles de sécurité. Vous devriez utiliser la version 5.10.x la plus récente disponible du noyau, sauf si la page d'errata indique le contraire.

Pour les utilisateurs ayant un débit limité ou une bande passante chère, si vous souhaitez mettre à jour le noyau Linux, une version en ligne de commande du paquet et des correctifs peuvent être téléchargées séparément. Ceci peut économiser du temps ou de l'argent pour une mise à jour d'un niveau de correctif mineure (subsequent) à l'intérieur d'une version mineure.

• M4 (1.4.18) — 1,180 Ko:
Page d'accueil : *http://www.gnu.org/software/m4/*
Téléchargement : *http://ftp.gnu.org/gnu/m4/m4-1.4.18.tar.xz*
Somme de contrôle MD5 : `730bb15d96fffe47e148d1e09235af82`

• Make (4.3) — 2,263 Ko:
Page d'accueil : *http://www.gnu.org/software/make/*
Téléchargement : *http://ftp.gnu.org/gnu/make/make-4.3.tar.gz*
Somme de contrôle MD5 : `fc7a67ea86ace13195b0bce683fd4469`

• Man-DB (2.9.4) — 1,865 Ko:
Page d'accueil : *https://www.nongnu.org/man-db/*
Téléchargement : *http://download.savannah.gnu.org/releases/man-db/man-db-2.9.4.tar.xz*
Somme de contrôle MD5 : `6e233a555f7b9ae91ce7cd0faa322bce`

• Man-pages (5.10) — 1,707 Ko:
Page d'accueil : *https://www.kernel.org/doc/man-pages/*
Téléchargement : *https://www.kernel.org/pub/linux/docs/man-pages/man-pages-5.10.tar.xz*
Somme de contrôle MD5 : `4ae3f74a1beddd919936e1058642644c`

• Meson (0.57.1) — 1,806 Ko:
Page d'accueil : *https://mesonbuild.com*
Téléchargement : *https://github.com/mesonbuild/meson/releases/download/0.57.1/meson-0.57.1.tar.gz*
Somme de contrôle MD5 : `fbd744560351491892478a36a1586815`

• MPC (1.2.1) — 820 Ko:
Page d'accueil : *http://www.multiprecision.org/*
Téléchargement : *https://ftp.gnu.org/gnu/mpc/mpc-1.2.1.tar.gz*
Somme de contrôle MD5 : `9f16c976c25bb0f76b50be749cd7a3a8`

• MPFR (4.1.0) — 1,490 Ko :
Page d'accueil : *https://www.mpfr.org/*
Téléchargement : *http://www.mpfr.org/mpfr-4.1.0/mpfr-4.1.0.tar.xz*
Somme de contrôle MD5 : `bdd3d5efba9c17da8d83a35ec552baef`

• Ncurses (6.2) — 3,346 Ko:
Page d'accueil : *http://www.gnu.org/software/ncurses/*
Téléchargement : *http://ftp.gnu.org/gnu/ncurses/ncurses-6.2.tar.gz*
Somme de contrôle MD5 : `e812da327b1c2214ac1aed440ea3ae8d`

• Ninja (1.10.2) — 209 Ko:
Page d'accueil : *https://ninja-build.org/*
Téléchargement : *https://github.com/ninja-build/ninja/archive/v1.10.2/ninja-1.10.2.tar.gz*
Somme de contrôle MD5 : `639f75bc2e3b19ab893eaf2c810d4eb4`

• OpenSSL (1.1.1j) — 9,593 Ko:
Page d'accueil : *https://www.openssl.org/*
Téléchargement : *https://www.openssl.org/source/openssl-1.1.1j.tar.gz*
Somme de contrôle MD5 : `cccaa064ed860a2b4d1303811bf5c682`

• Patch (2.7.6) — 766 Ko:
Page d'accueil : *https://savannah.gnu.org/projects/patch/*
Téléchargement : *http://ftp.gnu.org/gnu/patch/patch-2.7.6.tar.xz*
Somme de contrôle MD5 : `78ad9937e4caadcba1526ef1853730d5`

• Perl (5.32.1) — 12,316 Ko:
Page d'accueil : *https://www.perl.org/*
Téléchargement : *https://www.cpan.org/src/5.0/perl-5.32.1.tar.xz*
Somme de contrôle MD5 : `7f104064b906ad8c7329ca5e409a32d7`

• Pkg-config (0.29.2) — 1,970 Ko :
Page d'accueil : *https://www.freedesktop.org/wiki/Software/pkg-config*
Téléchargement : *https://pkg-config.freedesktop.org/releases/pkg-config-0.29.2.tar.gz*
Somme de contrôle MD5 : `f6e931e319531b736fadc017f470e68a`

• Procps (3.3.17) — 985 Ko:
Page d'accueil : *https://sourceforge.net/projects/procps-ng*
Téléchargement : *https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-3.3.17.tar.xz*
Somme de contrôle MD5 : `d60613e88c2f442ebd462b5a75313d56`

• Psmisc (23.4) — 362 Ko:
Page d'accueil : *http://psmisc.sourceforge.net/*
Téléchargement : *https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.4.tar.xz*
Somme de contrôle MD5 : `8114cd4489b95308efe2509c3a406bbf`

• Python (3.9.2) — 18,477 Ko:
Page d'accueil : *https://www.python.org/*
Téléchargement : *https://www.python.org/ftp/python/3.9.2/Python-3.9.2.tar.xz*
Somme de contrôle MD5 : `f0dc9000312abeb16de4eccce9a870ab`

• Documentation Python (3.9.2) — 6,659 Ko:
Téléchargement : *https://www.python.org/ftp/python/doc/3.9.2/python-3.9.2-docs-html.tar.bz2*
Somme de contrôle MD5 : `719cd64a4c5768b646b716df20229400`

• Readline (8.1) — 2,924 Ko:
Page d'accueil : *https://tiswww.case.edu/php/chet/readline/rltop.html*
Téléchargement : *http://ftp.gnu.org/gnu/readline/readline-8.1.tar.gz*
Somme de contrôle MD5 : `e9557dd5b1409f5d7b37ef717c64518e`

• Sed (4.8) — 1,317 Ko:
Page d'accueil : *http://www.gnu.org/software/sed/*
Téléchargement : *http://ftp.gnu.org/gnu/sed/sed-4.8.tar.xz*
Somme de contrôle MD5 : `6d906edfdb3202304059233f51f9a71d`

• Shadow (4.8.1) — 1,574 Ko:
Téléchargement : *https://github.com/shadow-maint/shadow/releases/download/4.8.1/shadow-4.8.1.tar.xz*
Somme de contrôle MD5 : `4b05eff8a427cf50e615bda324b5bc45`

• Systemd (247) — 9,656 Ko:
Page d'accueil : *https://www.freedesktop.org/wiki/Software/systemd/*
Téléchargement : *https://github.com/systemd/systemd/archive/v247/systemd-247.tar.gz*
Somme de contrôle MD5 : `b0d6e603b814908acc69db29bbfb023c`

• Pages de manuel de Systemd (247) — 612 Ko:
Page d'accueil : *https://www.freedesktop.org/wiki/Software/systemd/*
Téléchargement : *http://anduin.linuxfromscratch.org/LFS/systemd-man-pages-247.tar.xz*
Somme de contrôle MD5 : `438c98be200e1c3b308e58a3399d4465`

Note
L'équipe systemd de Linux From Scratch génère sa propre archive des pages de manuel à partir des sources de systemd. Cela est fait pour éviter des dépendances inutiles.

• Tar (1.34) — 2,174 Ko:
Page d'accueil : *http://www.gnu.org/software/tar/*
Téléchargement : *http://ftp.gnu.org/gnu/tar/tar-1.34.tar.xz*
Somme de contrôle MD5 : `9a08d29a9ac4727130b5708347c0f5cf`

• Tcl (8.6.11) — 10,020 Ko:
Page d'accueil : *http://tcl.sourceforge.net/*
Téléchargement : *https://downloads.sourceforge.net/tcl/tcl8.6.11-src.tar.gz*
Somme de contrôle MD5 : `8a4c004f48984a03a7747e9ba06e4da4`

• Documentation de Tcl (8.6.11) — 1,172 Ko :
Téléchargement : *https://downloads.sourceforge.net/tcl/tcl8.6.11-html.tar.gz*
Somme de contrôle MD5 : `e358a9140c3a171e42f18c8a7f6a36ea`

• Texinfo (6.7) — 4,237 Ko:
Page d'accueil : *http://www.gnu.org/software/texinfo/*
Téléchargement : *http://ftp.gnu.org/gnu/texinfo/texinfo-6.7.tar.xz*
Somme de contrôle MD5 : `d4c5d8cc84438c5993ec5163a59522a6`

• Time Zone Data (2021a) — 403 Ko :
Page d'accueil : *https://www.iana.org/time-zones*
Téléchargement : *https://www.iana.org/time-zones/repository/releases/tzdata2021a.tar.gz*
Somme de contrôle MD5 : `20eae7d1da671c6eac56339c8df85bbd`

• Util-linux (2.36.2) — 5,223 Ko:
Page d'accueil : *http://freecode.com/projects/util-linux*
Téléchargement : *https://www.kernel.org/pub/linux/utils/util-linux/v2.36/util-linux-2.36.2.tar.xz*
Somme de contrôle MD5 : `f78419af679ac9678190ad961eb3cf27`

• Vim (8.2.2433) — 15,051 Ko:
Page d'accueil : *https://www.vim.org*
Téléchargement : *http://anduin.linuxfromscratch.org/LFS/vim-8.2.2433.tar.gz*
Somme de contrôle MD5 : `a26555c8919cf40938d2428d834bf913`

Note
La version de vim change tous les jours. Pour récupérer la dernière version, visitez *https://github.com/vim/vim/releases*.

• XML::Parser (2.46) — 249 Ko :
Page d'accueil : *https://github.com/chorny/XML-Parser*
Téléchargement : *https://cpan.metacpan.org/authors/id/T/TO/TODDR/XML-Parser-2.46.tar.gz*
Somme de contrôle MD5 : `80bb18a8e6240fcf7ec2f7b57601c170`

• Xz Utils (5.2.5) — 1,122 Ko:
Page d'accueil : *https://tukaani.org/xz*
Téléchargement : *https://tukaani.org/xz/xz-5.2.5.tar.xz*
Somme de contrôle MD5 : `aa1621ec7013a19abab52a8aff04fe5b`

• Zlib (1.2.11) — 457 Ko:
Page d'accueil : *https://www.zlib.net/*
Téléchargement : *https://zlib.net/zlib-1.2.11.tar.xz*
Somme de contrôle MD5 : `85adef240c5f370b308da8c938951a68`

• Zstd (1.4.8) — 1,762 Ko :
Page d'accueil : *https://facebook.github.io/zstd/*
Téléchargement : *https://github.com/facebook/zstd/releases/download/v1.4.8/zstd-1.4.8.tar.gz*
Somme de contrôle MD5 : `e873db7cfa5ef05832e6d55a5a572840`

Taille totale de ces paquets : environ 428 Mo

3.3. Correctifs requis

En plus des paquets, quelques correctifs sont aussi requis. Ces correctifs corrigent certaines erreurs contenues dans les paquets, ces erreurs devraient être corrigées par le mainteneur. Les correctifs font aussi quelques modifications pour faciliter l'utilisation des paquets. Les correctifs suivants seront nécessaires pour construire un système LFS :

• Bzip2 Correctif documentation — 1.6 Ko :
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/10.1/bzip2-1.0.8-install_docs-1.patch*
Somme de contrôle MD5 : `6a5ac7e89b791aae556de0f745916f7f`

• Coreutils Correctif pour l'internationalisation — 166 Ko:
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/10.1/coreutils-8.32-i18n-1.patch*
Somme de contrôle MD5 : `cd8ebed2a67fff2e231026df91af6776`

• Glibc correctif FHS — 2.8 Ko :
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/10.1/glibc-2.33-fhs-1.patch*
Somme de contrôle MD5 : `9a5997c3452909b1769918c759eff8a2`

• Correctif réparant Kdb Backspace/Delete — 12 Ko :
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/10.1/kbd-2.4.0-backspace-1.patch*
Somme de contrôle MD5 : `f75cca16a38da6caa7d52151f7136895`

• Corrections en amont de Systemd — 8 Ko :
Téléchargement : *http://www.linuxfromscratch.org/patches/lfs/10.1/systemd-247-upstream_fixes-1.patch*
Somme de contrôle MD5 : `573a166f3620875208f78bf4dc431c8b`

Taille totale de ces correctifs : environ 190.4 Ko

En plus des correctifs requis ci-dessus, il existe un certain nombre de correctifs optionnels créés par la communauté LFS. Ces correctifs résolvent des problèmes mineurs ou activent des fonctionnalités qui ne sont pas disponibles par défaut. Vous pouvez consulter la base de données des correctifs à loisir sur *http://www.linuxfromscratch.org/patches/downloads/* et vous pouvez récupérer tout correctif supplémentaire correspondant aux besoins de votre système.

Chapitre 4. Dernières préparations

4.1. Introduction

Dans ce chapitre, nous allons effectuer quelques tâches supplémentaires pour préparer la construction du système temporaire. Nous allons créer un ensemble de répertoires dans `$LFS` pour l'installation des outils temporaires, ajouter un utilisateur non privilégié pour réduire les risques, et créer un environnement de construction adéquat pour cet utilisateur. Nous allons également expliquer l'unité de temps utilisée pour mesurer la durée de construction des paquets LFS, ou « SBU », et donner quelques informations sur les suites de tests des paquets.

4.2. Créer un ensemble limité de répertoire dans le système de fichiers LFS

La première chose à faire dans la partition LFS est de créer une hiérarchie de répertoires limitée pour que les programmes compilés dans le Chapitre 6 puissent être installés à leur emplacement final. Cela est requis pour que ces programmes temporaires puissent être remplacés lorsqu'on les reconstruira dans le Chapitre 8.

Créez la disposition requise des répertoires en lançant ce qui suit en tant que `root` :

```
mkdir -pv $LFS/{bin,etc,lib,sbin,usr,var}
case $(uname -m) in
  x86_64) mkdir -pv $LFS/lib64 ;;
esac
```

Les programmes du Chapitre 6 seront compilés avec un compilateur croisé (plus de détails dans la section Notes techniques sur la chaîne d'outils). Pour séparer ce compilateur croisé des autres programmes, il sera installé dans un répertoire spécial. Créez ce répertoire avec :

```
mkdir -pv $LFS/tools
```

4.3. Ajouter l'utilisateur LFS

Lorsque vous êtes connecté en tant qu'utilisateur `root`, faire une simple erreur peut endommager voire dévaster votre système. Donc, les paquets dans les deux prochains chapitres sont construits en tant qu'utilisateur non privilégié. Vous pouvez bien sûr utiliser votre propre nom d'utilisateur mais, pour faciliter l'établissement d'un environnement de travail propre, créez un nouvel utilisateur `lfs` comme membre d'un nouveau groupe (aussi nommé `lfs`) et utilisez cet utilisateur lors du processus d'installation. En tant que `root`, lancez les commandes suivantes pour créer le nouvel utilisateur :

```
groupadd lfs
useradd -s /bin/bash -g lfs -m -k /dev/null lfs
```

Voici la signification des options de la ligne de commande :

`-s /bin/bash`
Ceci fait de **bash** le shell par défaut de l'utilisateur `lfs`.

`-g lfs`
Cette option ajoute l'utilisateur `lfs` au groupe `lfs`.

`-m`
Ceci crée un répertoire personnel pour l'utilisateur `lfs`.

-k /dev/null
Ce paramètre empêche toute copie possible de fichiers provenant du répertoire squelette (par défaut, `/etc/skel`) en modifiant son emplacement par le périphérique spécial null.

lfs
Ceci est le nom réel pour l'utilisateur créé.

Pour vous connecter en tant qu'utilisateur `lfs` (et non pas de passer à l'utilisateur `lfs` alors que vous êtes connecté en tant que `root`, ce qui ne requiert pas de mot de passe pour l'utilisateur `lfs`), donnez un mot de passe à `lfs` :

```
passwd lfs
```

Donnez à `lfs` un accès complet aux répertoires de `$LFS` en indiquant que `lfs` est le propriétaire du répertoire :

```
chown -v lfs $LFS/{usr,lib,var,etc,bin,sbin,tools}
case $(uname -m) in
  x86_64) chown -v lfs $LFS/lib64 ;;
esac
```

Si un répertoire de travail séparé a été créé comme suggéré, faites en sorte que l'utilisateur `lfs` soit aussi le propriétaire de ce répertoire :

```
chown -v lfs $LFS/sources
```

Note

Sur certains systèmes hôtes, la commande suivante ne termine pas correctement et place la connexion vers l'utilisateur lfs en tâche de fond. Si l'invite de commande « lfs:~$ » n'apparaît pas immédiatement, saisissez la commande **fg** pour corriger le problème.

Ensuite, connectez-vous en tant que `lfs`. Ceci peut se faire via une console virtuelle, avec le gestionnaire d'affichage ou avec la commande suivante de substitution d'utilisateur :

```
su - lfs
```

Le « `-` » indique à **su** de lancer un shell de connexion. Vous trouverez la différence entre un shell de connexion et un autre dans la page de manuel `bash(1)` et **info bash**.

4.4. Configurer l'environnement

Configurez un bon environnement de travail en créant deux nouveaux fichiers de démarrage pour le shell **bash**. En étant connecté en tant qu'utilisateur `lfs`, lancez la commande suivante pour créer un nouveau `.bash_profile` :

```
cat > ~/.bash_profile << "EOF"
exec env -i HOME=$HOME TERM=$TERM PS1='\u:\w\$ ' /bin/bash
EOF
```

Lorsque vous êtes connecté en tant que `lfs`, le shell initial est habituellement un shell de *login* qui lit le fichier `/etc/profile` de l'hôte (contenant probablement quelques configurations et variables d'environnement) et puis `.bash_profile`. La commande **exec env -i.../bin/bash** dans le fichier `.bash_profile` remplace le shell en cours avec un nouveau ayant un environnement complètement vide sauf pour les variables `HOME`, `TERM`, et `PS1`. Ceci nous assure qu'aucune variable d'environnement non souhaitée et potentiellement dangereuse, provenant du système hôte, ne parvienne dans l'environnement de construction. La technique utilisée ici s'assure de créer un environnement propre.

La nouvelle instance du shell est un shell *non-login*, qui ne lit donc pas, et n'exécute pas, les fichiers `/etc/profile` ou `.bash_profile`, mais plutôt le fichier `.bashrc`. Créez maintenant le fichier `.bashrc` :

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
LFS_TGT=$(uname -m)-lfs-linux-gnu
PATH=/usr/bin
if [ ! -L /bin ]; then PATH=/bin:$PATH; fi
PATH=$LFS/tools/bin:$PATH
CONFIG_SITE=$LFS/usr/share/config.site
export LFS LC_ALL LFS_TGT PATH CONFIG_SITE
EOF
```

Voici la signification des paramètres dans `.bashrc`

`set +h`

La commande **set +h** désactive la fonction de hachage de **bash**. D'habitude, le hachage est une fonctionnalité utile—**bash** utilise une table de hachage pour se rappeler le chemin complet des fichiers exécutables pour éviter d'avoir à chercher dans `PATH` à chaque fois qu'il doit trouver le même exécutable. Néanmoins, les nouveaux outils devraient être utilisés dès leur installation. En désactivant la fonction de hachage, le shell cherchera en permanence dans `PATH` lorsqu'un programme doit être exécuté. Ainsi, le shell trouvera les nouveaux outils compilés dans `$LFS/tools` dès qu'ils sont disponibles et sans se rappeler la version précédente du même programme mais dans un autre emplacement.

`umask 022`

Configurer le masque de création de fichier (umask) à 022 nous assure que les nouveaux fichiers et répertoires créés sont modifiables uniquement par leurs propriétaires mais lisibles et exécutables par tout le monde (en supposant que l'appel système `open(2)` utilise les modes par défaut, les nouveaux fichiers auront les droits 644 et les répertoires 755).

`LFS=/mnt/lfs`

La variable `LFS` devrait être configurée avec le point de montage choisi.

`LC_ALL=POSIX`

La variable `LC_ALL` contrôle les paramètres linguistiques de certains programmes, faisant que leurs messages suivent les conventions d'un pays spécifié. Définir `LC_ALL` à « POSIX » ou « C » (les deux étant équivalents) garantit que tout fonctionnera comme prévu dans l'environnement chroot.

`LFS_TGT=(uname -m)-lfs-linux-gnu`

La variable `LFS_TGT` initialise une description de la machine personnalisée mais compatible lors de la construction de notre compilateur, de notre éditeur de liens croisés et lors de la compilation de notre chaîne d'outils temporaires. Vous trouverez plus d'informations dans les Notes techniques sur la chaîne d'outils.

`PATH=/usr/bin`

De nombreuses distributions modernes ont fusionné `/bin` et `/usr/bin`. Lorsque c'est le cas, la variable `PATH` standard n'a besoin que d'indiquer `/usr/bin` pour l'environnement ue Chapitre 6. Lorsque ce n'est pas le cas, la ligne suivante ajoute `/bin` au chemin de recherche.

`if [ ! -L /bin ]; then PATH=/bin:$PATH; fi`

Si `/bin` n'est pas un lien symbolique, il doit être ajouté à la variable `PATH`.

`PATH=$LFS/tools/bin:$PATH`

En plaçant `$LFS/tools/bin` au début du `PATH` standard, le compilateur croisé installé au début du Chapitre 5 est repéré par le shell immédiatement après son installation. Ceci, combiné à la désactivation du hachage, limite le risque que le compilateur de l'hôte ne soit utilisé à la place du compilateur croisé.

`CONFIG_SITE=$LFS/usr/share/config.site`

Dans le Chapitre 5 et le Chapitre 6, si cette variable n'est pas initialisée, les scripts **configure** peuvent essayer de charger des bouts de configuration de certaines distributions dans `/usr/share/config.site` sur le système hôte. Changer ce chemin permet d'éviter une contamination potentielle par l'hôte.

`export ...`

Bien que les commandes précédentes aient configurées certaines variables, pour les rendre visibles à des sous-shell, nous les exportons.

Important

Plusieurs distributions commerciales ajoutent une instance non documentée de `/etc/bash.bashrc` à l'initialisation de **bash**. Ce fichier peut modifier l'environnement de l'utilisateur `lfs` d'une manière qui peut affecter la construction des paquets critiques de LFS. Pour vous assurer que l'environnement de l'utilisateur `lfs` est propre, vérifiez la présence de `/etc/bash.bashrc` et, s'il est présent, déplacez-le ailleurs. En tant qu'utilisateur `root`, lancez :

```
[ ! -e /etc/bash.bashrc ] || mv -v /etc/bash.bashrc /etc/bash.bashrc.NOUSE
```

Après avoir fini d'utiliser l'utilisateur `lfs` à la fin du Chapitre 7, vous pouvez restaurer `/etc/bash.bashrc` (si vous le souhaitez).

Remarquez que le paquet Bash de LFS que nous construisons dans la Section 8.34, « Bash-5.1 » n'est pas configuré pour charger ou exécuter `/etc/bash.bashrc`, donc ce fichier est inutile sur un système LFS.

Enfin, pour avoir un environnement complètement préparé pour la construction des outils temporaires, chargez le profil de l'utilisateur tout juste créé :

```
source ~/.bash_profile
```

4.5. À propos des SBU

Beaucoup de personnes souhaitent savoir combien de temps la compilation et l'installation de chaque paquet va prendre. Mais Linux From Scratch est construit sur tant de systèmes différents qu'il est impossible de donner des temps précis. Le plus gros paquet (Glibc) prendra approximativement vingt minutes sur les systèmes les plus rapides mais pourrait prendre environ trois jours sur les moins rapides ! Au lieu de donner les temps constatés, l'unité de construction standard (SBU) est utilisée.

La mesure de SBU fonctionne ainsi : le premier paquet que vous compilez dans ce livre est Binutils lors du Chapitre 5. Le temps que prend la compilation de ce paquet est ce que nous appelons un SBU. Tous les autres temps de compilation sont exprimés par rapport à celui-ci.

Par exemple, considérez un paquet spécifique dont le temps de compilation correspond à 4,5 SBU. Ceci signifie que s'il vous a fallu 10 minutes pour compiler et installer la première passe de Binutils, alors vous savez que cela prendra *environ* 45 minutes pour construire ce paquet. Heureusement, la plupart des temps de construction sont bien plus courts que celui de Binutils.

En général, les SBU ne sont pas vraiment précis, car ils dépendent de trop de facteurs, dont la version de GCC sur votre machine hôte. Ils sont fournis ici pour donner une estimation du temps nécessaire pour installer un paquet mais ces nombres peuvent varier de plusieurs dizaines de minutes dans certains cas.

Note

Pour beaucoup de systèmes modernes avec plusieurs processeurs (ou cœurs), le temps de compilation d'un paquet peut être réduit en effectuant un « make parallèle », soit en réglant une variable d'environnement, soit en disant au programme **make** combien de processeurs sont disponibles. Par exemple, un CPU Intel i5-6500 peut prendre en charge quatre processus simultanément avec :

```
export MAKEFLAGS='-j4'
```

ou simplement en construisant avec :

```
make -j4
```

Si vous utilisez plusieurs processeurs de cette façon, les unités de SBU du livre vont varier encore plus que la normale. L'analyse de la sortie du processus de construction sera aussi plus difficile, car les lignes des différents processus seront mélangées. Si vous rencontrez un problème à une étape de la construction, revenez à une construction avec un seul processeur pour analyser correctement les messages d'erreur.

4.6. À propos des suites de tests

La plupart des paquets disposent d'une suite de tests. Lancer cette suite de tests pour un paquet nouvellement construit est généralement une bonne idée car cela peut apporter une « vérification de propreté » comme quoi tout a été compilé correctement. Une suite de tests réussissant l'ensemble des vérifications prouve généralement que le paquet fonctionne à peu près comme le développeur en avait l'intention. Néanmoins, cela ne garantit pas que le paquet ne contient pas de bogues.

Certaines des suites de tests sont plus importantes que d'autres. Par exemple, les suites de tests des paquets formant le cœur de l'ensemble des outils — GCC, Binutils, et Glibc — sont de la plus grande importance étant donné leur rôle central dans un système fonctionnel. Les suites de tests pour GCC et Glibc peuvent prendre beaucoup de temps pour terminer, surtout sur du matériel lent, mais elles sont fortement recommandées.

Note

Lancer les suites de tests dans le Chapitre 5 et le Chapitre 6 est impossible, parce que les programmes sont compilés avec un compilateur croisé, donc ils ne sont pas sensés être exécutables sur l'hôte de construction.

Un problème commun lors du lancement des suites de test pour Binutils et GCC est de manquer de pseudo-terminaux (PTY). Cela peut causer un nombre inhabituellement élevé d'échecs dans les tests. Ceci peut arriver pour un certain nombre de raisons, mais la plus probable est que le système hôte ne dispose pas d'un système de fichiers `devpts` configuré correctement. Ce problème est traité avec beaucoup plus de détails dans *http://fr.linuxfromscratch.org/faq/lfs#no-ptys*.

Quelquefois, les suites de test des paquets échoueront mais pour des raisons dont les développeurs sont conscients et qu'ils ont estimées non critique. Consultez les traces sur *http://www.linuxfromscratch.org/lfs/build-logs/10.1/* pour vérifier si ces échecs sont attendus. Ce site est valide pour tous les tests effectués dans ce livre.

Partie III. Construction des outils croisés LFS et des outils temporaires

Informations préliminaires importantes

Introduction

Cette partie est divisée en trois étapes : la première construit un compilateur croisé et ses bibliothèques associées ; la seconde utilise cette chaîne d'outils croisée pour construire plusieurs outils d'une façon qui les isole de la distribution hôte ; la troisième entre dans l'environnement chroot, qui améliore encore plus l'isolation, et construit le reste des outils requis pour construire le système final.

Important

Dans cette partie commence le vrai travail de construction d'un nouveau système. Elle demande beaucoup d'attention pour suivre avec précision les instructions que propose le livre. Vous devriez essayer de comprendre ce qu'elles font, et malgré votre empressement à finir la construction, vous devriez éviter de taper aveuglément les commandes montrées, mais plutôt lire la documentation quand vous ne comprenez pas quelque chose. Gardez aussi une trace de ce que vous tapez et des sorties des commandes, en les envoyant dans un fichier, avec l'outil **tee**. Cela permet de mieux diagnostiquer les problèmes s'il en apparaît.

La prochaine section est une introduction technique au processus de construction, tandis que la suivante contient des instructions générales **très importantes**.

Notes techniques sur la chaîne d'outils

Cette section explique certains détails rationnels et techniques derrière la méthode de construction. Il n'est pas essentiel de comprendre immédiatement tout ce qui se trouve dans cette section. La plupart des informations seront plus claires après avoir réalisé réellement une construction complète. Cette section peut servir de référence à tout moment lors du processus de construction.

Le but global des chapitres Chapitre 5 et Chapitre 6 est de fournir une zone temporaire qui contient un ensemble d'outils connus qui peuvent être isolés du système hôte. En utilisant **chroot**, les commandes dans le reste des chapitres se cantonneront à cet environnement, en assurant une construction du système LFS cible propre, sans soucis. Le processus de construction a été conçu pour minimiser les risques pour les nouveaux lecteurs et pour fournir une valeur éducative maximale en même temps.

Le processus de construction se base sur de la *compilation croisée*. La compilation croisée s'utilise normalement pour construire un compilateur et sa chaîne de construction pour une machine différente de celle utilisée pour la construction. Cela n'est pas strictement requis pour LFS, comme la machine où le nouveau système est construit est la même que celle utilisée pour la construction. Mais la compilation croisée a le grand avantage que tout ce qui est compilé ne peut pas dépendre de l'environnement hôte.

À propos de la compilation croisée

La compilation croisée utilise certains concepts qui méritent une section à part. Bien que vous puissiez passer cette section lors de votre première lecture, nous vous recommandons fortement d'y revenir plus tard pour bien comprendre le processus de construction.

Définissons d'abord certains termes utilisés dans ce contexte :

build (construction)
: est la machine où nous construisons les programmes. Remarquez que cette machine sera appelée « hôte » dans les autres sections.

host (hôte)
: est la machine ou le système où les programmes seront lancés. Remarquez que nous n'utilisons pas le terme « hôte » de la même manière ici que dans les autres sections.

target (cible)
: est seulement utilisé pour les compilateurs. C'est la machine pour laquelle le compilateur produit du code. Elle peut être différente de la machine hôte ou de construction.

Par exemple, imaginons le scénario suivant (parfois appelé « Canadian Cross ») : on peut avoir un compilateur sur une machine lente, appelons-la A, et le compilateur ccA. On peut aussi avoir une machine rapide (B) sans compilateur, et on veut produire du code pour une autre machine lente (C). Pour construire un compilateur pour une machine C, on effectuerait trois étapes :

Étape	Construction	Hôte	Cible	Action
1	A	A	B	construire un compilateur croisé cc1 avec ccA sur la machine A
2	A	B	C	construire un compilateur croisé cc2 avec cc1 sur la machine A
3	B	C	C	construire le compilateur ccC avec cc2 sur la machine B

Ensuite, tous les autres programmes requis par la machine C peuvent être compilés avec cc2 sur la machine rapide B. Remarquez qu'à moins que B ne puisse lancer les programmes produits pour C, il n'y a aucun moyen de tester les programmes construits avant de les lancer sur la machine C. Par exemple, pour tester ccC, on peut ajouter une quatrième étape :

Étape	Construction	Hôte	Cible	Action
4	C	C	C	reconstruire et tester ccC avec lui-même sur la machine C

Dans l'exemple au dessus, seuls cc1 et cc2 sont des compilateurs croisés, c'est à dire qu'ils produisent du code pour une machine différente de celle sur laquelle ils tournent. Les autres compilateurs ccA et ccC produisent du code pour la machine sur laquelle ils tournent. Ces compilateurs sont appelés des compilateurs *natifs*.

Implémentation de la compilation croisée dans LFS

Note

Presque tous les systèmes de construction utilisent des noms de la forme cpu-fabriquant-noyau-os, souvent appelé le triplet machine. Le lecteur attentif se demandera pourquoi on appelle un « triplet » un nom à quatre composants. La raison est historique : initialement, on utilisait trois composants pour désigner une machine sans ambiguïté, mais avec les nouvelles machines et les nouveaux systèmes, cela s'avère insuffisant. Le mot « triplet » est resté. Une façon simple de déterminer le nom du triplet machine est de lancer le script **config.guess** venant avec les sources d'un grand nombre de paquets. Déballez les sources de binutils, lancez le script **`./config.guess`** et notez la sortie. Par exemple, pour un processeur Intel 32 bits moderne, la sortie sera du type *i686-pc-linux-gnu*. Sur un système 64 bits cela sera *x86_64-pc-linux-gnu*.

De même, faites attention au nom de l'éditeur de liens de la plateforme, souvent appelé le chargeur dynamique (à ne pas confondre avec l'éditeur de liens standard **ld** faisant partie de Binutils). Le chargeur dynamique fourni par Glibc trouve et charge les bibliothèques partagées nécessaires à un programme pour s'exécuter, puis l'exécute. Le nom de l'éditeur dynamique pour une machine Intel 32 bits sera `ld-linux.so.2` (`ld-linux-x86-64.so.2` pour les systèmes 64 bits). Une façon sûre de déterminer le nom de l'éditeur de liens dynamiques est d'inspecter un binaire au hasard du système hôte en exécutant : **`readelf -l <nom du binaire> | grep interpreter`** et de noter le résultat. La référence faisant autorité couvrant toutes les plateformes est dans le fichier `shlib-versions` à la racine du répertoire des sources de Glibc.

Pour simuler une compilation croisée, le nom du triplet hôte est légèrement ajusté en changeant la partie « fabriquant » dans la variable `LFS_TGT`. Nous utilisons aussi l'option *`--with-sysroot`* lors de la construction de l'éditeur des liens et du compilateur croisés pour leur dire où trouver les fichiers hôtes requis. Cela s'assure qu'aucun autre programme construit dans le Chapitre 6 ne peut se lier aux bibliothèques sur la machine de construction. Seules deux étapes sont requises, et une autre pour les tests :

Étape	Construction	Hôte	Cible	Action
1	pc	pc	lfs	construire un compilateur croisé cc1 avec cc-pc sur pc
2	pc	lfs	lfs	construire un compilateur cc-lfs avec cc1 sur pc
3	lfs	lfs	lfs	reconstruire et tester cc-lfs avec lui-

Étape	Construction	Hôte	Cible	Action
				même sur lfs

Dans le tableau plus haut, « sur pc » signifie que les commandes sont lancées sur une machine qui utilise la distribution déjà installée. « Sur lfs » signifie que les commandes sont lancées dans un environnement chroot.

n fait, il y a plus à savoir sur la compilation croisée : le langage C n'est pas seulement un compilateur, mais définie aussi une bibliothèque standard. Dans ce livre, on utilise la bibliothèque C de GNU, glibc. Cette bibliothèque doit être compilée pour la machine lfs, c'est-à-dire, avec le compilateur croisé cc1. Mais le compilateur lui-même utilise une bibliothèque interne implémentant des instructions complexes qui ne sont pas disponibles dans l'ensemble d'instructions de l'assembleur. Cette bibliothèque interne, libgcc, doit être liée à la bibliothèque glibc pour fonctionner correctement ! En plus, la bibliothèque standard du C++ (libstdc++) a aussi besoin d'être liée à la glibc. La solution pour ce problème de poule et d'œuf est de d'abord construire une libgcc dégradée basée sur cc1, qui n'a pas de fonctionnalité avancée comme les threads et le traitement des exceptions, puis de construire glibc avec ce compilateur dégradé (glibc elle-même n'est pas dégradée), puis de construire libstdc++. Mais cette dernière n'aura pas les fonctionnalités que libgcc n'a pas non plus.

Ce n'est pas la fin de l'histoire : la conclusion du paragraphe précédent est que cc1 est incapable de trouver une libstdc++ complètement fonctionnelle, mais c'est le seul compilateur disponible pour construire les bibliothèques C/C++ lors de la deuxième étape ! Évidemment, le compilateur construit à l'étape 2, cc-lfs, serait capable de construire ces bibliothèques, mais (1) le système de construction de GCC ne sait pas qu'il est utilisable sur pc, et (2) l'utiliser sur pc risquerait de le lier à des bibliothèques de pc, comme cc-lfs est un compilateur natif. Donc nous devons compiler libstdc++ plus tard, dans le chroot.

Autres détails sur la procédure

Le compilateur croisé sera installé dans un répertoire `$LFS/tools` séparé, comme il ne fera pas partie du système final.

Binutils est tout d'abord installé parce que les exécutions de Glibc et GCC par **configure** réalisent quelques tests de fonctionnalités sur l'assembleur et l'éditeur de liens pour déterminer quelle fonctionnalité logicielle activer ou désactiver. Ceci est plus important que ce que vous pouvez imaginer. Un GCC ou une Glibc mal configuré peut aboutir à une chaîne d'outils subtilement cassée, et l'impact d'une telle cassure ne se verrait pas avant la fin de la construction de la distribution complète. Un échec dans la suite de tests surlignera habituellement cette erreur avant que trop de travail supplémentaire n'ait été réalisé.

Binutils installe son assembleur et son éditeur de liens à deux endroits, `$LFS/tools/bin` et `$LFS/tools/$LFS_TGT/bin`. Les outils dans un emplacement sont liés en dur à l'autre. Un aspect important de l'éditeur de liens est son ordre de recherche des bibliothèques. Vous pouvez obtenir des informations détaillées à partir de **ld** en lui passant le paramètre *`--verbose`*. Par exemple, un **ld --verbose | grep SEARCH** illustrera les chemins de recherche réels et leur ordre. Il montre quels fichiers sont liés par **ld** en compilant un programme de test et en passant le paramètre *`--verbose`* à l'éditeur de liens. Par exemple, **$LFS_TGT-gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded** affichera tous les fichiers ouverts avec succès lors de l'édition des liens.

Le prochain paquet installé est GCC. Voici un exemple de ce qui peut être vu pendant l'exécution de son script **configure** :

```
checking what assembler to use... /mnt/lfs/tools/i686-lfs-linux-gnu/bin/as
checking what linker to use... /mnt/lfs/tools/i686-lfs-linux-gnu/bin/ld
```

C'est important pour les raisons mentionnées plus haut. Cela démontre aussi que le script configure de GCC ne cherche pas les répertoires du PATH pour trouver les outils à utiliser. Néanmoins, lors d'une opération normale de **gcc**, les mêmes chemins de recherche ne sont pas forcément utilisés. Pour trouver quel éditeur de liens standard **gcc** utilisera, lancez : **$LFS_TGT-gcc -print-prog-name=ld**.

Vous pouvez obtenir des informations détaillées à partir de **gcc** en lui fournissant l'option en ligne de commande *-v* lors de la compilation d'un programme de test. Par exemple, **gcc -v dummy.c** affichera des informations détaillées sur les étapes du préprocesseur, de la compilation et de l'assemblage, avec les chemins de recherche inclus par **gcc** et leur ordre.

Nous installons ensuite les en-têtes de l'API de Linux nettoyées. Elles permettent à la bibliothèque standard (Glibc) d'interagir avec les fonctionnalités que le noyau Linux fournira.

Le paquet installé ensuite est Glibc. Les choses les plus importantes à prendre en considération pour construire Glibc sont le compilateur, les outils binaires et les en-têtes du noyau. Le compilateur ne pose généralement pas de problème car Glibc utilise toujours le compilateur lié au paramètre *--host* passé à son script configure, par exemple, dans notre cas, le compilateur sera **$LFS_TGT-gcc**. Les outils binaires et les en-têtes du noyau peuvent être un peu plus compliqués. Ne prenez donc pas de risque et utilisez les options de configure disponibles pour assurer les bonnes sélections. Après l'exécution de **configure**, vérifiez le contenu du fichier `config.make` dans le répertoire `glibc-build` pour tous les détails importants. Notez l'utilisation de *CC="$LFS_TGT-gcc"* (où `$LFS_TGT` est étendue) pour contrôler les outils binaires utilisés, et l'utilisation des paramètres *-nostdinc* et *-isystem* pour contrôler le chemin de recherche des en-têtes du compilateur. Ces éléments soulignent un aspect important du paquet glibc — il est auto-suffisant en termes de machinerie de construction et ne repose généralement pas sur la chaîne d'outils par défaut.

Comme nous venons de le dire, la bibliothèque standard C++ est ensuite compilée, suivi dans le Chapitre 6 par tous les programmes qui ont besoin d'eux-mêmes pour être construits. L'étape initiale de tous ces paquets utilise la variable `DESTDIR` pour que les programmes soient installés dans le système de fichiers LFS.

À la fin du Chapitre 6 le compilateur lfs natif est installé. binutils-pass2 est d'abord construit, avec la même installation `DESTDIR` que les autres programmes, puis la seconde passe de GCC est construite, sans libstdc++ et les autres bibliothèques non importantes. À cause d'une logique bizarre dans le script de construction de GCC, `CC_FOR_TARGET` devient **cc** quand l'hôte est le même que la cible, mais est différent du système de construction. C'est pourquoi nous mettons explicitement *CC_FOR_TARGET=$LFS_TGT-gcc* dans les options de configure.

En entrant dans l'environnement chroot dans le Chapitre 7, la première tache consiste à installer libstdc++. Ensuite, on effectue des installations temporaires de programmes requis pour le bon fonctionnement de la chaîne d'outils. À partir de là, la chaîne de construction de base est auto-suffisante et auto-hébergée. Dans le Chapitre 8, on construit, teste et installe les versions finales de tous les paquets requis pour un système complètement fonctionnel.

Instructions générales de compilation

Lorsque vous construisez des paquets, il y a plusieurs présupposés dans les instructions :

- Plusieurs paquets sont corrigés avant d'être compilés, mais seulement dans le cas où la correction est nécessaire pour résoudre un problème. Souvent, le correctif est nécessaire à la fois dans ce chapitre et dans les suivants, mais quelques fois dans un seul chapitre. Donc, ne vous inquiétez pas lorsque des instructions pour un correctif téléchargé semblent manquer. Des messages d'avertissements sur un décalage (*offset*) ou sur autre chose (*fuzz*) peuvent apparaître lors de l'application d'un correctif. Ne vous inquiétez pas pour ces messages, le correctif a bien été appliqué.
- Pendant la compilation de la plupart des paquets, plusieurs messages d'avertissement du compilateur défileront sur votre écran. Ceci est normal et peut être ignoré sans danger. Ces messages d'avertissement ne sont que des avertissements— sur une utilisation obsolète, mais pas invalide, de la syntaxe de C ou de C++. Les standards C changent assez souvent et quelques paquets continuent à utiliser les anciens standards. Ce n'est pas un véritable problème mais cela provoque les messages.
- Vérifiez une dernière fois que la variable d'environnement `LFS` est configurée correctement :

```
echo $LFS
```

Assurez-vous que le résultat contient le bon répertoire vers le point de montage de la partition LFS, qui est `/mnt/lfs`, suivant notre exemple.

- Enfin, deux points importants doivent être précisés :

Important

Les instructions de construction supposent que vous avez défini correctement les Prérequis du système hôte, y compris les liens symboliques :

- **bash** est le shell utilisé.
- **sh** est un lien symbolique vers **bash**.
- **/usr/bin/awk** est un lien symbolique vers **gawk**.
- **/usr/bin/yacc** est un lien symbolique vers **bison** ou un petit script qui exécute bison.

Important

Pour remettre en évidence la procédure de construction :

1. Mettez tous les codes sources et les correctifs dans un répertoire qui sera accessible depuis l'environnement chroot, tel que `/mnt/lfs/sources/`.
2. Allez dans le répertoire des codes sources.
3. Pour chaque paquet :
   a. En utilisant le programme **tar**, décompressez le paquet à construire. Dans les chapitres Chapitre 5 et Chapitre 6, assurez-vous d'être l'utilisateur *lfs* lors de l'extraction du paquet.
   b. Allez dans le répertoire créé lorsque le paquet a été décompressé.
   c. Suivez les instructions du livre pour construire le paquet.
   d. Revenez au répertoire des codes sources.
   e. Supprimez le répertoire source que vous avez extrait sauf instruction contraire.

Chapitre 5. Compilation d'une chaîne d'outils croisée

5.1. Introduction

Ce chapitre montre comment construire un compilateur croisé et ses outils associés. Bien qu'ici la compilation croisée soit fausse, le principe est le même que pour une vraie compilation croisée.

Les programmes compilés dans ce chapitre vont être installés dans le répertoire `$LFS/tools` de façon à les garder séparés des fichiers installés dans les chapitres suivants. Les bibliothèques en revanche, sont installées à leur emplacement final, comme elles appartiennent au système que nous voulons construire.

5.2. Binutils-2.36.1 — Passe 1

Le paquet Binutils contient un éditeur de liens, un assembleur et d'autres outils pour gérer des fichiers objets.

Temps de construction approximatif: 1 SBU
Espace disque requis: 640 Mo

5.2.1. Installation de Binutils croisé

Note

Revenez en arrière et relisez les remarques de la section Instructions générales de compilation. La compréhension des remarques notées importantes vous évitera beaucoup de problèmes plus tard.

Il est important que Binutils soit le premier paquet compilé parce que Glibc et GCC réalisent différents tests sur l'éditeur de liens et l'assembleur disponibles pour déterminer leurs propres fonctionnalités à activer.

La documentation de Binutils recommande de construire Binutils dans un répertoire de construction dédié :

```
mkdir -v build
cd build
```

Note

Pour que les valeurs SBU listées dans le reste du livre vous soient utiles, mesurez le temps pris pour construire ce paquet, de la configuration jusqu'à la première installation. Pour cela, englobez les commandes dans une commande **time** de cette façon : **`time { ../configure ... && make && make install; }`**.

Maintenant, préparez la compilation de Binutils :

```
../configure --prefix=$LFS/tools       \
             --with-sysroot=$LFS        \
             --target=$LFS_TGT          \
             --disable-nls              \
             --disable-werror
```

Voici la signification des options de configure :

`--prefix=$LFS/tools`
Ceci dit au script configure de se préparer à installer les programmes de binutils dans le répertoire `$LFS/tools`.

`--with-sysroot=$LFS`
Pour de la compilation croisée, ceci dit au système de construction de chercher dans $LFS les bibliothèques système cibles comme nécessaire.

`--target=$LFS_TGT`
Vu que la description de la machine dans la variable `LFS_TGT` est légèrement différente de la valeur renvoyée par le script **config.guess**, ce paramètre va dire au script **configure** d'ajuster le système de construction de binutils pour la construction d'un éditeur de lien croisé.

`--disable-nls`
Ceci désactive l'internationalisation (i18n), car ce n'est pas nécessaire pour des outils temporaires.

`--disable-werror`
Ceci empêche la compilation de s'arrêter lorsqu'interviennent des événements comme des avertissements du compilateur du système hôte.

Continuez avec la compilation du paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails sur ce paquet sont disponibles dans Section 8.18.2, « Contenu de Binutils. »

5.3. GCC-10.2.0 — Passe 1

Le paquet GCC contient la collection de compilateurs GNU, qui inclut les compilateurs C et C++.

Temps de construction approximatif: 11 SBU

Espace disque requis: 3.7 Go

5.3.1. Installation de GCC croisé

GCC requiert les paquets GMP, MPFR et MPC. Comme il se peut que ces paquets ne soient pas inclus dans votre distribution hôte, ils vont être compilés en même temps que GCC. Déballez chaque paquet dans le répertoire des sources de GCC et renommez les répertoires ainsi créés pour que les procédures de construction de GCC les utilisent automatiquement :

Note

Beaucoup d'incompréhensions existent concernant ce chapitre. Les procédures sont les mêmes que celles des autres chapitres, comme expliqué plus haut (Instructions de compilation des paquets). Extrayez d'abord l'archive tar de gcc du répertoire des sources puis rendez-vous dans le répertoire créé. C'est seulement là que vous devriez suivre les instructions ci-dessous.

```
tar -xf ../mpfr-4.1.0.tar.xz
mv -v mpfr-4.1.0 mpfr
tar -xf ../gmp-6.2.1.tar.xz
mv -v gmp-6.2.1 gmp
tar -xf ../mpc-1.2.1.tar.gz
mv -v mpc-1.2.1 mpc
```

Sur les systèmes x86_64, définissez « lib » comme nom de répertoire par défaut pour les bibliothèques 64 bits :

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
 ;;
esac
```

La documentation de GCC recommande de construire GCC dans un répertoire de construction dédié :

```
mkdir -v build
cd       build
```

Préparez la compilation de GCC :

```
../configure                                       \
    --target=$LFS_TGT                              \
    --prefix=$LFS/tools                            \
    --with-glibc-version=2.11                      \
    --with-sysroot=$LFS                            \
    --with-newlib                                  \
    --without-headers                              \
    --enable-initfini-array                        \
    --disable-nls                                  \
    --disable-shared                               \
    --disable-multilib                             \
    --disable-decimal-float                        \
    --disable-threads                              \
    --disable-libatomic                            \
    --disable-libgomp                              \
    --disable-libquadmath                          \
    --disable-libssp                               \
    --disable-libvtv                               \
    --disable-libstdcxx                            \
    --enable-languages=c,c++
```

Voici la signification des options de configure :

`--with-glibc-version=2.11`
Cette option s'assure que le paquet sera compatible avec la version de glibc de l'hôte. Il est configuré à la version de glibc requise la plus basse spécifiée dans les Prérequis du système hôte.

`--with-newlib`
Vu qu'aucune bibliothèque C fonctionnelle n'est encore disponible, ceci garantit que la constante inhibit_libc soit définie lors de la construction de libgcc. Cela empêche la compilation d'un code exigeant la prise en charge de la libc.

`--without-headers`
Lors de la compilation d'un compilateur croisé complet, GCC exige des en-têtes standards compatibles avec le système cible. Pour nos objectifs, ces en-têtes ne seront pas nécessaires. Ce paramètre empêche GCC de les chercher.

`--enable-initfini-array`
Ce paramètre force l'utilisation de certaines structures de données internes qui sont requises mais ne peuvent pas être détectées lors de la construction d'un compilateur croisé.

`--disable-shared`
Ce paramètre oblige GCC à lier ses bibliothèques internes de manière statique. On procède ainsi parce que les bibliothèques partagées requièrent glibc, qui n'est pas encore installé sur le système cible.

`--disable-multilib`
Sur du x86_64, LFS ne prend pas en charge une configuration multilib. Ce paramètre n'a pas d'importance pour x86.

`--disable-decimal-float, --disable-threads, --disable-libatomic, --disable-libgomp, --disable-libquadmath, --disable-libssp, --disable-libvtv, --disable-libstdcxx`
Ces paramètres désactivent la prise en charge de l'extension pour les nombres décimaux en virgules flottantes, de threading, libatomic, libgomp, libquadmath, libssp, libvtv et de la bibliothèque standard C++. La compilation

de ces fonctions va échouer lors de la construction d'un compilateur croisé et celles-ci sont inutiles pour la compilation croisée de la libc temporaire.

--enable-languages=c,c++

Cette option nous assure que seuls les compilateurs C et C++ seront construits. Ce sont les seuls langages actuellement nécessaires.

Compilez GCC en lançant :

```
make
```

Installez le paquet :

```
make install
```

Cette construction de GCC a installé quelques en-têtes internes au système. Normalement l'un d'entre eux, `limits.h`, incluerait à son tour l'en-tête `limits.h` du système, dans ce cas, `$LFS/usr/include/limits.h`. Cependant, au moment de la construction de GCC `$LFS/usr/include/limits.h` n'existe pas, donc l'en-tête interne qui vient d'être construit est un fichier partiel, auto-contenu et n'inclus pas les fonctionnalités étendues de l'en-tête système. Cela est suffisant pour construire glibc, mais l'en-tête interne complet sera requis plus tard. Créez une version complète de l'en-tête interne avec une commande identique à ce que le système de construction de GCC fait dans des circonstances normales :

```
cd ..
cat gcc/limitx.h gcc/glimits.h gcc/limity.h > \
  `dirname $($LFS_TGT-gcc -print-libgcc-file-name)`/install-tools/include/limits
```

Les détails sur ce paquet sont disponibles dans Section 8.26.2, « Contenu de GCC. »

5.4. Linux-5.10.17 API Headers

Les en-têtes de l'API du noyau Linux (dans linux-5.10.17.tar.xz) expose les API du noyau à Glibc.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 1.1 Go

5.4.1. Installation de Linux API Headers

Le noyau linux a besoin de montrer une interface de programmation de l'application (Application Programming Interface, API) à utiliser (Glibc dans LFS). Cela est possible en nettoyant certains fichiers d'en-tête C qui sont laissés dans le paquet des sources du noyau Linux.

Assurez-vous qu'il n'y a pas de vieux fichiers embarqués dans le paquet :

```
make mrproper
```

Maintenant extrayez les en-têtes publics du noyau depuis les sources. La cible make recommandée « headers_install » ne peut pas être utilisée car elle requiert rsync, qui n'est pas forcément disponible. On place les en-têtes dans `./usr` puis on les copie vers l'emplacement requis.

```
make headers
find usr/include -name '.*' -delete
rm usr/include/Makefile
cp -rv usr/include $LFS/usr
```

5.4.2. Contenu des en-têtes de l'API du noyau Linux

En-têtes installés: /usr/include/asm/*.h, /usr/include/asm-generic/*.h, /usr/include/drm/*.h, /usr/include/linux/*.h, /usr/include/misc/*.h, /usr/include/mtd/*.h, /usr/include/rdma/*.h, /usr/include/scsi/*.h, /usr/include/sound/*.h, /usr/include/video/*.h et /usr/include/xen/*.h
Répertoires installés: /usr/include/asm, /usr/include/asm-generic, /usr/include/drm, /usr/include/linux, /usr/include/misc, /usr/include/mtd, /usr/include/rdma, /usr/include/scsi, /usr/include/sound, /usr/include/video et /usr/include/xen

Descriptions courtes

`/usr/include/asm/*.h`	Les en-têtes ASM de l'API de Linux
`/usr/include/asm-generic/*.h`	Les en-têtes ASM génériques de l'API de Linux
`/usr/include/drm/*.h`	Les en-têtes DRM de l'API de Linux
`/usr/include/linux/*.h`	Les en-têtes linux de l'API de Linux
`/usr/include/misc/*.h`	Des en-têtes diverses de l'API de Linux
`/usr/include/mtd/*.h`	Les en-têtes MTD de l'API de Linux
`/usr/include/rdma/*.h`	Les en-têtes RDMA de l'API de Linux
`/usr/include/scsi/*.h`	Les en-têtes SCSI de l'API de Linux
`/usr/include/sound/*.h`	Les en-têtes son de l'API de Linux
`/usr/include/video/*.h`	Les en-têtes vidéo de l'API de Linux
`/usr/include/xen/*.h`	Les en-têtes Xen de l'API de Linux

5.5. Glibc-2.33

Le paquet Glibc contient la bibliothèque C principale. Cette bibliothèque fournit toutes les routines basiques pour allouer de la mémoire, rechercher des répertoires, ouvrir et fermer des fichiers, les lire et les écrire, gérer les chaînes, faire correspondre des modèles, faire de l'arithmétique et ainsi de suite.

Temps de construction approximatif: 4.2 SBU
Espace disque requis: 770 Mo

5.5.1. Installation de Glibc

Tout d'abord, créez un lien symbolique pour respecter le LSB. En plus, pour x86_64, créez un lien symbolique de compatibilité requis pour le bon fonctionnement du chargeur de bibliothèques dynamiques :

```
case $(uname -m) in
    i?86)   ln -sfv ld-linux.so.2 $LFS/lib/ld-lsb.so.3
    ;;
    x86_64) ln -sfv ../lib/ld-linux-x86-64.so.2 $LFS/lib64
            ln -sfv ../lib/ld-linux-x86-64.so.2 $LFS/lib64/ld-lsb-x86-64.so.3
    ;;
esac
```

Certains programmes de Glibc utilisent le répertoire `/var/db` qui ne respecte pas le FHS pour stocker leurs données d'exécution. Appliquez le correctif suivant pour que ces programmes stockent leurs données d'exécution à l'emplacement indiqué par le FHS :

```
patch -Np1 -i ../glibc-2.33-fhs-1.patch
```

La documentation de Glibc recommande de construire Glibc dans un répertoire dédié :

```
mkdir -v build
cd       build
```

Ensuite, préparez la compilation de Glibc :

```
../configure                             \
      --prefix=/usr                      \
      --host=$LFS_TGT                    \
      --build=$(../scripts/config.guess) \
      --enable-kernel=3.2                \
      --with-headers=$LFS/usr/include    \
      libc_cv_slibdir=/lib
```

Voici la signification des options de configure :

`--host=$LFS_TGT, --build=$(../scripts/config.guess)`
L'effet combiné de ces commutateurs est que le système de construction de Glibc se configure pour se compiler de manière croisée en utilisant l'éditeur de liens croisé et le compilateur croisé dans `$LFS/tools`.

`--enable-kernel=3.2`
Ceci indique à Glibc de compiler la bibliothèque avec la prise en charge des noyaux Linux 3.2 et supérieurs. Les contournements pour les noyaux plus anciens ne sont pas activés.

`--with-headers=$LFS/usr/include`
Ceci dit à Glibc de se compiler contre les en-têtes récemment installés dans le répertoire $LFS/usr/include, afin qu'il connaisse exactement les fonctionnalités du noyau et puisse s'optimiser en conséquence.

`libc_cv_slibdir=/lib`
Cela s'assure que la bibliothèque est installée dans /lib au lieu du répertoire /lib64 par défaut sur les machines 64 bits.

`libc_cv_include_x86_isa_level=no`
Cette option désactive la propriété « x86 ISA needed » dans les bibliothèques Glibc. Utilisez-la **si** vous construisez Glibc avec l'option `-march` dans les `CFLAGS`, pour contourner un problème dans Glibc-2.33 qui la casse.

Lors de cette étape, le message d'avertissement suivant peut apparaître :

```
configure: WARNING:
*** These auxiliary programs are missing or
*** incompatible versions: msgfmt
*** some features will be disabled.
*** Check the INSTALL file for required versions.
```

Le programme **msgfmt**, manquant ou incompatible, ne pose généralement pas de problème. Ce programme **msgfmt** fait partie du paquet Gettext que la distribution hôte devrait fournir.

Note

Il a été reporté que ce paquet pouvait échouer avec un « make parallèles ». Si cela arrive, relancez la commande avec l'option « -j1 ».

Compilez le paquet :

```
make
```

Installez le paquet :

Avertissement

Si `LFS` n'est pas correctement initialisée, et malgré les recommandations, que vous construisez en `root`, la commande suivante installera la glibc nouvellement construite sur votre système hôte, ce qui le rendra inutilisable. Alors assurez-vous que l'environnement est correctement initialisé avant de lancer la commande suivante.

```
make DESTDIR=$LFS install
```

Voici la signification de l'option de make install ::

`DESTDIR=$LFS`
La variable make `DESTDIR` est utilisée par presque tous les paquets pour définir l'emplacement où le paquet devrait être installé. Si elle n'est pas indiquée, elle correspond par défaut à la racine (`/`). Ici, nous spécifions que le paquet doit être installé dans `$LFS`, qui deviendra la racine après Section 7.4, « Entrer dans l'environnement chroot ».

Attention

À ce moment, il est impératif de vous arrêter et de vous assurer que les fonctions de base (compilation et édition des liens) du nouvel ensemble d'outils fonctionnent comme prévu. Pour effectuer un test de propreté, lancez les commandes suivantes :

```
echo 'int main(){}' > dummy.c
$LFS_TGT-gcc dummy.c
readelf -l a.out | grep '/ld-linux'
```

Si tout fonctionne correctement, il ne devrait pas y avoir d'erreurs et la sortie de la dernière commande sera de la forme :

```
[Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
```

Notez que pour les machines 32 bits, le nom de l'interpréteur sera `/lib/ld-linux.so.2`.

Si l'affichage diffère ou s'il n'y a aucun affichage, alors quelque chose ne se passe pas bien. Enquêtez et tracez vos étapes pour trouver où se cache le problème et comment le corriger. Ce problème doit être corrigé avant de continuer.

Une fois que tout va bien, nettoyez les fichiers de test :

```
rm -v dummy.c a.out
```

Note

La construction des paquets dans le prochain chapitre servira de test supplémentaire pour vérifier que l'ensemble d'outils a été construit correctement. Si certains paquets, en particulier binutils-pass2 ou gcc-pass2, échouent à se construire, c'est une indication que quelque chose ne va pas dans les installations précédentes de Binutils, GCC, ou Glibc.

Maintenant que notre chaîne d'outils croisée est complète, finalisez l'installation de l'en-tête limits.h. Pour cela, lancez un utilitaire fournit par les développeurs de GCC :

```
$LFS/tools/libexec/gcc/$LFS_TGT/10.2.0/install-tools/mkheaders
```

Les détails sur ce paquet sont situés dans Section 8.5.3, « Contenu de Glibc. »

5.6. Libstdc++ de GCC-10.2.0, passe 1

Libstdc++ est la bibliothèque standard du C++. Elle est utilisée pour compiler du code C++ (une partie de GCC est écrit en C++) mais nous avons dû retarder son installation lorsqu'on a construit gcc-pass1, car elle dépend de glibc, qui n'était pas encore disponible dans le répertoire cible.

Temps de construction approximatif: 0.4 SBU
Espace disque requis: 953 Mo

5.6.1. Installation de Libstdc++ Cible

Note

Libstdc++ fait partie des sources de GCC. Vous devriez d'abord déballer l'archive tar de GCC et vous rendre dans le répertoire `gcc-10.2.0`.

Créez un répertoire de construction séparé pour libstdc++ et rentrez-y :

```
mkdir -v build
cd build
```

Préparez la compilation de libstdc++ :

```
../libstdc++-v3/configure           \
    --host=$LFS_TGT                 \
    --build=$(../config.guess)      \
    --prefix=/usr                   \
    --disable-multilib              \
    --disable-nls                   \
    --disable-libstdcxx-pch         \
    --with-gxx-include-dir=/tools/$LFS_TGT/include/c++/10.2.0
```

Voici la signification des options de configure :

`--host=...`
Indique d'utiliser le compilateur croisé que nous venons tout juste de construire à la place de celui de `/usr/bin`.

`--disable-libstdcxx-pch`
Ce paramètre empêche l'installation des fichiers inclus pré-compilés, qui ne sont pas nécessaires pour l'instant.

`--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/10.2.0`
C'est l'emplacement où le compilateur C++ recherche les fichiers d'en-tête standards. Dans une construction normale, ces informations sont passées automatiquement aux options **configure** de libstdc++ à partir du premier niveau de répertoire. Dans notre cas, il faut donner explicitement ces informations.

Compilez libstdc++ en lançant :

```
make
```

Installez la bibliothèque :

```
make DESTDIR=$LFS install
```

Les détails sur ce paquet sont situés dans Section 8.26.2, « Contenu de GCC. »

Chapitre 6. Compilation croisée des outils temporaires

6.1. Introduction

Ce chapitre montre comme compiler les utilitaires de base de manière croisée en utilisant la chaîne d'outils croisée qui vient d'être construite. Ces utilitaires sont installés à leur emplacement final, mais ne peuvent pas encore être utilisés. Les tâches de base utilisent toujours les outils de l'hôte. Cependant, les bibliothèques installées sont utilisées à l'édition des liens.

Il sera possible d'utiliser les utilitaires au prochain chapitre après être entré dans l'environnement « chroot ». Mais tous les paquets construits dans le chapitre actuel devront être construits avant de faire cela. Donc, nous ne pouvons pas encore être indépendants du système hôte.

Encore une fois, rappelons qu'une valeur incorrecte de `LFS` et la construction en tant que `root` peuvent rendre votre ordinateur inutilisable. Ce chapitre doit être entièrement effectué en tant qu'utilisateur `lfs`, avec l'environnement décrit dans Section 4.4, « Configurer l'environnement ».

6.2. M4-1.4.18

Le paquet M4 contient un processeur de macros.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 22 Mo

6.2.1. Installation de M4

Tout d'abord, effectuez quelques corrections introduites avec glibc-2.28 :

```
sed -i 's/IO_ftrylockfile/IO_EOF_SEEN/' lib/*.c
echo "#define _IO_IN_BACKUP 0x100" >> lib/stdio-impl.h
```

Préparez la compilation de M4 :

```
./configure --prefix=/usr   \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Les détails de ce paquet se trouvent sur Section 8.12.2, « Contenu de M4. »

6.3. Ncurses-6.2

Le paquet Ncurses contient les bibliothèques de gestion des écrans type caractère, indépendant des terminaux.

Temps de construction approximatif: 0.7 SBU
Espace disque requis: 48 Mo

6.3.1. Installation de Ncurses

Tout d'abord, assurez-vous que **gawk** est trouvé pendant la configuration :

```
sed -i s/mawk// configure
```

Ensuite, lancez les commandes suivantes pour construire le programme « tic » sur l'hôte :

```
mkdir build
pushd build
  ../configure
  make -C include
  make -C progs tic
popd
```

Préparez la compilation de Ncurses :

```
./configure --prefix=/usr                \
            --host=$LFS_TGT              \
            --build=$(./config.guess)    \
            --mandir=/usr/share/man      \
            --with-manpage-format=normal \
            --with-shared                \
            --without-debug              \
            --without-ada                \
            --without-normal             \
            --enable-widec
```

Voici la signification des nouvelles options de configure :

`--with-manpage-format=normal`
Cela évite que Ncurses n'installe les pages de manuel compressées, ce qui peut arriver si la distribution hôte elle-même a des pages de manuel compressées.

`--without-ada`
Cela s'assure que Ncurses ne construise pas la prise en charge du compilateur Ada qui peut être présent sur l'hôte mais qui ne sera pas disponible une fois dans l'environnement **chroot**.

`--enable-widec`
Cette option amène les bibliothèques « wide-character » (comme `libncursesw.so.6.2`) à être compilée au lieu de celles normales (comme `libncurses.so.6.2`). Ces bibliothèques « wide-character » sont utilisables à la fois en locales multibyte et 8-bit traditionnelles, alors que les bibliothèques normales ne fonctionnent correctement que dans les locales 8-bit. Les bibliothèques « Wide-character » et normales sont compatibles entre leurs sources mais pas entre leurs binaires.

`--without-normal`
Ce drapeau désactive la construction et l’installation de la plupart des bibliothèques statiques.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS TIC_PATH=$(pwd)/build/progs/tic install
echo "INPUT(-lncursesw)" > $LFS/usr/lib/libncurses.so
```

Voici la signification des options d'installation :

`TIC_PATH=$(pwd)/build/progs/tic`
Nous devons passer le chemin de **tic** tout juste construit qui peut être lancé sur la machine de construction, pour pouvoir créer la base de données de terminaux sans erreur.

echo "INPUT(-lncursesw)" > $LFS/usr/lib/libncurses.so
La bibliothèque `libncurses.so` est requise par quelques paquets que nous allons bientôt construire. Nous créons ce petit script de liaison, comme cela est fait dans Chapitre 8.

Déplacez les bibliothèques partagées dans le répertoire `/lib`, où elles sont supposées être :

```
mv -v $LFS/usr/lib/libncursesw.so.6* $LFS/lib
```

Comme les bibliothèques ont été déplacées, un lien symbolique pointe vers un fichier inexistant. Recréez-le :

```
ln -sfv ../../lib/$(readlink $LFS/usr/lib/libncursesw.so) $LFS/usr/lib/libncurse
```

Les détails de ce paquet se trouvent sur Section 8.28.2, « Contenu de Ncurses. »

6.4. Bash-5.1

Le paquet Bash contient le shell Bourne-Again.

Temps de construction approximatif:	0.4 SBU
Espace disque requis:	66 Mo

6.4.1. Installation de Bash

Préparez la compilation de Bash :

```
./configure --prefix=/usr                      \
            --build=$(support/config.guess) \
            --host=$LFS_TGT                    \
            --without-bash-malloc
```

Voici la signification des options de configure :

`--without-bash-malloc`
Cette option désactive l'utilisation de la fonction d'allocation de la mémoire de Bash (`malloc`) connue pour causer des erreurs de segmentation. En désactivant cette option, Bash utilisera la fonction `malloc` de Glibc qui est plus stable.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Déplacez les exécutables à l'emplacement attendu :

```
mv $LFS/usr/bin/bash $LFS/bin/bash
```

Créez un lien pour les programmes qui utilisent **sh** comme shell :

```
ln -sv bash $LFS/bin/sh
```

Les détails sur ce paquet se trouvent sur Section 8.34.2, « Contenu de Bash. »

6.5. Coreutils-8.32

Le paquet Coreutils contient des outils pour afficher et configurer les caractéristiques basiques d'un système.

Temps de construction approximatif: 0.5 SBU
Espace disque requis: 170 Mo

6.5.1. Installation de Coreutils

Préparez la compilation de Coreutils :

```
./configure --prefix=/usr                     \
            --host=$LFS_TGT                   \
            --build=$(build-aux/config.guess) \
            --enable-install-program=hostname \
            --enable-no-install-program=kill,uptime
```

Voici la signification des options de configuration

`--enable-install-program=hostname`
Cela active la construction et l'installation du binaire **hostname** – elles sont désactivées par défaut mais le binaire est requis par la suite de tests de Perl.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Déplacez les programmes à leur emplacement final. Bien que ce ne soit pas nécessaire dans cet environnement temporaire, nous devons le faire parce que certains programmes codent la position des exécutables en dur :

```
mv -v $LFS/usr/bin/{cat,chgrp,chmod,chown,cp,date,dd,df,echo} $LFS/bin
mv -v $LFS/usr/bin/{false,ln,ls,mkdir,mknod,mv,pwd,rm}        $LFS/bin
mv -v $LFS/usr/bin/{rmdir,stty,sync,true,uname}               $LFS/bin
mv -v $LFS/usr/bin/{head,nice,sleep,touch}                    $LFS/bin
mv -v $LFS/usr/bin/chroot                                     $LFS/usr/sbin
mkdir -pv $LFS/usr/share/man/man8
mv -v $LFS/usr/share/man/man1/chroot.1                        $LFS/usr/share/man
sed -i 's/"1"/"8"/'                                           $LFS/usr/share/man
```

Les détails de ce paquet se trouvent sur Section 8.52.2, « Contenu de Coreutils. »

6.6. Diffutils-3.7

Le paquet Diffutils contient les programmes montrant les différences entre fichiers ou répertoires.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 26 Mo

6.6.1. Installation de Diffutils

Préparez la compilation de Diffutils :

```
./configure --prefix=/usr --host=$LFS_TGT
```

Compilez le paquet :

```
make
```

Installez ce paquet :

```
make DESTDIR=$LFS install
```

Les détails de ce paquet se trouvent sur Section 8.54.2, « Contenu de Diffutils. »

6.7. File-5.39

Le paquet File contient un outil pour déterminer le type d'un fichier ou des fichiers donnés.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 31 Mo

6.7.1. Installation de File

La commande **file** sur l'hôte de construction doit être à la même version que celle que nous construisons pour créer le fichier de signature. Lancez les commandes suivantes pour la construire :

```
mkdir build
pushd build
  ../configure --disable-bzlib      \
               --disable-libseccomp \
               --disable-xzlib      \
               --disable-zlib
  make
popd
```

Voici la signification des nouvelles options de configure :

`--disable-*`
Le script de configuration essaye d'utiliser certains paquets de la distribution hôte si les fichiers de bibliothèques correspondantes existent. Cela peut causer un échec à la construction si un fichier de bibliothèque existe, mais pas les fichiers d'en-têtes correspondants. Ces options évitent d'utiliser ces fonctionnalités inutiles de l'hôte.

Préparez la compilation de File :

```
./configure --prefix=/usr --host=$LFS_TGT --build=$(./config.guess)
```

Compilez le paquet :

```
make FILE_COMPILE=$(pwd)/build/src/file
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Les détails de ce paquet se trouvent sur Section 8.10.2, « Contenu de File. »

6.8. Findutils-4.8.0

Le paquet Findutils contient des programmes de recherche de fichiers. Ces programmes sont fournis pour rechercher récursivement dans une hiérarchie de répertoires et pour créer, maintenir et chercher dans une base de données (souvent plus rapide que la recherche récursive mais moins fiable si la base de données n'a pas été mise à jour récemment).

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 42 Mo

6.8.1. Installation de Findutils

Préparez la compilation de Findutils :

```
./configure --prefix=/usr   \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Déplacez l'exécutable à son emplacement final :

```
mv -v $LFS/usr/bin/find $LFS/bin
sed -i 's|find:=${BINDIR}|find:=/bin|' $LFS/usr/bin/updatedb
```

Les détails de ce paquet se trouvent sur Section 8.56.2, « Contenu de Findutils. »

6.9. Gawk-5.1.0

Le paquet Gawk contient des programmes de manipulation de fichiers texte.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 46 Mo

6.9.1. Installation de Gawk

Tout d'abord, assurez-vous que certains fichiers inutiles ne sont pas installés :

```
sed -i 's/extras//' Makefile.in
```

Préparez la compilation de Gawk :

```
./configure --prefix=/usr   \
            --host=$LFS_TGT \
            --build=$(./config.guess)
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Les détails de ce paquet se trouvent sur Section 8.55.2, « Contenu de Gawk. »

6.10. Grep-3.6

Le paquet Grep contient des programmes de recherche dans le contenu de fichiers.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 26 Mo

6.10.1. Installation de Grep

Préparez la compilation de Grep :

```
./configure --prefix=/usr   \
            --host=$LFS_TGT \
            --bindir=/bin
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Les détails de ce paquet se trouvent sur Section 8.33.2, « Contenu de Grep. »

6.11. Gzip-1.10

Le paquet Gzip contient des programmes de compression et décompression de fichiers.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 10 Mo

6.11.1. Installation de Gzip

Préparez la compilation de Gzip :

```
./configure --prefix=/usr --host=$LFS_TGT
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Déplacez l'exécutable à son emplacement final :

```
mv -v $LFS/usr/bin/gzip $LFS/bin
```

Les détails de ce paquet se trouvent sur Section 8.60.2, « Contenu de Gzip. »

6.12. Make-4.3

Le paquet Make contient un programme pour contrôler la génération d'exécutables et d'autres fichiers non-sources d'un paquet à partir des fichiers sources.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 16 Mo

6.12.1. Installation de Make

Préparez la compilation de Make :

```
./configure --prefix=/usr   \
            --without-guile \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Voici la signification des nouvelles options de configure :

`--without-guile`
Bien que nous compilions de manière croisée, configure essaye d'utiliser le guile de l'hôte s'il le trouve. Cela fait échouer la compilation, donc ce paramètre évite de l'utiliser.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Les détails de ce paquet se trouvent sur Section 8.64.2, « Contenu de Make. »

6.13. Patch-2.7.6

Le paquet Patch contient un programme permettant de modifier et de créer des fichiers en appliquant un fichier correctif (appelé habituellement « patch ») créé généralement par le programme **diff**.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 13 Mo

6.13.1. Installation de Patch

Préparez la compilation de Patch :

```
./configure --prefix=/usr   \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Les détails de ce paquet se trouvent sur Section 8.65.2, « Contenu de Patch. »

6.14. Sed-4.8

Le paquet Sed contient un éditeur de flux.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 21 Mo

6.14.1. Installation de Sed

Préparez la compilation de Sed :

```
./configure --prefix=/usr   \
            --host=$LFS_TGT \
            --bindir=/bin
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Les détails de ce paquet se trouvent sur Section 8.29.2, « Contenu de Sed. »

6.15. Tar-1.34

Le paquet Tar fournit la possibilité de créer des archives tar et effectuer diverses manipulations d'archives. Tar peut être utilisé sur des archives précédemment créées pour extraire des fichiers, ajouter des fichiers supplémentaires, mettre à jour ou lister les fichiers qui étaient déjà stockés.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 40 Mo

6.15.1. Installation de Tar

Préparez la compilation de Tar :

```
./configure --prefix=/usr                      \
            --host=$LFS_TGT                    \
            --build=$(build-aux/config.guess) \
            --bindir=/bin
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Les détails de ce paquet se trouvent sur Section 8.67.2, « Contenu de Tar. »

6.16. Xz-5.2.5

Le paquet Xz contient des programmes de compression et de décompression de fichiers. Il offre les possibilités des formats lzma et des formats de compression récents. La compression de fichiers textes avec **xz** donne un meilleur pourcentage de compression qu'avec les commandes **gzip** ou **bzip2** traditionnelles.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 16 Mo

6.16.1. Installation de Xz

Préparez la compilation de Xz :

```
./configure --prefix=/usr                       \
            --host=$LFS_TGT                     \
            --build=$(build-aux/config.guess) \
            --disable-static                    \
            --docdir=/usr/share/doc/xz-5.2.5
```

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Assurez-vous que tous les fichiers essentiels sont dans le bon répertoire :

```
mv -v $LFS/usr/bin/{lzma,unlzma,lzcat,xz,unxz,xzcat}  $LFS/bin
mv -v $LFS/usr/lib/liblzma.so.*                       $LFS/lib
ln -svf ../../lib/$(readlink $LFS/usr/lib/liblzma.so) $LFS/usr/lib/liblzma.so
```

Les détails de ce paquet se trouvent sur Section 8.8.2, « Contenu de Xz. »

6.17. Binutils-2.36.1 — Passe 2

Le paquet Binutils contient un éditeur de liens, un assembleur et d'autres outils pour gérer des fichiers objets.

Temps de construction approximatif: 1.3 SBU
Espace disque requis: 517 Mo

6.17.1. Installation de Binutils

Créez à nouveau un répertoire de construction séparé :

```
mkdir -v build
cd       build
```

Préparez Binutils pour la compilaton :

```
../configure                   \
    --prefix=/usr              \
    --build=$(../config.guess) \
    --host=$LFS_TGT            \
    --disable-nls              \
    --enable-shared            \
    --disable-werror           \
    --enable-64-bit-bfd
```

Voici la signification des nouvelles options de configure :

`--enable-shared`
Construit `libbfd` en tant que bibliothèque partagée.

`--enable-64-bit-bfd`
Active la prise en charge du 64 bits (sur les hôtes avec des tailles de mots plus petites). Cela n'est peut-être pas nécessaire sur les systèmes 64 bits, mais ça ne fait pas de mal.

Compilez le paquet :

```
make
```

Installez le paquet, et contournez un problème qui fait que `libctf.so` se lie avec la zlib de la distribution hôte :

```
make DESTDIR=$LFS install
install -vm755 libctf/.libs/libctf.so.0.0.0 $LFS/usr/lib
```

Les détails sur ce paquet se trouvent sur Section 8.18.2, « Contenu de Binutils. »

6.18. GCC-10.2.0 — Passe 2

Le paquet GCC contient la collection de compilateurs GNU, qui inclut les compilateurs C et C++.

Temps de construction approximatif: 11 SBU

Espace disque requis: 3.7 Go

6.18.1. Installation de GCC

Comme pour la première construction de GCC, les paquets GMP, MPFR et MPC sont requis. Déballez les archives et déplacez-les dans les répertoires avec le nom requis :

```
tar -xf ../mpfr-4.1.0.tar.xz
mv -v mpfr-4.1.0 mpfr
tar -xf ../gmp-6.2.1.tar.xz
mv -v gmp-6.2.1 gmp
tar -xf ../mpc-1.2.1.tar.gz
mv -v mpc-1.2.1 mpc
```

Si vous construisez sur x86_64, changez le nom de répertoire par défaut pour les bibliothèques 64 bits en « lib » :

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' -i.orig gcc/config/i386/t-linux64
  ;;
esac
```

Créez à nouveau un répertoire de construction séparé :

```
mkdir -v build
cd       build
```

Créez un lien symbolique qui permet à libgcc de se construire avec la prise en charge des thread posix :

```
mkdir -pv $LFS_TGT/libgcc
ln -s ../../../libgcc/gthr-posix.h $LFS_TGT/libgcc/gthr-default.h
```

Avant de démarrer la construction de GCC, rappelez-vous d'effacer les variables d'environnement qui modifient les drapeaux d'optimisation par défaut.

Maintenant préparez GCC à la compilation :

```
../configure                                       \
    --build=$(../config.guess)                     \
    --host=$LFS_TGT                                \
    --prefix=/usr                                  \
    CC_FOR_TARGET=$LFS_TGT-gcc                     \
    --with-build-sysroot=$LFS                      \
    --enable-initfini-array                        \
    --disable-nls                                  \
    --disable-multilib                             \
    --disable-decimal-float                        \
    --disable-libatomic                            \
    --disable-libgomp                              \
    --disable-libquadmath                          \
    --disable-libssp                               \
    --disable-libvtv                               \
    --disable-libstdcxx                            \
    --enable-languages=c,c++
```

Voici la signification des nouvelles options de configure :

`-with-build-sysroot=$LFS`
: Normalement, utiliser *`--host`* s'assure que le compilateur croisé est utilisé pour construire GCC, et que le compilateur sait qu'il doit chercher les en-têtes et les bibliothèques dans `$LFS`. Mais le système de construction de GCC utilise d'autres outils qui ne connaissent pas cet emplacement. Ce paramètre est requis pour qu'ils trouvent les fichiers requis dans `$LFS` et non sur l'hôte.

`--enable-initfini-array`
: Cette option est automatiquement activée lors de la construction d'un compilateur natif avec un compilateur natif sur x86. Mais ici, nous construisons avec un compilateur croisé, donc nous devons explicitement utiliser cette option.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make DESTDIR=$LFS install
```

Comme touche finale, créez un lien symbolique utilitaire. De nombreux programmes et scripts lancent **cc** au lieu de **gcc**, pour que les programmes restent génériques et donc utilisables sur n'importe quel type de système UNIX où le compilateur C de GNU n'est pas toujours installé. Lancer **cc** laisse l'administrateur système libre de décider quel compilateur C installer :

```
ln -sv gcc $LFS/usr/bin/cc
```

Les détails de ce paquet se trouvent sur Section 8.26.2, « Contenu de GCC. »

Chapitre 7. Entrée dans le chroot et construction des outils temporaires supplémentaires

7.1. Introduction

Ce chapitre montre comme construire les dernières parties du système temporaire : tout d'abord, les outils requis par la machinerie de construction de plusieurs paquets, puis trois paquets requis pour lancer les tests. Maintenant que toutes les dépendances circulaires ont été résolues, nous pouvons utiliser un environnement « chroot », complètement isolé du système d'exploitation hôte utilisé pour la construction, à l'exception du noyau.

Pour faire fonctionner correctement l'environnement isolé, il faut établir quelques canaux de communication avec le noyau. Cela se fait à travers les *systèmes de fichiers noyau virtuel*, qui doit être monté en entrant dans l'environnement chroot. Vous devriez vérifier qu'ils sont montés en lançant **findmnt**.

Jusqu'au chapitre Section 7.4, « Entrer dans l'environnement chroot », les commandes doivent être lancées en tant que `root`, avec la variable d'environnement `LFS` correctement initialisée. Après l'entrée dans le chroot, toutes les commandes sont lancées en root, heureusement sans accès à l'OS de l'ordinateur sur lequel vous construisez LFS. Soyez prudent cependant, car il est facile de détruire le système LFS complet avec des commandes mal formées.

7.2. Changer de propriétaire

Note

Les commandes dans la suite de ce livre doivent être exécutées alors que vous êtes connecté en tant que `root` et pas en tant qu'utilisateur `lfs`. Contrôlez à nouveau que `$LFS` est paramétré dans l'environnement de `root`.

Pour l'instant, la hiérarchie complète des répertoires de `$LFS` appartient à l'utilisateur `lfs`, un utilisateur qui n'existe que sur le système hôte. Si les répertoires et fichiers dans `$LFS` restent ainsi, ils appartiendront à un ID utilisateur sans compte correspondant. C'est dangereux car un compte utilisateur créé plus tard pourrait se voir attribuer ce même ID utilisateur et être propriétaire du répertoire `$LFS`, exposant ainsi ces fichiers à des manipulations suspectes.

Pour éviter ce problème, changez le propriétaire du répertoire `$LFS` en le rendant à l'utilisateur `root` en exécutant la commande suivante :

```
chown -R root:root $LFS/{usr,lib,var,etc,bin,sbin,tools}
case $(uname -m) in
  x86_64) chown -R root:root $LFS/lib64 ;;
esac
```

7.3. Préparer les systèmes de fichiers virtuels du noyau

Différents systèmes de fichiers exportés par le noyau sont utilisés pour communiquer avec le noyau lui-même. Ces systèmes de fichiers sont virtuels du fait qu'aucun espace disque n'est utilisé pour eux. Le contenu de ces systèmes de fichiers réside en mémoire.

Commencez en créant les répertoires dans lesquels les systèmes de fichiers seront montés :

```
mkdir -pv $LFS/{dev,proc,sys,run}
```

7.3.1. Création des nœuds initiaux vers les périphériques

Quand le noyau démarre le système, il a besoin de la présence de quelques fichiers de périphériques, en particulier les périphériques `console` et `null`. Les nœuds de périphérique doivent être créés sur le disque dur afin d'être disponible avant que le noyau ne remplisse `/dev`, et aussi quand Linux est démarré avec *`init=/bin/bash`*. Créez les périphériques en exécutant les commandes suivantes :

```
mknod -m 600 $LFS/dev/console c 5 1
mknod -m 666 $LFS/dev/null c 1 3
```

7.3.2. Monter et peupler /dev

La méthode recommandée pour peupler le répertoire `/dev` de périphériques est de monter un système de fichiers virtuel (comme `tmpfs`) sur le répertoire `/dev`, et d'autoriser la création dynamique des périphériques sur le système de fichiers virtuel une fois qu'ils sont détectés ou que quelque chose tente d'y accéder. La création de périphériques est généralement faite par Udev lors du démarrage. Comme ce nouveau système ne contient pas encore Udev et n'a pas encore été démarré, il est nécessaire de monter et de peupler `/dev` manuellement. Cela se fait en montant en double le répertoire `/dev` du système hôte. Le montage en double est un type spécial de montage qui vous permet de créer le miroir d'un répertoire ou d'un point de montage à un autre endroit. Utilisez la commande suivante pour réaliser cela :

```
mount -v --bind /dev $LFS/dev
```

7.3.3. Monter les systèmes de fichiers virtuels du noyau

Maintenant montez les systèmes de fichiers virtuels du noyau qui en résultent :

```
mount -v --bind /dev/pts $LFS/dev/pts
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
mount -vt tmpfs tmpfs $LFS/run
```

Dans certains systèmes hôtes, `/dev/shm` est un lien symbolique vers `/run/shm`. Le tmpfs /run a été monté tout à l'heure, donc vous ne devez créer un répertoire que dans ce cas précis.

```
if [ -h $LFS/dev/shm ]; then
  mkdir -pv $LFS/$(readlink $LFS/dev/shm)
fi
```

7.4. Entrer dans l'environnement chroot

Maintenant que tous les paquets requis pour construire le reste des outils nécessaires sont sur le système, il est temps d'entrer dans l'environnement chroot pour finir d'installer les outils temporaires restant. Nous utiliserons aussi cet environnement pour installer le système final. En tant que `root`, lancez la commande suivante pour entrer dans ce petit monde peuplé seulement, pour le moment, des outils temporaires :

```
chroot "$LFS" /usr/bin/env -i   \
    HOME=/root                  \
    TERM="$TERM"                \
    PS1='(lfs chroot) \u:\w\$ ' \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin \
    /bin/bash --login +h
```

L'option *-i* donnée à la commande **env** effacera toutes les variables de l'environnement chroot. Après cela, seules les variables HOME, TERM, PS1 et PATH sont de nouveau initialisées. La construction *TERM=$TERM* initialisera la variable TERM à l'intérieur du chroot avec la même valeur qu'à l'extérieur ; cette variable est nécessaire pour que des programmes comme **vim** et **less** fonctionnent correctement. Si vous avez besoin de la présence d'autres variables, telles que CFLAGS or CXXFLAGS, c'est le bon moment pour les initialiser de nouveau.

À partir de maintenant, il n'est plus nécessaire d'utiliser la variable LFS parce que tout le travail sera restreint au système de fichiers LFS, car on a dit au shell Bash que $LFS est maintenant le répertoire racine (/).

Remarquez que /tools/bin n'est pas dans le PATH. Ceci signifie que la chaîne d'outils croisée ne sera plus utilisé dans l'environnement chroot. Ceci survient quand le shell ne se « rappelle » plus des emplacements des binaires exécutés— Pour cette raison, le hachage est désactivé en passant l'option *+h* à **bash**.

Remarquez que l'invite **bash** dira `I have no name!`. Ceci est normal car le fichier /etc/passwd n'a pas encore été créé.

Note

Il est important que toutes les commandes pour le reste de ce chapitre et les chapitres suivants soient lancées à l'intérieur de l'environnement chroot. Si vous devez quitter cet environnement pour une quelconque raison (un redémarrage par exemple), assurez-vous que les systèmes de fichiers virtuels sont montés comme expliqué dans Section 7.3.2, « Monter et peupler /dev » et Section 7.3.3, « Monter les systèmes de fichiers virtuels du noyau » et entrez de nouveau dans chroot avant de continuer les installations.

7.5. Créer les répertoires

Il est temps de créer la structure complète du système de fichiers LFS.

Créez quelques répertoires de plus haut niveau qui ne font pas partie de l'ensemble limité requis dans les chapitres précédents en lançant la commande suivante :

Note

Certains des répertoires suivants ont déjà été créés plus tôt avec des instructions explicites ou lors de l'installation de certains paquets. Ils sont répétés ici pour être complet.

```
mkdir -pv /{boot,home,mnt,opt,srv}
```

Créez l'ensemble de sous-répertoires requis sous la racine en lançant les commandes suivantes :

```
mkdir -pv /etc/{opt,sysconfig}
mkdir -pv /lib/firmware
mkdir -pv /media/{floppy,cdrom}
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{color,dict,doc,info,locale,man}
mkdir -pv /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
mkdir -pv /var/{cache,local,log,mail,opt,spool}
mkdir -pv /var/lib/{color,misc,locate}

ln -sfv /run /var/run
ln -sfv /run/lock /var/lock

install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
```

Par défaut, les répertoires sont créés avec les droits 755, ce qui n'est pas souhaitable pour tous les répertoires. Dans la commande ci-dessus, deux modifications seront effectuées—une pour le répertoire principal de `root`, et une autre pour les répertoires des fichiers temporaires.

Le premier changement de droit nous assure que n'importe qui ne pourra pas entrer dans le répertoire `/root`—de façon identique à ce que ferait un utilisateur pour son répertoire principal. Le deuxième changement assure que tout utilisateur peut écrire dans les répertoires `/tmp` et `/var/tmp`, mais ne peut pas supprimer les fichiers des autres utilisateurs. Cette dernière interdiction est due au « sticky bit », le bit (1) le plus haut dans le masque 1777.

7.5.1. Remarques à propos de la conformité FHS

L'arborescence des répertoires est basée sur le standard FHS *Filesystem Hierarchy Standard*, (disponible sur *https://refspecs.linuxfoundation.org/fhs.shtml*). Le FHS stipule aussi l'existence de quelques répertoires comme `/usr/local/games` et `/usr/share/games`. Nous créons seulement les répertoires nécessaires. Néanmoins, n'hésitez pas à créer ces répertoires.

7.6. Créer les fichiers et les liens symboliques essentiels

Historiquement, Linux gère la liste des systèmes de fichiers montés dans le fichier `/etc/mtab`. Les noyaux modernes gèrent cette liste en interne via le système de fichiers `/proc`. Pour contenter les outils qui s'attendent à la présence de `/etc/mtab`, créez le lien symbolique suivant :

```
ln -sv /proc/self/mounts /etc/mtab
```

Créez un fichier `/etc/hosts` de base qui sera utilisé dans certaines suites de tests, et par l'un des fichiers de configuration de Perl :

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

Afin que l'utilisateur `root` puisse s'identifier et que le nom « root » soit reconnu, il doit y avoir des entrées cohérentes dans les fichiers `/etc/passwd` et `/etc/group`.

Créez le fichier `/etc/passwd` en lançant la commande suivante :

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/bin/false
daemon:x:6:6:Daemon User:/dev/null:/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/run/dbus:/bin/false
systemd-bus-proxy:x:72:72:systemd Bus Proxy:/:/bin/false
systemd-journal-gateway:x:73:73:systemd Journal Gateway:/:/bin/false
systemd-journal-remote:x:74:74:systemd Journal Remote:/:/bin/false
systemd-journal-upload:x:75:75:systemd Journal Upload:/:/bin/false
systemd-network:x:76:76:systemd Network Management:/:/bin/false
systemd-resolve:x:77:77:systemd Resolver:/:/bin/false
systemd-timesync:x:78:78:systemd Time Synchronization:/:/bin/false
systemd-coredump:x:79:79:systemd Core Dumper:/:/bin/false
uuidd:x:80:80:UUID Generation Daemon User:/dev/null:/bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
EOF
```

Le mot de passe réel pour `root` sera paramétré plus tard.

Créez le fichier `/etc/group` en exécutant la commande suivante :

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:daemon
sys:x:2:
kmem:x:3:
tape:x:4:
tty:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
systemd-journal:x:23:
input:x:24:
mail:x:34:
kvm:x:61:
systemd-bus-proxy:x:72:
systemd-journal-gateway:x:73:
systemd-journal-remote:x:74:
systemd-journal-upload:x:75:
systemd-network:x:76:
systemd-resolve:x:77:
systemd-timesync:x:78:
systemd-coredump:x:79:
uuidd:x:80:
wheel:x:97:
nogroup:x:99:
users:x:999:
EOF
```

Les groupes créés ne font partie d'aucun standard—ce sont des groupes décidés en partie en fonction des besoins de la configuration de Udev dans le chapitre 9, et en partie par la coutume utilisée par un certain nombre de distributions Linux existantes. En outre, certaines suites de tests s'appuient sur des groupes et des utilisateurs spécifiques. La base Linux standard (Linux Standard Base ou LSB, disponible sur *http://refspecs.linuxfoundation.org/lsb.shtml*) recommande seulement cela, ainsi que la présence d'un groupe `root` (GID 0) et d'un groupe `bin` (GID 1). Tous les autres noms de groupe et GID peuvent être librement choisis par l'administrateur du système puisque les programmes bien écrits ne dépendent pas des numéros GID, mais utilisent plutôt le nom du groupe.

Certains tests dans Chapitre 8 ont besoin d'un utilisateur non privilégié. Nous ajoutons cet utilisateur ici et supprimons ce compte à la fin de ce chapitre.

```
echo "tester:x:$(ls -n $(tty) | cut -d" " -f3):101::/home/tester:/bin/bash" >> /
echo "tester:x:101:" >> /etc/group
install -o tester -d /home/tester
```

Pour supprimer l'invite « I have no name! », démarrez un nouveau shell. Comme nous avons créé les fichiers `/etc/passwd` et `/etc/group`, la résolution du nom d'utilisateur et de groupe fonctionnera à présent :

```
exec /bin/bash --login +h
```

Remarquez l'utilisation du paramètre *+h*. Il dit à **bash** de ne pas utiliser son hachage de chemin interne. Sans ce paramètre, **bash** se rappellerait des chemins vers les binaires qu'il a exécutés. Pour s'assurer que les binaires nouvellement compilés seront utilisés dès qu'ils seront installés, le paramètre *+h* sera utilisée durant tout le chapitre.

Les programmes **login**, **agetty**, et **init** (et d'autres) utilisent un nombre de journaux applicatifs pour enregistrer des informations comme qui s'est connecté sur le système et quand. Mais ces programmes n'écriront pas vers ces journaux s'ils n'existent pas. Initialisez les journaux et donnez-leur les bons droits :

```
touch /var/log/{btmp,lastlog,faillog,wtmp}
chgrp -v utmp /var/log/lastlog
chmod -v 664  /var/log/lastlog
chmod -v 600  /var/log/btmp
```

Le fichier `/var/log/wtmp` enregistre toutes les connexions et les déconnexions. Le fichier `/var/log/lastlog` enregistre le moment de dernière connexion de chaque utilisateur. Le fichier `/var/log/faillog` enregistre les tentatives de connexion échouées. Le fichier `/var/log/btmp` enregistre les mauvaises tentatives de connexion.

Note

Le fichier `/run/utmp` enregistre les utilisateurs qui sont actuellement connectés. Ce fichier est créé de manière dynamique dans les scripts de démarrage.

7.7. Libstdc++ de GCC-10.2.0, Passe 2

Lors de la construction de gcc-pass2 nous avons repoussé l'installation de la bibliothèque standard C++ parce qu'aucun compilateur convenable n'était disponible pour la construire. Nous ne pouvons pas utiliser le compilateur construit dans cette section parce que c'est un compilateur natif et qu'il ne doit pas être utilisé en dehors du chroot ou il risquerait de polluer les bibliothèques avec des composants de l'hôte.

Temps de construction approximatif: 0.8 SBU
Espace disque requis: 1.1 Go

7.7.1. Installation de Target Libstdc++

Note

Libstdc++ fait partie des sources de GCC. Vous devez d'abord déballer l'archive de GCC et vous déplacer dans le répertoire `gcc-10.2.0`.

Créez un lien qui existe lors de la construction de libstdc++ dans l'arborescence des sources de gcc :

```
ln -s gthr-posix.h libgcc/gthr-default.h
```

Créez un répertoire de construction séparé pour libstdc++ et entrez dedans :

```
mkdir -v build
cd       build
```

Préparez la compilation de libstdc++ :

```
../libstdc++-v3/configure           \
    CXXFLAGS="-g -O2 -D_GNU_SOURCE"  \
    --prefix=/usr                    \
    --disable-multilib               \
    --disable-nls                    \
    --host=$(uname -m)-lfs-linux-gnu \
    --disable-libstdcxx-pch
```

Voici la signification des options de configure :

`CXXFLAGS="-g -O2 -D_GNU_SOURCE"`
Ces drapeaux sont passés par le Makefile de plus haut niveau lors de la construction complète de GCC.

`--host=$(uname -m)-lfs-linux-gnu`
Nous devons mimer ce qui arriverait si ce paquet était construit avec la construction complète du compilateur. Ce drapeau serait passé au configure par la machinerie de construction de GCC.

`--disable-libstdcxx-pch`
Ce paramètre évite l'installation des fichiers d'en-tête pré-compilés, qui ne sont pas requis pour le moment.

Compilez libstdc++ en lançant :

```
make
```

Installez la bibliothèque :

```
make install
```

Les détails de ce paquet se trouvent dans Section 8.26.2, « Contenu de GCC. »

7.8. Gettext-0.21

Le paquet Gettext contient des outils pour l'internationalisation et la localisation. Ceci permet aux programmes d'être compilés avec la prise en charge des langues natives (*Native Language Support* ou NLS), pour afficher des messages dans la langue native de l'utilisateur.

Temps de construction approximatif: 1.8 SBU
Espace disque requis: 310 Mo

7.8.1. Installation de Gettext

Pour notre ensemble temporaire d'outils, nous avons besoin uniquement d'installer trois programmes de Gettext.

Préparez la compilation de Gettext :

```
./configure --disable-shared
```

Voici la signification des options de configure :

`--disable-shared`
Nous n'avons pas besoin d'installer les bibliothèques partagées de Gettext pour l'instant, donc il n'y a pas besoin de les construire.

Compilez le paquet :

```
make
```

Installez les programmes **msgfmt**, **msgmerge** et **xgettext** :

```
cp -v gettext-tools/src/{msgfmt,msgmerge,xgettext} /usr/bin
```

Les détails de ce paquet se trouvent sur Section 8.31.2, « Contenu de Gettext. »

7.9. Bison-3.7.5

Le paquet Bison contient un générateur d'analyseurs.

Temps de construction approximatif: 0.3 SBU
Espace disque requis: 52 Mo

7.9.1. Installation de Bison

Préparez la compilation de Bison :

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/bison-3.7.5
```

Voici la signification des nouvelles options de configure :

`--docdir=/usr/share/doc/bison-3.7.5`
Cela dit au système de construction d'installer la documentation de bison dans un répertoire versionné.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails de ce paquet se trouvent dans Section 8.32.2, « Contenu de Bison. »

7.10. Perl-5.32.1

Le paquet Perl contient le langage pratique d'extraction et de rapport (*Practical Extraction and Report Language*).

Temps de construction approximatif: 1.7 SBU
Espace disque requis: 268 Mo

7.10.1. Installation de Perl

Préparez la compilation de Perl :

```
sh Configure -des                                         \
             -Dprefix=/usr                                \
             -Dvendorprefix=/usr                          \
             -Dprivlib=/usr/lib/perl5/5.32/core_perl      \
             -Darchlib=/usr/lib/perl5/5.32/core_perl      \
             -Dsitelib=/usr/lib/perl5/5.32/site_perl      \
             -Dsitearch=/usr/lib/perl5/5.32/site_perl     \
             -Dvendorlib=/usr/lib/perl5/5.32/vendor_perl \
             -Dvendorarch=/usr/lib/perl5/5.32/vendor_perl
```

Voici la signification des nouvelles options de configure :

`-des`
C'est la combinaison de trois options : -d utilise les valeurs par défaut pour tous les éléments ; -e s'assure que toutes les tâches sont effectuées ; -s rend silencieuses les sorties non importantes.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails de ce paquet se trouvent dans Section 8.40.2, « Contenu de Perl. »

7.11. Python-3.9.2

Le paquet Python 3 contient l'environnement de développement Python. Il est utile pour programmer en orienté-objet, écrire des scripts, prototyper de plus grands programmes ou pour développer des applications complètes.

Temps de construction approximatif: 0.9 SBU
Espace disque requis: 374 Mo

7.11.1. Installation de Python

Note

Il y a deux fichiers de paquet dont le nom commence par « python ». Celui à extraire est `Python-3.9.2.tar.xz` (attention à la majuscule sur la première lettre).

Préparez Python pour la compilation :

```
./configure --prefix=/usr   \
            --enable-shared \
            --without-ensurepip
```

Voici la signification de l'option de configure :

`--enable-shared`
Ce paramètre évite l'installation des bibliothèques statiques.

`--without-ensurepip`
Ce paramètre désactive l'installateur de paquets Python, qui n'est pas requise pour le moment.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails de ce paquet se trouvent dans Section 8.49.2, « Contenu de Python 3. »

7.12. Texinfo-6.7

Le paquet Texinfo contient des programmes de lecture, écriture et conversion des pages Info.

Temps de construction approximatif: 0.3 SBU
Espace disque requis: 105 Mo

7.12.1. Installation de Texinfo

Préparez la compilation de Texinfo :

```
./configure --prefix=/usr
```

Note

Pendant la procédure de configuration, un test indique une erreur pour TestXS_la-TestXS.lo. Cela n'a pas de sens pour LFS et vous pouvez l'ignorer.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails de ce paquet se trouvent sur Section 8.68.2, « Contenu de Texinfo. »

7.13. Util-linux-2.36.2

le paquet Util-linux contient divers programmes utilitaires.

Temps de construction approximatif: 0.7 SBU
Espace disque requis: 134 Mo

7.13.1. Installation de Util-linux

Les FHS recommandent d'utiliser le répertoire `/var/lib/hwclock` au lieu du répertoire habituel `/etc` comme emplacement du fichier `adjtime`. Créez ce répertoire avec :

```
mkdir -pv /var/lib/hwclock
```

Préparez la compilation de Util-linux :

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime    \
            --docdir=/usr/share/doc/util-linux-2.36.2 \
            --disable-chfn-chsh  \
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
            --disable-runuser    \
            --disable-pylibmount \
            --disable-static     \
            --without-python     \
            runstatedir=/run
```

Voici la signification des options de configure :

`ADJTIME_PATH=/var/lib/hwclock/adjtime`
Cela met en place l'emplacement du fichier enregistrant les informations sur l'horloge matérielle en accord avec la FHS. Cela n'est pas strictement requis pour cet outil temporaire, mais cela évite de créer le fichier ailleurs, et qu'il doive être remplacé ou supprimé en construisant le paquet util-linux final.

`--disable-*`
Ces paramètres évitent des avertissements à propos de la construction des composants qui requièrent des paquets qui ne sont pas dans LFS ou pas encore installés.

`--without-python`
Ce paramètre désactive l'utilisation de Python. Cela évite de construire des liaisons inutiles.

`runstatedir=/run`
Ce paramètre indique l'emplacement du socket utilisé par **uuidd** et `libuuid`.

Compilez le paquet :

```
make
```

Installez le paquet :

```
make install
```

Les détails de ce paquet se trouvent sur Section 8.73.2, « Contenu d'Util-linux. »

7.14. Nettoyage et sauvegarde du système temporaire

Les fichiers .la de libtool ne sont utiles que pour lier des bibliothèques statiques. Ils ne sont pas requis, et potentiellement dangereux, quand on utilise des bibliothèques dynamiques, surtout pour les systèmes de construction qui n'utilisent pas les autotools. Toujours dans le chroot, supprimez ces fichiers maintenant :

```
find /usr/{lib,libexec} -name \*.la -delete
```

Supprimez la documentation des outils temporaires, pour éviter qu'elle ne se retrouve sur le système final, et sauvez environ 35 :Mo :

```
rm -rf /usr/share/{info,man,doc}/*
```

Note

Toutes les étapes restantes dans cette section sont optionnelles. Cependant, dès que vous commencez à installer des paquets dans Chapitre 8, les outils temporaires seront écrasés. Donc il peut être une bonne idée d'effectuer une sauvegarde des outils temporaires comme décrit plus bas. Les autres étapes ne sont requise que si vous avez vraiment très peu d'espace disque.

Les étapes suivantes sont effectuées depuis l'extérieur de l'environnement chroot. Cela signifie que vous devez quitter l'environnement chroot avant de continuer. La raison pour cela est que :

- cela s'assure que les objets ne sont pas utilisés pendant leur manipulation,
- cela permet d'accéder aux emplacements du système de fichiersr en dehors de l'environnement chroot pour stocker/lire l'archive de sauvegarde qui ne doit pas être placée dans la hiérarchie `$LFS` pour des raisons de sécurité.

Quittez l'environnement chroot et démontez les systèmes de fichiers virtuels du noyau :

Note

Toutes les instructions suivantes sont effectuées en `root`. Faites particulièrement attention aux commandes que vous utilisez car une erreur peut ici modifier votre système hôte. Soyez conscient que la variable d'environnement `LFS` est initialisée par défaut pour l'utilisateur `lfs`, mais ce n'est *pas* le cas pour `root`. Lorsque les commandes doivent être exécutées par `root`, assurez-vous que vous avez correctement initialisé `LFS`. On en a parlé dans Section 2.6, « Définir la variable $LFS ».

```
exit
umount $LFS/dev{/pts,}
umount $LFS/{sys,proc,run}
```

7.14.1. Stripping

Si la partition LFS est plutôt petite, il est bon de savoir que des éléments inutiles peuvent être enlevés. Les exécutables et les bibliothèques construites jusqu'ici contiennent un peu plus de 90 Mo de symbole de débogages inutiles.

Débarrassez les binaires de leurs symboles de débogage :

```
strip --strip-debug $LFS/usr/lib/*
strip --strip-unneeded $LFS/usr/{,s}bin/*
strip --strip-unneeded $LFS/tools/bin/*
```

Ces commandes passeront un certain nombre de fichiers en rapportant qu'elles n'ont pas pu reconnaître leur format de fichier. La plupart sont des scripts au lieu de binaires.

Prenez soin de ne *PAS* utiliser `--strip-unneeded` sur les bibliothèques. Les bibliothèques statiques seraient détruites et les paquets de la chaîne d'outils devraient être de nouveau reconstruits.

Maintenant, vous devriez avoir au moins 5 Go d'espace libre sur la partition chroot qui peut être utilisé pour construire et installer Glibc et GCC dans la prochaine phase. Si vous pouvez construire et installer Glibc, vous pouvez construire et installer le reste aussi. Vous pouvez vérifier l'espace libre avec la commande **df -h $LFS**.

7.14.2. Backup

Maintenant que les outils essentiels ont été créés, il est temps de réfléchir à faire une sauvegarde. Lorsque tous les tests passent correctement dasn les paquets construits précédemment, vos outils temporaires sont dans un bon état et peuvent être sauvegardés pour être utilisés plus tard. Si vous avez une erreur fatale dans le chapitres suivants, cela permet de tout supprimer et de recommencer (en étant plus prudent). Malheureusement, tous les outils temporaires seront supprimés. Pour éviter de passer du temps à refaire quelque chose qui a déjà été construit, préparez une sauvegarde.

Assurez-vous d'avoir au moins 600 Mo d'espace libre (les archives de sources seront incluses dans l'archive de sauvegarde) dans le répertoire personnel de l'utilisateur `root`.

Créez l'archive de sauvegarde en lançant la commande suivante :

```
cd $LFS &&
tar -cJpf $HOME/lfs-temp-tools-10.1-systemd.tar.xz .
```

Remplacez `$HOME` par un répertoire de votre choix si vous ne voulez pas avoir la sauvegarde dans le répertoire personnel de `root`.

7.14.3. Restore

Au cas où vous fassiez une erreur et deviez recommencer de zéro, vous pouvez utiliser cette archive pour restaurer les outils temporaires et gagner du temps lors de la correction. Comme les sources se trouvent dans `$LFS`, elles sont incluses dans l'archive de sauvegarde aussi, donc elles n'ont pas besoin d'être re-téléchargées. Après avoir vérifié que `$LFS` est correctement initialisée, restaurez la sauvegarde en exécutant les commandes suivantes: :

```
cd $LFS &&
rm -rf ./* &&
tar -xpf $HOME/lfs-temp-tools-10.1-systemd.tar.xz
```

De nouveau, vérifiez que l'environnement en correctement paramétré et continuez à construire le reste du système.

Important

Si vous avez quitté l'environnement chroot pour vous débarrasser des symboles de débogage, créer une sauvegarde ou redémarrer une construction depuis une sauvegarde, rappelez-vous de monter les systèmes de fichiers virtuels du noyau comme décrit dans Section 7.3, « Préparer les systèmes de fichiers virtuels du noyau » et de re-rentrer dans l'environnement chroot (voir Section 7.4, « Entrer dans l'environnement chroot ») avant de continuer.

Partie IV. Construction du système LFS

Chapitre 8. Installer les logiciels du système de base

8.1. Introduction

Dans ce chapitre, nous commençons la construction du système LFS pour de bon.

Nous arrivons à la dernière étape de l'installation de ce logiciel. Bien que, dans beaucoup de cas, les instructions d'installation pourraient être plus courtes et plus génériques, nous avons opté pour fournir les instructions complètes pour chaque paquet et minimiser ainsi les possibilités d'erreurs. La clé pour apprendre ce qui fait fonctionner un système Linux est de savoir à quoi sert chaque paquet et pourquoi vous (ou le système) en avez besoin.

Nous ne vous recommandons pas d'utiliser les optimisations. Elles peuvent faire qu'un programme s'exécute un peu plus rapidement, mais elles peuvent aussi causer des problèmes de compilation et des difficultés à l'exécution de ce programme. Si un paquet refuse de compiler lors de l'utilisation d'optimisation, essayez de le compiler sans optimisation pour voir si cela corrige le problème. Même si le paquet compile avec les optimisations, il y a un risque qu'il ait été mal compilé à cause des interactions complexes entre le code et les outils de construction. Remarquez aussi que l'utilisation des options `-march` et `-mtune` peut causer des problèmes avec les paquets de la chaîne d'outils (Binutils, GCC et Glibc). Le petit potentiel de gains obtenu en utilisant les optimisations de compilation est souvent minime comparé aux risques. Les utilisateurs construisant une LFS pour la première fois sont encouragés à construire sans optimisations personnalisées. Le système sera toujours très rapide et restera stable en même temps.

Avant les instructions d'installation, chaque page d'installation fournit des informations sur le paquet, incluant une description concise de ce qu'il contient, approximativement combien de temps prendra la construction et combien d'espace disque est nécessaire pendant le processus de construction. Après les instructions d'installation, il y a une liste de programmes et de bibliothèques (avec quelques brèves descriptions de ceux-ci) que le paquet installe.

Note

Les valeurs SBU et l'espace disque requis incluent les données de suites de tests pour tous les paquets de Chapitre 8 auxquels elles sont applicables. Les valeurs de SBU ont été calculées avec un seul cœur de CPU (-j1) pour toutes les opérations.

8.1.1. À propos des bibliothèques

En général, les éditeurs de LFS déconseillent la construction et l'installation de bibliothèques statiques. L'objectif initial de la plupart des bibliothèques statique a été rendu obsolète dans un système Linux moderne. Par ailleurs la liaison statique de bibliothèques dans un programme peut être nuisible. Si une mise à jour des bibliothèques est nécessaire pour retirer un problème de sécurité, tous les programmes qui utilisent cette bibliothèque vont devoir être liés à nouveau à la nouvelle bibliothèque. Comme l'utilisation de bibliothèques statiques n'est pas toujours évident, on ne connaît même pas forcément les programmes adéquats (et les procédures requises pour faire la liaison).

Dans les procédures de ce chapitre, nous retirons ou désactivons l'installation de la plupart des bibliothèques statiques. Généralement cela ce fait en activant le drapeau `--disable-static` lors de l'exécution de **configure**. Dans d'autres cas, des autres moyens sont nécessaires. Dans de rares cas, surtout pour glibc et gcc, l'utilisation de bibliothèques statiques reste essentielle pour le processus de construction de paquets.

Pour une discussion plus complète à propos des bibliothèques, regardez la discussion *Bibliothèques : statiques ou partagées ?* dans le livre BLFS.

8.2. Gestion de paquets

La gestion de paquets est un ajout souvent demandé au livre LFS. Un gestionnaire de paquets permet de conserver une trace des fichiers installés, simplifiant ainsi leur suppression ou leur mise à jour. Un gestionnaire de paquets gérera tant les fichiers binaires et de bibliothèque que l'installation des fichiers de configuration. Avant tout, NON

—cette section ne parle pas d'un gestionnaire de paquets particulier, elle n'en recommande pas non plus. Elle fait un tour des techniques les plus populaires pour indiquer comment elles fonctionnent. Le parfait gestionnaire de paquets pourrait faire partie de ces techniques ou pourrait être une combinaison d'une ou plusieurs techniques. Cette section mentionne brièvement les problèmes pouvant survenir lors de la mise à jour des paquets.

Parmi les raisons de l'absence d'un gestionnaire de paquets mentionné dans LFS ou BLFS :

- S'occuper de la gestion de paquets est en dehors des buts de ces livres— visant à apprendre comment un système Linux est construit.
- Il existe de nombreuses solutions pour la gestion de paquets, chacune ayant ses forces et ses faiblesses. En inclure une qui satisfait tout le monde est difficile.

Des astuces ont été écrites sur le thème de la gestion de paquets. Visitez le *Projet des astuces* et voyez celui qui satisfait vos besoins.

8.2.1. Problèmes de mise à jour

Un gestionnaire de paquets facilite la mise à jour des nouvelles versions au moment de leur sortie. Généralement, les instructions dans les livres LFS et BLFS peuvent être utilisées pour mettre à jour vers de nouvelles versions. Voici quelques points à connaître pour une mise à jour de paquets, spécifiquement sur un système en cours de fonctionnement.

- Si Glibc doit être mis à jour vers une nouvelle version (par exemple, glibc-2.31 vers glibc-2.32), il est plus sûr de reconstruire LFS. Bien que vous *pourriez* être capable de ne pas reconstruire tous les paquets dans leur ordre de dépendances, nous ne vous le recommandons pas.
- Si un paquet contenant une bibliothèque partagée est mis à jour et si le nom de cette dernière est modifié, alors les paquets liés dynamiquement à la bibliothèque devront être recompilés pour être liés à la nouvelle bibliothèque. Remarquez qu'il n'y a aucune corrélation entre la version du paquet et le nom de la bibliothèque. Par exemple, considérez un paquet foo-1.2.3 qui installe une bibliothèque partagée de nom `libfoo.so.1`. Disons que vous mettez à jour le paquet avec une nouvelle version foo-1.2.4 qui installe une bibliothèque partagée de nom `libfoo.so.2`. Dans ce cas, tous les paquets liés dynamiquement à `libfoo.so.1` doivent être recompilés pour être liés à `libfoo.so.2`. Vous ne devez pas supprimer les anciennes bibliothèques jusqu'à ce que les paquets indépendants soient recompilés.

8.2.2. Techniques de gestion de paquets

Ce qui suit est une liste de techniques habituelles de gestion de paquets. Avant de prendre une décision sur un gestionnaire de paquets, faites une recherche sur les différentes techniques et notamment leurs faiblesses.

8.2.2.1. Tout est dans ma tête !

Oui, c'est une technique de gestion de paquets. Certains n'éprouvent pas le besoin d'un gestionnaire de paquets parce qu'ils connaissent très bien les paquets et connaissent les fichiers installés par chaque paquet. Certains utilisateurs n'en ont pas besoin parce qu'ils planifient la reconstruction entière de LFS lorsqu'un paquet est modifié.

8.2.2.2. Installer dans des répertoires séparés

C'est une gestion des paquets tellement simple qu'elle ne nécessite aucun paquet supplémentaire pour gérer les installations. Chaque paquet est installé dans un répertoire séparé. Par exemple, le paquet foo-1.1 est installé dans `/usr/pkg/foo-1.1` et un lien symbolique est créé de `/usr/pkg/foo` vers `/usr/pkg/foo-1.1`. Lors de l'installation de la nouvelle version foo-1.2, elle est installée dans `/usr/pkg/foo-1.2` et l'ancien lien symbolique est remplacé par un lien symbolique vers la nouvelle version.

Les variables d'environnement telles que `PATH`, `LD_LIBRARY_PATH`, `MANPATH`, `INFOPATH` et `CPPFLAGS` ont besoin d'être étendues pour inclure `/usr/pkg/foo`. Pour plus de quelques paquets, ce schéma devient ingérable.

8.2.2.3. Gestion de paquet par lien symbolique

C'est une variante de la technique précédente. Chaque paquet est installé de façon similaire au schéma précédent. Mais au lieu de réaliser le lien symbolique, chaque fichier dispose d'un lien symbolique vers son équivalent dans la hiérarchie `/usr`. Ceci supprime le besoin d'étendre les variables d'environnement. Bien que les liens symboliques peuvent être créés par l'utilisateur, pour automatiser la création, certains gestionnaires de paquets ont été écrits avec cette approche. Parmi les plus populaires se trouvent Stow, Epkg, Graft et Depot.

L'installation doit être faussée, de façon à ce que chaque paquet pense qu'il est installé dans `/usr` alors qu'en réalité il est installé dans la hiérarchie `/usr/pkg`. Installer de cette manière n'est généralement pas une tâche triviale. Par exemple, considérez que vous installez un paquet libfoo-1.1. Les instructions suivantes pourraient ne pas installer correctement le paquet :

```
./configure --prefix=/usr/pkg/libfoo/1.1
make
make install
```

L'installation fonctionnera, mais les paquets dépendants pourraient ne pas lier libfoo comme vous vous y attendriez. Si vous compilez un paquet qui se lie à `/usr/pkg/libfoo/1.1/lib/libfoo.so.1` au lieu de `/usr/lib/libfoo.so.1` comme vous le prévoyez. La bonne approche est d'utiliser la stratégie `DESTDIR` pour fausser l'installation du paquet. Cette approche fonctionne ainsi :

```
./configure --prefix=/usr
make
make DESTDIR=/usr/pkg/libfoo/1.1 install
```

La plupart des paquets prennent en charge cette approche, mais elle pose problème à certains. Pour les paquets non compatibles, vous pouvez soit les installer manuellement soit trouver plus simple d'installer les paquets problématiques dans `/opt`.

8.2.2.4. Basé sur le temps

Avec cette technique, un fichier est balisé avec l'heure avant l'installation du paquet. Après l'installation, une simple utilisation de la commande **find** avec les options appropriées peut générer une trace de tous les fichiers installés après que le fichier temps a été créé. install-log est un gestionnaire de paquets écrit avec cette approche.

Bien que ce schéma a l'avantage d'être simple, il a deux inconvénients. Si à l'installation, les fichiers sont installés sans balise de temps autre que l'heure actuelle, ces fichiers ne seront pas suivis par le gestionnaire de paquets. De plus, ce schéma peut seulement être utilisé lorsqu'un seul paquet est installé à la fois. Les traces ne sont pas fiables si deux paquets sont installés dans deux consoles différentes.

8.2.2.5. Tracer les scripts d'installation

Avec cette approche, les commandes que les scripts d'installation accomplissent sont enregistrées. Il y a deux techniques que vous pouvez utiliser :

Vous pouvez initialiser la variable d'environnement `LD_PRELOAD` pour qu'elle pointe vers une bibliothèque à précharger avant l'installation. Lors de l'utilisation de cette dernière, cette bibliothèque trace les paquets en cours d'installation en s'attachant eux-mêmes aux différents exécutables comme **cp**, **install**, **mv** et trace les appels système qui modifient le système de fichiers. Pour que cette approche fonctionne, tous les exécutables ont besoin d'être liés dynamiquement sans bit suid ou sgid. Le préchargement de la bibliothèque pourrait causer quelques effets de bord involontaires lors de l'installation ; donc, réalisez quelques tests pour vous assurer que le gestionnaire de paquets ne casse rien et trace bien tous les fichiers appropriés.

La seconde technique est d'utiliser **strace**, qui trace tous les appels du système faits pendant l'exécution des scripts d'installation.

8.2.2.6. Créer des archives de paquets

Dans ce schéma, l'installation d'un paquet est faussée dans un répertoire séparé comme décrit plus haut. Après l'installation, une archive du paquet est créée en utilisant les fichiers installés. L'archive est ensuite utilisée pour installer le paquet soit sur la machine locale soit même sûr d'autres machines.

Cette approche est utilisée par la plupart des gestionnaires de paquets trouvés dans les distributions commerciales. Les exemples de gestionnaires qui suivent cette approche sont RPM (qui est parfois requis par la *Spécification de base de Linux Standard*), pkg-utils, apt de Debian, et le système de portage de Gentoo. Une astuce décrivant comment adopter ce style de gestion de paquets pour les systèmes LFS se trouve à *http://www.fr.linuxfromscratch.org/view/astuces/fakeroot-fr.txt*.

La création de fichiers de paquet qui incluent des informations de dépendance est complexe et va au-delà de l'objectif de LFS.

Slackware utilise un système basé sur **tar** pour les archives de paquets. Ce système ne gère volontairement pas les dépendances de paquets car d'autres gestionnaires de paquets plus complexes le font. Pour des détails sur la gestion de paquets, voir *http://www.slackbook.org/html/package-management.html*.

8.2.2.7. Gestion basée sur les utilisateurs

Cette méthode, unique à LFS, a été décrite par Matthias Benkmann et est disponible sur le *Projet des astuces*. Dans cette méthode, chaque paquet est installé en tant qu'utilisateur séparé dans les emplacements standards. Les fichiers appartenant à un paquet sont facilement identifiés grâce à l'identifiant de l'utilisateur. Les avantages et inconvénients de cette approche sont trop complexes pour les décrire dans cette section. Pour plus de détails, voir l'astuce sur *http://www.fr.linuxfromscratch.org/view/astuces/gestionnaire-paquets-utilisateur.txt*.

8.2.3. Déployer LFS sur plusieurs systèmes

Un des avantages du système LFS est qu'il n'y a pas de fichiers dépendant de la position des fichiers sur un système de disque. Cloner la construction d'un système LFS sur un autre ordinateur avec une architecture similaire au système de base est aussi facile que l'utilisation de **tar** sur la partition LFS qui contient le répertoire racine (environ 250Mo décompressés pour une construction LFS de base), en copiant ce fichier via un transfert par réseau ou par CD-ROM vers le nouveau système et en le décompressant. À partir de là, vous devrez modifier quelques fichiers de configuration. Les fichiers de configuration que vous pouvez devoir mettre à jour comprennent : `/etc/hosts`, `/etc/fstab`, `/etc/passwd`, `/etc/group`, `/etc/shadow` et `/etc/ld.so.conf`.

Vous pouvez construire un noyau personnalisé pour le nouveau système, selon les différences du matériel du système avec la configuration du noyau initial.

Note

Il y a eu quelques rapports de problèmes lors de la copie entre architectures similaires mais non identiques. Par exemple, l'ensemble d'instructions pour l'architecture Intel n'est pas identique avec celle pour un processeur AMD et les versions plus récentes de certains processeurs peuvent avoir des instructions qui ne sont pas disponibles pour des versions antérieures.

Enfin, vous devez rendre le nouveau système amorçable via Section 10.4, « Utiliser GRUB pour paramétrer le processus de démarrage ».

8.3. Man-pages-5.10

Le paquet Man-pages contient environ 2 200 pages de manuel.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 4.7 Mo

8.3.1. Installation de Man-pages

Installez Man-pages en lançant :

```
make install
```

8.3.2. Contenu de Man-pages

Fichiers installés: différentes pages de manuel

Descriptions courtes

`pages man` Décrivent les fonctions C et C++, les fichiers périphériques importants et des fichiers de configuration significatifs

8.4. Iana-Etc-20210202

Le paquet Iana-Etc fournit des données pour les services et protocoles réseau.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 4.7 Mo

8.4.1. Installation de Iana-Etc

Pour ce paquet, nous avons uniquement besoin de copier les fichiers à leur place :

```
cp services protocols /etc
```

8.4.2. Contenu de Iana-Etc

Fichiers installés: /etc/protocols et /etc/services

Descriptions courtes

`/etc/protocols` Décrit les différents protocoles Internet DARPA disponibles à partir du sous-système TCP/IP

`/etc/services` Fournit une correspondance entre des noms de services internet et leurs numéros de ports et types de protocoles affectés

8.5. Glibc-2.33

Le paquet Glibc contient la bibliothèque C principale. Cette bibliothèque fournit toutes les routines basiques pour allouer de la mémoire, rechercher des répertoires, ouvrir et fermer des fichiers, les lire et les écrire, gérer les chaînes, faire correspondre des modèles, faire de l'arithmétique et ainsi de suite.

Temps de construction approximatif: 19 SBU
Espace disque requis: 2.5 Go

8.5.1. Installation de Glibc

Certains programmes de Glibc utilisent le répertoire non conforme au FHS `/var/db` pour stocker leurs données d'exécution. Appliquez le correctif suivant pour que ces programmes stockent leurs données à des endroits respectant le FHS :

```
patch -Np1 -i ../glibc-2.33-fhs-1.patch
```

Corrigez un bogue qui pose problème avec les applications dans le chroot :

```
sed -e '402a\      *result = local->data.services[database_index];' \
    -i nss/nss_database.c
```

La documentation de Glibc recommande de construire Glibc dans un répertoire de construction dédié :

```
mkdir -v build
cd       build
```

Préparez la compilation de Glibc :

```
../configure --prefix=/usr                            \
             --disable-werror                         \
             --enable-kernel=3.2                      \
             --enable-stack-protector=strong          \
             --with-headers=/usr/include              \
             libc_cv_slibdir=/lib
```

Voici la signification des options de configure :

`--disable-werror`
Cette option désactive l'option -Werror passée à GCC. Ceci est nécessaire pour lancer la suite de tests.

`--enable-kernel=3.2`
Cette option dit au système de construction que cette glibc peut être utilisée avec les noyaux au plus aussi vieux que 3.2. Cela signifie générer des contournements au cas où on ne peut pas utiliser un appel système introduit dans une version ultérieure.

`--enable-stack-protector=strong`
Cette option augmente la sécurité du système en ajoutant du code supplémentaire pour repérer les dépassements de tampon comme dans les attaques par la pile.

`--with-headers=/usr/include`
Cette option dit au système de construction où se trouvent les en-têtes de l'API du noyau.

`libc_cv_slibdir=/lib`
Cette variable indique la bibliothèque correcte pour chaque système. Nous ne voulons pas utiliser lib64.

`libc_cv_include_x86_isa_level=no`
: Cette option désactive la propriété « x86 ISA needed » dans les bibliothèques de Glibc. Utilisez-la **si** vous construisez Glibc avec l'option `-march` dans les `CFLAGS`, pour contourner un problème avec Glibc-2.33 qui la casse.

Compilez le paquet :

```
make
```

Important

Dans cette section, la suite de tests de Glibc est considérée comme critique. Ne la sautez sous aucun prétexte.

En général, quelques tests ne réussissent pas, mais vous pouvez le plus souvent ignorer les échecs listés ci-dessous.

```
make check
```

Vous verrez probablement quelques échecs lors des tests. La suite de tests de Glibc est quelque peu dépendante du système hôte. Voici une liste des problèmes les plus fréquents dans certaines versions de LFS :

- *io/tst-lchmod* est connu pour échouer dans l'environnement chroot de LFS.
- *misc/tst-ttyname* est connu pour échouer dans l'environnement chroot de LFS.
- *elf/tst-cpu-features-cpuinfo* peut échouer sur certaines architectures.
- Le test *nss/tst-nss-files-hosts-multi* peut échouer pour des raisons non encore déterminées.
- Les tests *rt/tst-cputimer{1,2,3}* dépendent du noyau du système hôte. Les noyaux 4.14.91–4.14.96, 4.19.13–4.19.18 et 4.20.0–4.20.5 sont connus pour causer des échecs sur ces tests.
- Les tests de math échouent parfois sur des systèmes où le processeur n'est pas un intel relativement récent ou un AMD.

Bien que ce ne soit qu'un simple message, l'étape d'installation de Glibc se plaindra de l'absence de `/etc/ld.so.conf`. Supprimez ce message d'avertissement avec :

```
touch /etc/ld.so.conf
```

Corrigez le Makefile généré pour passer un test de cohérence inutile qui échoue dans l'environnement partiel de LFS :

```
sed '/test-installation/s@$(PERL)@echo not running@' -i ../Makefile
```

Installez le paquet :

```
make install
```

Installez le fichier de configuration et le répertoire d'exécution de **nscd** :

```
cp -v ../nscd/nscd.conf /etc/nscd.conf
mkdir -pv /var/cache/nscd
```

Installez les fichiers de prise en charge de **nscd** pour systemd :

```
install -v -Dm644 ../nscd/nscd.tmpfiles /usr/lib/tmpfiles.d/nscd.conf
install -v -Dm644 ../nscd/nscd.service /lib/systemd/system/nscd.service
```

Ensuite, installez les locales qui permettent à votre système de répondre en une langue différente. Aucune n'est indispensable, mais si certaines sont absentes, les suites de test des futurs paquets peuvent sauter des situations de test importantes.

Vous pouvez installer les locales individuelles en utilisant le programme **localedef** Par exemple, la première commande **localedef** ci-dessous combine la définition de la locale du codage indépendant `/usr/share/i18n/locales/cs_CZ` avec la définition de la page de codes `/usr/share/i18n/charmaps/UTF-8.gz` et envoie le résultat vers le fichier `/usr/lib/locale/locale-archive`. Les instructions suivantes installeront les paramètres minimums des locales nécessaires pour le déroulement optimal des tests :

```
mkdir -pv /usr/lib/locale
localedef -i POSIX -f UTF-8 C.UTF-8 2> /dev/null || true
localedef -i cs_CZ -f UTF-8 cs_CZ.UTF-8
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i de_DE -f UTF-8 de_DE.UTF-8
localedef -i el_GR -f ISO-8859-7 el_GR
localedef -i en_GB -f UTF-8 en_GB.UTF-8
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i it_IT -f UTF-8 it_IT.UTF-8
localedef -i ja_JP -f EUC-JP ja_JP
localedef -i ja_JP -f SHIFT_JIS ja_JP.SIJS 2> /dev/null || true
localedef -i ja_JP -f UTF-8 ja_JP.UTF-8
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i tr_TR -f UTF-8 tr_TR.UTF-8
localedef -i zh_CN -f GB18030 zh_CN.GB18030
localedef -i zh_HK -f BIG5-HKSCS zh_HK.BIG5-HKSCS
```

En outre, installez la locale de votre pays, de votre langue et de votre codage.

Vous pouvez alternativement installer les locales listées dans le fichier `glibc-2.33/localedata/SUPPORTED` (il inclut toutes les locales citées ci-dessus et d'autres) en une fois avec la commande suivante qui prend beaucoup de temps :

```
make localedata/install-locales
```

Puis utilisez la commande **localedef** pour créer et installer les locales non listées dans le fichier `glibc-2.33/localedata/SUPPORTED` dans le cas peu probable où vous en auriez besoin.

Note

Glibc utilise maintenant libidn2 lors de la résolution de noms de domaines internationalisés. C'est une dépendance à l'exécution. Si cette fonctionnalité est requise, les instructions pour installer libidn2 se trouvent sur la *page libidn2 de BLFS*.

8.5.2. Configurer Glibc

8.5.2.1. Ajout de nsswitch.conf

Le fichier `/etc/nsswitch.conf` doit être créé parce que les valeurs par défaut de Glibc ne fonctionnent pas bien dans un environnement en réseau.

Créez un nouveau fichier `/etc/nsswitch.conf` en lançant ce qui suit :

```
cat > /etc/nsswitch.conf << "EOF"
# Begin /etc/nsswitch.conf

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

protocols: files
services: files
ethers: files
rpc: files

# End /etc/nsswitch.conf
EOF
```

8.5.2.2. Ajout des données de fuseaux horaires

Installez et configurez les données de fuseaux horaires avec ce qui suit :

```
tar -xf ../../tzdata2021a.tar.gz

ZONEINFO=/usr/share/zoneinfo
mkdir -pv $ZONEINFO/{posix,right}

for tz in etcetera southamerica northamerica europe africa antarctica  \
          asia australasia backward; do
    zic -L /dev/null   -d $ZONEINFO       ${tz}
    zic -L /dev/null   -d $ZONEINFO/posix ${tz}
    zic -L leapseconds -d $ZONEINFO/right ${tz}
done

cp -v zone.tab zone1970.tab iso3166.tab $ZONEINFO
zic -d $ZONEINFO -p America/New_York
unset ZONEINFO
```

Voici la signification de la commande zic :

`zic -L /dev/null ...`

Ceci crée des fuseaux horaires posix, sans secondes intercalaires. Par convention, on met cela dans `zoneinfo` et dans `zoneinfo/posix`. Il faut mettre les fuseaux horaires POSIX dans `zoneinfo`, sinon diverses suites de tests renverront des erreurs. Sur un système embarqué, où il y a peu de place et vous ne souhaitez pas

mettre à jour les fuseaux horaires, vous pouvez économiser 1,9 Mo en n'utilisant pas le répertoire `posix`, mais certaines applications ou suites de tests pourraient ne pas donner de bons résultats.

`zic -L leapseconds ...`

Ceci crée de bons fuseaux horaires incluant les secondes intercalaires. Sur un système embarqué, où il y a peu de place et vous ne souhaitez pas mettre à jour les fuseaux horaires, ou si vous vous moquez de la bonne heure, vous pouvez économiser 1.9Mio en ne mettant pas de répertoire `right`.

`zic ... -p ...`

Ceci crée le fichier `posixrules`. Nous utilisons New York car POSIX exige des règles temporelles d'enregistrement à jour quotidiennement pour respecter les règles américaines.

Une façon de déterminer dans quel fuseau horaire vous vous situez consiste à lancer le script suivant :

```
tzselect
```

Après avoir répondu à quelques questions sur votre emplacement, le script affichera le nom du fuseau horaire (quelque chose comme *Europe/Paris*). Il y a aussi d'autres fuseaux horaires listés dans `/usr/share/zoneinfo` comme *America/Montreal* ou *EST5EDT* qui ne sont pas identifiés par le script mais qui peuvent être utilisés.

Puis créez le fichier `/etc/localtime` en lançant :

```
ln -sfv /usr/share/zoneinfo/<xxx> /etc/localtime
```

Remplacez *<xxx>* par le nom du fuseau horaire sélectionné (par exemple America/Montreal).

8.5.2.3. Configurer le chargeur dynamique

Par défaut, le chargeur dynamique (`/lib/ld-linux.so.2`) cherche dans `/lib` et `/usr/lib` les bibliothèques partagées nécessaires aux programmes lors de leur exécution. Néanmoins, s'il existe des bibliothèques dans d'autres répertoires que `/lib` et `/usr/lib`, leur emplacement doit être ajouté dans le fichier `/etc/ld.so.conf` pour que le chargeur dynamique les trouve. `/usr/local/lib` et `/opt/lib` sont deux répertoires connus pour contenir des bibliothèques supplémentaires, donc ajoutez ces deux répertoires au chemin de recherche du chargeur dynamique.

Créez un nouveau fichier `/etc/ld.so.conf` en lançant ce qui suit :

```
cat > /etc/ld.so.conf << "EOF"
# Début de /etc/ld.so.conf
/usr/local/lib
/opt/lib

EOF
```

Si vous le désirez, le chargeur dynamique peut également chercher un répertoire et inclure le contenu de fichiers qui s'y trouvent. Les fichiers de ce répertoire include sont en général constitués d'une ligne spécifiant le chemin vers la bibliothèque désirée. Pour ajouter cette possibilité, lancez les commandes suivantes :

```
cat >> /etc/ld.so.conf << "EOF"
# Ajout d'un répertoire include
include /etc/ld.so.conf.d/*.conf

EOF
mkdir -pv /etc/ld.so.conf.d
```

8.5.3. Contenu de Glibc

Programmes installés: catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, makedb, mtrace, nscd, pcprofiledump, pldd, sln, sotruss, sprof, tzselect, xtrace, zdump et zic

Bibliothèques installées: ld-2.33.so, libBrokenLocale.{a,so}, libSegFault.so, libanl.{a,so}, libc.{a,so}, libc_nonshared.a, libcrypt.{a,so}, libdl.{a,so}, libg.a, libm.{a,so}, libmcheck.a, libmemusage.so, libmvec.{a,so}, libnsl.{a,so}, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libpcprofile.so, libpthread.{a,so}, libpthread_nonshared.a, libresolv.{a,so}, librt.{a,so}, libthread_db.so et libutil.{a,so}

Répertoires installés: /usr/include/arpa, /usr/include/bits, /usr/include/gnu, /usr/include/net, /usr/include/netash, /usr/include/netatalk, /usr/include/netax25, /usr/include/neteconet, /usr/include/netinet, /usr/include/netipx, /usr/include/netiucv, /usr/include/netpacket, /usr/include/netrom, /usr/include/netrose, /usr/include/nfs, /usr/include/protocols, /usr/include/rpc, /usr/include/sys, /usr/lib/audit, /usr/lib/gconv, /usr/lib/locale, /usr/libexec/getconf, /usr/share/i18n, /usr/share/zoneinfo, /var/cache/nscd et /var/lib/nss_db

Descriptions courtes

catchsegv Peut être utilisé pour créer une trace de la pile lorsqu'un programme s'arrête avec une erreur de segmentation

gencat Génère des catalogues de messages

getconf Affiche les valeurs de configuration du système pour les variables spécifiques du système de fichiers

getent Récupère les entrées à partir d'une base de données administrative

iconv Réalise une conversion de l'ensemble des caractères

iconvconfig Crée des fichiers de configuration pour le module **iconv**

ldconfig Configure les liens du chargeur dynamique

ldd Indique les bibliothèques partagées requises pour chaque programme ou bibliothèque partagée

lddlibc4 Assiste **ldd** avec des fichiers objets

locale Affiche diverses informations sur la locale courante

localedef Compile les spécifications de locale

makedb Crée une base de données simple à partir d'une entrée textuelle

mtrace Lit et interprète un fichier de trace mémoire et affiche un résumé dans un format lisible par un humain

nscd Un démon pour les services de noms fournissant un cache pour les requêtes les plus communes

pcprofiledump Affiche des informations générées par un profilage du PC

pldd Liste les objets dynamiques partagés utilisés en exécutant des processus

sln Un programme **ln** lié statiquement

sotruss Retrace les procédures d'appel d'une bibliothèque partagée vers une commande indiquée

sprof Lit et affiche les données de profilage des objets partagés

tzselect	Demande à l'utilisateur l'emplacement géographique du système et donne la description du fuseau horaire correspondante
xtrace	Trace l'exécution d'un programme en affichant la fonction en cours d'exécution
zdump	Afficheur de fuseau horaire
zic	Compilateur de fuseau horaire
`ld-2.33.so`	Le programme d'aide des bibliothèques partagées exécutables
`libBrokenLocale`	Utilisé en interne par Glibc comme une arme grossière pour résoudre les locales cassées (comme certaines applications Motif). Voir les commentaires dans `glibc-2.33/locale/broken_cur_max.c` pour plus d'informations
`libSegFault`	Un gestionnaire de signaux d'erreurs de segmentation, utilisé par **catchsegv**
`libanl`	Une bibliothèque asynchrone de recherche de noms
`libc`	La principale bibliothèque C
`libcrypt`	La bibliothèque de chiffrement
`libdl`	La bibliothèque de l'interface du chargeur dynamique
`libg`	Bibliothèque factice ne contenant aucune fonction. C'était auparavant une bibliothèque d'exécution pour **g++**
`libm`	La bibliothèque mathématique
`libmcheck`	Active le test d'allocation de mémoire lorsqu'on y relie quelque chose
`libmemusage`	Utilisé par **memusage** pour aider à la récupération d'informations sur l'utilisation de la mémoire par un programme
`libnsl`	La bibliothèque de services réseau
`libnss`	Les bibliothèques « Name Service Switch », contenant des fonctions de résolution de noms d'hôtes, de noms d'utilisateurs, de noms de groupes, d'alias, de services, de protocoles et ainsi de suite.
`libpcprofile`	Peut être préchargé pour profiler le PC d'un exécutable
`libpthread`	La bibliothèque threads POSIX
`libresolv`	Contient des fonctions de création, d'envoi et d'interprétation de paquets pour les serveurs de noms de domaine Internet
`librt`	Contient des fonctions fournissant la plupart des interfaces spécifiées par l'extension temps réel de POSIX.1b
`libthread_db`	Contient des fonctions utiles pour construire des débogueurs de programmes multi-threads
`libutil`	Contient du code pour les fonctions « standard » utilisées par de nombreux outils Unix

8.6. Zlib-1.2.11

Le paquet Zlib contient des routines de compression et décompression utilisées par quelques programmes.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 5.0 Mo

8.6.1. Installation de Zlib

Préparez la compilation de Zlib :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

La bibliothèque partagée doit être déplacée vers le dossier `/lib`, et par conséquent le fichier `.so` dans `/usr/lib` devra être recréé :

```
mv -v /usr/lib/libz.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libz.so) /usr/lib/libz.so
```

Supprimez une bibliothèque statique inutile :

```
rm -fv /usr/lib/libz.a
```

8.6.2. Contenu de Zlib

Bibliothèques installées: libz.so

Descriptions courtes

`libz` Contient des fonctions de compression et décompression utilisées par quelques programmes

8.7. Bzip2-1.0.8

Le paquet Bzip2 contient des programmes de compression et décompression de fichiers. Compresser des fichiers texte avec **bzip2** permet d'atteindre un taux de compression bien meilleur qu'avec l'outil **gzip**.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 7.5 Mo

8.7.1. Installation de Bzip2

Appliquez un correctif qui installera la documentation de ce paquet :

```
patch -Np1 -i ../bzip2-1.0.8-install_docs-1.patch
```

La commande suivante garantit l'installation de liens symboliques relatifs :

```
sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
```

Assurez-vous que les pages de manuel s'installent au bon endroit :

```
sed -i "s@(PREFIX)/man@(PREFIX)/share/man@g" Makefile
```

Préparez la compilation de Bzip2 avec :

```
make -f Makefile-libbz2_so
make clean
```

Voici la signification du paramètre de make :

`-f Makefile-libbz2_so`
Ceci fera que Bzip2 sera construit en utilisant un fichier `makefile` différent, dans ce cas le fichier `Makefile-libbz2_so` qui crée une bibliothèque `libbz2.so` dynamique et lie les outils Bzip2 avec.

Compilez et testez le paquet :

```
make
```

Installez les programmes :

```
make PREFIX=/usr install
```

Installez le binaire dynamique **bzip2** dans le répertoire `/bin`, créez les liens symboliques nécessaires et nettoyez :

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

Supprimez une bibliothèque statique inutile :

```
rm -fv /usr/lib/libbz2.a
```

8.7.2. Contenu de Bzip2

Programmes installés: bunzip2 (lien vers bzip2), bzcat (lien vers bzip2), bzcmp (lien vers bzdiff), bzdiff, bzegrep (lien vers bzgrep), bzfgrep (lien vers bzgrep), bzgrep, bzip2, bzip2recover, bzless (lien vers bzmore) et bzmore
Bibliothèques installées: libbz2.so
Répertoire installé: /usr/share/doc/bzip2-1.0.8

Descriptions courtes

bunzip2	Décompresse les fichiers compressés avec bzip
bzcat	Décompresse vers la sortie standard
bzcmp	Lance **cmp** sur des fichiers compressés avec bzip
bzdiff	Lance **diff** sur des fichiers compressés avec bzip
bzegrep	Lance **egrep** sur des fichiers compressés avec bzip
bzfgrep	Lance **fgrep** sur des fichiers compressés avec bzip
bzgrep	Lance **grep** sur des fichiers compressés avec bzip
bzip2	Compresse les fichiers en utilisant l'algorithme de compression de texte par tri de blocs de Burrows-Wheeler avec le codage Huffman ; le taux de compression est meilleur que celui auquel parviennent les outils de compression plus conventionnels utilisant les algorithmes « Lempel-Ziv », comme **gzip**
bzip2recover	Essaie de récupérer des données à partir de fichiers endommagés, compressés avec bzip
bzless	Lance **less** sur des fichiers compressés avec bzip
bzmore	Lance **more** sur des fichiers compressés avec bzip
`libbz2`	La bibliothèque implémentant la compression de données sans perte par tri de blocs, utilisant l'algorithme de Burrows-Wheeler

8.8. Xz-5.2.5

Le paquet Xz contient des programmes de compression et de décompression de fichiers. Il offre les possibilités des formats lzma et des formats de compression récents. La compression de fichiers textes avec **xz** donne un meilleur pourcentage de compression qu'avec les commandes **gzip** ou **bzip2** traditionnelles.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 15 Mo

8.8.1. Installation de Xz

Préparez la compilation de Xz :

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.2.5
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet et assurez-vous que tous les fichiers essentiels sont dans le bon répertoire :

```
make install
mv -v   /usr/bin/{lzma,unlzma,lzcat,xz,unxz,xzcat} /bin
mv -v /usr/lib/liblzma.so.* /lib
ln -svf ../../lib/$(readlink /usr/lib/liblzma.so) /usr/lib/liblzma.so
```

8.8.2. Contenu de Xz

Programmes installés: lzcat (lien vers xz), lzcmp (lien vers xzdiff), lzdiff (lien vers xzdiff), lzegrep (lien vers xzgrep), lzfgrep (lien vers xzgrep), lzgrep (lien vers xzgrep), lzless (lien vers xzless), lzma (lien vers xz), lzmadec, lzmainfo, lzmore (lien vers xzmore), unlzma (lien vers xz), unxz (lien vers xz), xz, xzcat (lien vers xz), xzcmp (lien vers xzdiff), xzdec, xzdiff, xzegrep (lien vers xzgrep), xzfgrep (lien vers xzgrep), xzgrep, xzless et xzmore
Bibliothèques installées: liblzma.so
Répertoires installés: /usr/include/lzma et /usr/share/doc/xz-5.2.5

Descriptions courtes

lzcat Décompresse sur la sortie standard
lzcmp Lance **cmp** sur des fichiers LZMA compressés
lzdiff Lance **diff** sur des fichiers LZMA compressés
lzegrep Lance **grep** sur des fichiers LZMA compressés
lzfgrep Lance **fgrep** sur des fichiers LZMA compressés
lzgrep Lance **grep** sur des fichiers LZMA compressés
lzless Lance **less** sur des fichiers LZMA compressés
lzma Compresse ou décompresse des fichiers en utilisant le format LZMA

lzmadec Un décodeur petit et rapide pour des fichiers LZMA compressés

lzmainfo Affiche les informations contenues dans l'en-tête du fichier LZMA compressé

lzmore Lance **more** sur des fichiers LZMA compressés

unlzma Décompresse des fichiers en utilisant le format LZMA

unxz Décompresse des fichiers en utilisant le format XZ

xz Compresse ou décompresse des fichiers en utilisant le format XZ

xzcat Décompresse sur la sortie standard

xzcmp Lance **cmp** sur des fichiers Xz compressés

xzdec Un décodeur petit et rapide pour des fichiers compressés XZ

xzdiff Lance **diff** sur des fichiers LZMA compressés

xzegrep Lance **egrep** sur des fichiers XZ compressés

xzfgrep Lance **fgrep** sur des fichiers XZ compressés

xzgrep Lance **grep** sur des fichiers XZ compressés

xzless Lance **less** sur des fichiers XZ compressés

xzmore Lance **more** sur des fichiers XZ compressés

`liblzma` La bibliothèque qui implémente la compression sans perte, de données rangées par blocs, utilisant les algorithmes de la chaîne Lempel-Ziv-Markov

8.9. Zstd-1.4.8

Zstandard est un algorithme de compression en temps réel qui fournit des ratios de compression élevés. Il propose une très large gamme de rapports entre compression et vitesse tout en étant soutenu par un décodeur très rapide.

Temps de construction approximatif: 1.1 SBU
Espace disque requis: 59 Mo

8.9.1. Installation de Zstd

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make prefix=/usr install
```

Supprimez une bibliothèque statique et déplacez la bibliothèque partagée dans `/lib`. Le fichier `.so` dans `/usr/lib` doit aussi être recréé :

```
rm -v /usr/lib/libzstd.a
mv -v /usr/lib/libzstd.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libzstd.so) /usr/lib/libzstd.so
```

8.9.2. Contenu de Zstd

Programmes installés: zstd, zstdcat (lien vers zstd), zstdgrep, zstdless, zstdmt (lien vers zstd) et unzstd (lien vers zstd)
Bibliothèques installées: libzstd.so

Descriptions courtes

zstd Compresse ou décompresse des fichiers avec le format ZSTD
zstdgrep Lance **grep** sur des fichiers compressés avec ZSTD
zstdless Lance **less** sur des fichiers compressés avec ZSTD
`libzstd` La bibliothèque implémentant la compression de données sans perte, avec l'algorithme ZSTD

8.10. File-5.39

Le paquet File contient un outil pour déterminer le type d'un fichier ou des fichiers donnés.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 14 Mo

8.10.1. Installation de File

Préparez la compilation de File :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.10.2. Contenu de File

Programmes installés: file
Bibliothèque installée: libmagic.so

Descriptions courtes

file Tente de classifier chaque fichier donné. Il réalise ceci en exécutant différents tests—tests sur le système de fichiers, tests des nombres magiques et tests de langages

`libmagic` Contient des routines pour la reconnaissance de nombres magiques utilisés par le programme **file**

8.11. Readline-8.1

Le paquet Readline est un ensemble de bibliothèques qui offrent des fonctionnalités d'édition de la ligne de commande et d'historique.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 16 Mo

8.11.1. Installation de Readline

La réinstallation de Readline aura pour conséquence que les vieilles bibliothèques seront déplacées vers <nom_bibliotheque>.old. Même si cela n'est pas normalement un problème, cela peut dans certains cas provoquer un bogue de lien dans **ldconfig**. Cela peut être évité en effectuant les deux seds suivants :

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

Préparez la compilation de Readline :

```
./configure --prefix=/usr    \
            --disable-static \
            --with-curses    \
            --docdir=/usr/share/doc/readline-8.1
```

Voici la signification de l'option de configure :

`--with-curses`
Cette option dit à Readline qu'il peut trouver les fonctions de la bibliothèque termcap dans la bibliothèque curses, au lieu d'une bibliothèque termcap séparée. Elle permet aussi de générer un fichier `readline.pc` correct.

Compilez le paquet :

```
make SHLIB_LIBS="-lncursesw"
```

Voici la signification de l'option de make :

`SHLIB_LIBS="-lncursesw"`
Cette option force Readline à se lier à la bibliothèque `libncursesw`.

Ce paquet n'est pas fourni avec une suite de tests.

Installez le paquet :

```
make SHLIB_LIBS="-lncursesw" install
```

Maintenant, déplacez les bibliothèques dynamiques à un endroit plus adéquat et corrigez certains liens symboliques :

```
mv -v /usr/lib/lib{readline,history}.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libreadline.so) /usr/lib/libreadline.so
ln -sfv ../../lib/$(readlink /usr/lib/libhistory.so ) /usr/lib/libhistory.so
```

Si désiré, installez la documentation :

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.1
```

8.11.2. Contenu de Readline

Bibliothèques installées: libhistory.so et libreadline.so
Répertoires installés: /usr/include/readline et /usr/share/doc/readline-8.1

Descriptions courtes

`libhistory` Fournit une interface utilisateur cohérente pour rappeler des lignes dans l'historique

`libreadline` Fournit un ensemble de commandes pour manipuler du texte entré dans une session interactive d'un programme

8.12. M4-1.4.18

Le paquet M4 contient un processeur de macros.

Temps de construction approximatif: 0.4 SBU
Espace disque requis: 32 Mo

8.12.1. Installation de M4

Tout d'abord, effectuez quelques corrections requises avec glibc-2.28 et supérieur :

```
sed -i 's/IO_ftrylockfile/IO_EOF_SEEN/' lib/*.c
echo "#define _IO_IN_BACKUP 0x100" >> lib/stdio-impl.h
```

Préparez la compilation de M4 :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.12.2. Contenu de M4

Programme installé: m4

Descriptions courtes

m4 Copie les fichiers donnés tout en résolvant les macros qu'ils contiennent. Ces macros sont soit internes soit définies par l'utilisateur et peuvent prendre un nombre illimité d'arguments. En plus de la simple expansion de macros, **m4** dispose de fonctions pour inclure des fichiers nommés, lancer des commandes Unix, faire des opérations arithmétiques, manipuler du texte, pour la récursion et ainsi de suite. Le programme **m4** peut être utilisé soit comme interface d'un compilateur soit comme évaluateur de macros à part.

8.13. Bc-3.3.0

Le paquet Bc contient un langage de traitement des nombres à la précision de votre choix.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 6.7 Mo

8.13.1. Installation de Bc

Prépare la compilation de Bc :

```
PREFIX=/usr CC=gcc ./configure.sh -G -O3
```

Voici la signification des options de configure :

CC=gcc"
Ces paramètres spécifient le compilateur à utiliser.

-O3
Spécifie le niveau d'optimisation à utiliser.

-G
Évite certaines parties de la suite de tests qui ne fonctionnent pas sans une version installée de GNU bc.

Compilez le paquet :

```
make
```

Pour tester bc, lancez :

```
make test
```

Installez le paquet :

```
make install
```

8.13.2. Contenu de Bc

Programmes installés: bc et dc

Descriptions courtes

bc Une calculatrice en ligne de commandes

dc Une calculatrice en ligne de commande en polonais inversé (reverse-polish)

8.14. Flex-2.6.4

Le paquet Flex contient un outil de génération de programmes reconnaissant des modèles de texte.

Temps de construction approximatif: 0.4 SBU
Espace disque requis: 36 Mo

8.14.1. Installation de Flex

Préparez la compilation de Flex :

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/flex-2.6.4 \
            --disable-static
```

Compilez le paquet :

```
make
```

Pour tester les résultats (environ 0.5 SBU), lancez :

```
make check
```

Installez le paquet :

```
make install
```

Quelques programmes ne connaissent pas encore **flex** et essaient de lancer son prédécesseur, **lex**. Pour ces programmes, créez un lien symbolique nommé `lex` lançant `flex` en mode d'émulation **lex** :

```
ln -s flex /usr/bin/lex
```

8.14.2. Contenu de Flex

Programmes installés: flex, flex++ (lien vers flex), et lex (lien vers flex)
Bibliothèques installées: libfl.so
Répertoire installé: /usr/share/doc/flex-2.6.4

Descriptions courtes

flex Un outil pour générer des programmes reconnaissant des modèles dans un texte ; cela permet une grande diversité pour spécifier les règles de recherche de modèle, éradiquant ainsi le besoin de développer un programme spécialisé

flex++ Une extension de flex, est utilisée pour générer du code et des classes C++. C'est un lien symbolique vers **flex**

lex Un lien symbolique qui exécute **flex** en mode d'émulation **lex**

`libfl` La bibliothèque `flex`

8.15. Tcl-8.6.11

Le paquet Tcl contient le langage de commande des outils, un langage de script robuste. Le paquet Expect est écrit en Tcl.

Temps de construction approximatif: 3.8 SBU
Espace disque requis: 80 Mo

8.15.1. Installation de Tcl

Ce paquet et les deux suivants (Expect et DejaGNU) sont installés pour prendre en charge le lancement des suites de tests de binutils, GCC et d'autres paquets. Installer trois paquets pour les tests peut sembler excessif, mais c'est toujours rassurant, sinon essentiel, de savoir que les outils les plus importants fonctionnent correctement.

Tout d'abord, déballez la documentation en lançant la commande suivante :

```
tar -xf ../tcl8.6.11-html.tar.gz --strip-components=1
```

Préparez la compilation de Tcl :

```
SRCDIR=$(pwd)
cd unix
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            $([ "$(uname -m)" = x86_64 ] && echo --enable-64bit)
```

Voici la signification des options de configure :

`$([ "$(uname -m)" = x86_64 ] && echo --enable-64bit)`
La construction *`$(<commande shell>)`* est remplacée par la sortie de commande shell. Ici cette sortie est vide si vous lancez sur une machine 32 bit et *`--enable-64bit`* si vous la lancez sur une machine 64 bits.

Construisez le paquet :

```
make

sed -e "s|$SRCDIR/unix|/usr/lib|" \
    -e "s|$SRCDIR|/usr/include|"  \
    -i tclConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/tdbc1.1.2|/usr/lib/tdbc1.1.2|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.2/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/tdbc1.1.2/library|/usr/lib/tcl8.6|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.2|/usr/include|"            \
    -i pkgs/tdbc1.1.2/tdbcConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/itcl4.2.1|/usr/lib/itcl4.2.1|" \
    -e "s|$SRCDIR/pkgs/itcl4.2.1/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/itcl4.2.1|/usr/include|"            \
    -i pkgs/itcl4.2.1/itclConfig.sh

unset SRCDIR
```

Les diverses instructions « sed » après la commande « make » suppriment des références au répertoire de construction des fichiers de configuration et les remplace par les répertoires d'installation. Ce n'est pas requis pour le reste de LFS, mais peut être requis pour un paquet construit plus tard avec Tcl.

Pour tester les résultats, lancez :

```
make test
```

Note

Dans les résultats des tests, il y a plusieurs endroits associés à clock.test qui indiquent un échec, mais le résumé à la fin n'indique aucun échec. clock.test fonctionne sur un système LFS complet.

Installez le paquet :

```
make install
```

Rendez la bibliothèque installée réinscriptible pour que les symboles de débogages puissent être supprimés plus tard :

```
chmod -v u+w /usr/lib/libtcl8.6.so
```

Installez les en-têtes de Tcl. Le paquet suivant, Expect, en a besoin.

```
make install-private-headers
```

Maintenant créez un lien symbolique nécessaire :

```
ln -sfv tclsh8.6 /usr/bin/tclsh
```

Enfin, renommez une page de manuel qui entre en conflit avec une page de manuel de Perl :

```
mv /usr/share/man/man3/{Thread,Tcl_Thread}.3
```

8.15.2. Contenu de Tcl

Programmes installés: tclsh (lien vers tclsh8.6) et tclsh8.6
Bibliothèque installée: libtcl8.6.so et libtclstub8.6.a

Descriptions courtes

tclsh8.6 Le shell de commande de Tcl
tclsh Un lien vers tclsh8.6
`libtcl8.6.so` La bibliothèque Tcl
`libtclstub8.6.a` La bibliothèque de base de Tcl

8.16. Expect-5.45.4

Le paquet Expect contient des outils pour automatiser, via des dialogues scriptés, des applications interactives comme **telnet**, **ftp**, **passwd**, **fsck**, **rlogin** et **tip**. Expect est aussi utile pour tester ces mêmes applications et faciliter toutes sortes de taches qui sont trop compliquées avec quoi que ce soit d'autre. La boîte à outils DejaGnu est écrit en Expect.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 3.9 Mo

8.16.1. Installation de Expect

Préparez la compilation d'Expect :

```
./configure --prefix=/usr           \
            --with-tcl=/usr/lib     \
            --enable-shared         \
            --mandir=/usr/share/man \
            --with-tclinclude=/usr/include
```

Voici la signification des options de configure :

`--with-tcl=/usr/lib`
Ce paramètre est requis pour dire à **configure** où le script **tclConfig.sh** se trouve.

`--with-tclinclude=/usr/include`
Cela dit explicitement à Expect où trouver les en-têtes internes de Tcl.

Construisez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make test
```

Installez-le :

```
make install
ln -svf expect5.45.4/libexpect5.45.4.so /usr/lib
```

8.16.2. Contenu d'Expect

Programme installé: expect
Bibliothèque installée: libexpect-5.45.so

Descriptions courtes

expect Communique avec les autres programmes interactifs suivant un script.

`libexpect-5.45.so` Contient des fonctions qui permettent à Expect d'être utilisé comme une extension Tcl ou d'être utilisé directement à partir du langage C ou du langage C++ (sans Tcl)

8.17. DejaGNU-1.6.2

Le paquet DejaGnu contient un ensemble de travail pour lancer les suites de tests d'outils GNU. Il est écrit en **expect**, qui lui-même utilise Tcl (langage de commande des outils).

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 4.6 Mo

8.17.1. Installation de DejaGNU

Préparez DejaGNU à la compilation :

```
./configure --prefix=/usr
makeinfo --html --no-split -o doc/dejagnu.html doc/dejagnu.texi
makeinfo --plaintext       -o doc/dejagnu.txt  doc/dejagnu.texi
```

Construisez et installez le paquet :

```
make install
install -v -dm755  /usr/share/doc/dejagnu-1.6.2
install -v -m644   doc/dejagnu.{html,txt} /usr/share/doc/dejagnu-1.6.2
```

Pour tester les résultats, lancez :

```
make check
```

8.17.2. Contenu de DejaGNU

Programme installé: runtest

Descriptions courtes

runtest Un script enveloppe qui repère le bon shell **expect** puis lance DejaGNU

8.18. Binutils-2.36.1

Le paquet Binutils contient un éditeur de liens, un assembleur et d'autres outils pour gérer des fichiers objets.

Temps de construction approximatif: 6.2 SBU
Espace disque requis: 4.9 Go

8.18.1. Installation de Binutils

Vérifiez que les pseudo-terminaux (PTY) fonctionnent correctement dans l'environnement en effectuant un simple test :

```
expect -c "spawn ls"
```

Cette commande devrait afficher ce qui suit :

```
spawn ls
```

Si, à la place, la sortie affiche le message ci-dessous, c'est que l'environnement n'est pas paramétré pour la bonne opération PTY. Vous devez résoudre ce problème avant de lancer les suites de test de Binutils et de GCC :

```
The system has no more ptys.
Ask your system administrator to create more.
```

Maintenant supprimez un test qui empêche les tests de continuer jusqu'à la fin :

```
sed -i '/@\tincremental_copy/d' gold/testsuite/Makefile.in
```

La documentation de Binutils recommande de construire Binutils dans un répertoire de construction dédié :

```
mkdir -v build
cd       build
```

Préparez la compilation de Binutils :

```
../configure --prefix=/usr       \
             --enable-gold       \
             --enable-ld=default \
             --enable-plugins    \
             --enable-shared     \
             --disable-werror    \
             --enable-64-bit-bfd \
             --with-system-zlib
```

Voici la signification des options de configure :

`--enable-gold`
Construit l'éditeur de liens gold et l'installe en tant que ld.gold (à côté de l'éditeur de liens par défaut).

`--enable-ld=default`
Construit l'éditeur de liens bdf original et l'installe à côté de ld (l'éditeur par défaut) et ld.bfd.

`--enable-plugins`
Active la prise en charge des greffons pour l'éditeur de lien.

`--enable-64-bit-bfd`
Active la prise en charge 64-bits (sur les systèmes avec une taille de mot plus petite). Elle n'est pas forcément requise sur les systèmes 64 bits, mais elle ne fait pas de mal.

`--with-system-zlib`
Utilise la version installée de la bibliothèque zlib plutôt que de construire la version incluse.

Compilez le paquet :

```
make tooldir=/usr
```

Voici la signification des options de configure :

`tooldir=/usr`
Normalement, le répertoire tooldir (où seront placés les exécutables) est configuré avec `$(exec_prefix)/$(target_alias)`. Par exemple, les machines x86_64 l'étendront en `/usr/x86_64-unknown-linux-gnu`. Comme il s'agit d'un système personnalisé, nous n'avons pas besoin d'un répertoire spécifique à notre cible dans `/usr`. `$(exec_prefix)/$(target_alias)` serait utilisée si le système avait pour but une cross-compilation (par exemple, compiler un paquet sur une machine Intel qui génère du code pouvant être exécuté sur des machines PowerPC).

Important

La suite de tests de Binutils dans cette section est considérée comme critique. Ne la sautez sous aucun prétexte.

Testez les résultats :

```
make -k check
```

Quatre tests nommés « Run property … » sont connus pour échouer.

Installez le paquet :

```
make tooldir=/usr install
```

Supprimez des bibliothèques statiques inutiles :

```
rm -fv /usr/lib/lib{bfd,ctf,ctf-nobfd,opcodes}.a
```

8.18.2. Contenu de Binutils

Programmes installés:	addr2line, ar, as, c++filt, dwp, elfedit, gprof, ld, ld.bfd, ld.gold, nm, objcopy, objdump, ranlib, readelf, size, strings et strip
Bibliothèques installées:	libbfd.so, libctf.so, libctf-nobfd.so et libopcodes.so
Répertoire installé:	/usr/lib/ldscripts

Descriptions courtes

addr2line	Traduit les adresses de programme en noms de fichier et numéros de ligne ; suivant une adresse et le nom d'un exécutable, il utilise les informations de débogage disponibles dans l'exécutable pour déterminer le fichier source et le numéro de ligne associé à cette adresse
ar	Crée, modifie et extrait à partir d'archives
as	Un assembleur qui assemble la sortie de **gcc** en un fichier objet
c++filt	Utilisé par l'éditeur de liens pour récupérer les symboles C++ et Java, et pour empêcher les fonctions surchargées d'arrêter brutalement le programme
dwp	L'utilitaire d'empaquetage DWARF
elfedit	Met à jour l'en-tête ELF des fichiers ELF

gprof	Affiche les données de profilage d'appels dans un graphe
ld	Un éditeur de liens combinant un certain nombre d'objets et de fichiers archives en un seul fichier, en déplaçant leurs données et en regroupant les références de symboles
ld.gold	Une version réduite de ld qui ne prend en charge que le format de fichier elf
ld.bfd	Lien dur vers **ld**
nm	Liste les symboles disponibles dans un fichier objet
objcopy	Traduit un type de fichier objet en un autre
objdump	Affiche des informations sur le fichier objet donné, les options contrôlant les informations à afficher ; l'information affichée est surtout utile aux programmeurs qui travaillent sur les outils de compilation
ranlib	Génère un index du contenu d'une archive et le stocke dans l'archive ; l'index liste tous les symboles définis par les membres de l'archive qui sont des fichiers objet déplaçables
readelf	Affiche des informations sur les binaires du type ELF
size	Liste les tailles des sections et la taille totale pour les fichiers objets donnés
strings	Affiche, pour chaque fichier donné, la séquence de caractères affichables qui sont d'au moins la taille spécifiée (par défaut, 4) ; pour les fichiers objets, il affiche, par défaut, seulement les chaînes des sections d'initialisation et de chargement alors que pour les autres types de fichiers, il parcourt le fichier entier
strip	Supprime les symboles des fichiers objets
`libbfd`	Bibliothèque des descripteurs de fichiers binaires
`libctf`	la bibliothèque de prise en charge du format de débogage compatible ANSI-C
`libctf-nobfd`	Une variante de libctf qui n'utilise par la fonctionnalité libbfd
`libopcodes`	Une bibliothèque de gestion des opcodes—la « version lisible » des instructions du processeur ; elle est utilisée pour construire des outils comme **objdump**

8.19. GMP-6.2.1

Le paquet GMP contient des bibliothèques de maths. Elles contiennent des fonctions utiles pour l'arithmétique à précision arbitraire.

Temps de construction approximatif: 1.0 SBU
Espace disque requis: 52 Mo

8.19.1. Installation de GMP

Note

Si vous construisez pour un x86 32 bits, mais si vous avez un processeur capable d'exécuter du code 64 bits *et* si vous avez spécifié `CFLAGS` dans l'environnement, le script configure va essayer de configurer pour du 64 bits et va échouer. Évitez cela en invoquant la commande configure ci-dessous avec

```
ABI=32 ./configure ...
```

Note

Les paramètres par défaut de GMP produisent des bibliothèques optimisées pour le processeur de l'hôte. Si vous souhaitez obtenir des bibliothèques convenables pour des processeurs moins puissants, vous pouvez créer des bibliothèques génériques comme suit :

```
cp -v configfsf.guess config.guess
cp -v configfsf.sub   config.sub
```

Préparez la compilation de GMP :

```
./configure --prefix=/usr    \
            --enable-cxx     \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.2.1
```

Voici la signification des nouvelles options de configure :

`--enable-cxx`
Ce paramètre active la prise en charge de C++

`--docdir=/usr/share/doc/gmp-6.2.1`
Cette variable indique le bon emplacement de la documentation.

Compilez le paquet et générez la documentation HTML :

```
make
make html
```

Important

La suite de tests de GMP dans cette section est considérée comme critique. Ne la sautez en aucun cas.

Testez les résultats :

```
make check 2>&1 | tee gmp-check-log
```

Attention

Le code de gmp est hautement optimisé pour le processeur sur lequel il est construit. Parfois, le code chargé de détecter le processeur identifie mal les capacités du système et produira des erreurs dans les tests ou d'autres applications utilisant les bibliothèques gmp avec le message « Illegal instruction ». Dans ce cas, gmp devrait être reconfiguré avec l'option --build=x86_64-unknown-linux-gnu et reconstruit.

Assurez-vous que les 197 tests de la suite de tests réussissent tous. Vérifiez les résultats en lançant la commande suivante :

```
awk '/# PASS:/{total+=$3} ; END{print total}' gmp-check-log
```

Installez le paquet et sa documentation :

```
make install
make install-html
```

8.19.2. Contenu de GMP

Bibliothèques installées: libgmp.so et libgmpxx.so
Répertoire installé: /usr/share/doc/gmp-6.2.1

Descriptions courtes

`libgmp` Contient les fonctions de maths de précision

`libgmpxx` Contient des fonctions de maths de précision pour C++

8.20. MPFR-4.1.0

Le paquet MPFR contient des fonctions pour des maths à précision multiple.

Temps de construction approximatif: 0.8 SBU
Espace disque requis: 38 Mo

8.20.1. Installation de MPFR

Préparez la compilation de MPFR :

```
./configure --prefix=/usr        \
            --disable-static     \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.1.0
```

Compilez le paquet et générez la documentation HTML :

```
make
make html
```

Important

La suite de tests de MPFR est considérée comme critique. Ne la sautez en aucun cas.

Testez les résultats et assurez-vous que tous les tests ont réussi :

```
make check
```

Installez le paquet et sa documentation :

```
make install
make install-html
```

8.20.2. Contenu de MPFR

Bibliothèques installées: libmpfr.so
Répertoire installé: /usr/share/doc/mpfr-4.1.0

Descriptions courtes

`libmpfr` Contient des fonctions de maths à précision multiple

8.21. MPC-1.2.1

Le paquet MPC contient une bibliothèque pour le calcul arithmétique de nombres complexes avec une haute précision au choix et l'arrondissement correct du résultat.

Temps de construction approximatif: 0.3 SBU
Espace disque requis: 22 Mo

8.21.1. Installation de MPC

Préparez la compilation de MPC :

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.2.1
```

Compilez le paquet :

```
make
make html
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
make install-html
```

8.21.2. Contenu de MPC

Bibliothèques installées: libmpc.so
Dossiers installés: /usr/share/doc/mpc-1.2.1

Descriptions courtes

`libmpc` Contient des fonctions mathématiques complexes

8.22. Attr-2.4.48

Le paquet attr contient des outils d'administration des attributs étendus des objets du système de fichier.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 4.2 Mo

8.22.1. Installation d'Attr

Préparez la compilation d'Attr :

```
./configure --prefix=/usr     \
            --disable-static  \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.4.48
```

Compilez le paquet :

```
make
```

Il faut lancer les tests sur un système de fichiers qui prend en charge les attributs étendus, comme les systèmes de fichiers ext2, ext3, ou ext4. Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

Il faut supprimer la bibliothèque partagée de `/lib`, et donc, recréer le fichier `.so` dans `/usr/lib` :

```
mv -v /usr/lib/libattr.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libattr.so) /usr/lib/libattr.so
```

8.22.2. Contenu d'Attr

Programmes installés: attr, getfattr et setfattr
Bibliothèque installée: libattr.so
Répertoires installés: /usr/include/attr et /usr/share/doc/attr-2.4.48

Descriptions courtes

attr Etend les attributs étendus des objets d'un système de fichiers
getfattr Affiche les attributs étendus des objets d'un système de fichiers
setfattr Définit les attributs étendus des objets d'un système de fichiers
`libattr` Contient la bibliothèque de fonction pour la manipulation des attributs étendu

8.23. Acl-2.2.53

Le paquet Acl contient des outils d'administration des Access Control Lists (listes de contrôle d'accès) qui sont utilisés pour définir plus finement des droits d'accès de votre choix aux fichiers et aux répertoires.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 2.8 Mo

8.23.1. Installation d'Acl

Préparez la compilation d'Acl :

```
./configure --prefix=/usr         \
            --disable-static      \
            --libexecdir=/usr/lib \
            --docdir=/usr/share/doc/acl-2.2.53
```

Compilez le paquet :

```
make
```

Il faut lancer les tests d'Acl sur un système de fichiers qui prend en charge les contrôles d'accès après la construction de Coreutils avec les bibliothèques Acl. Si vous le souhaitez, revenez à ce paquet et lancez **make check** après avoir construit Coreutils plus loin dans ce chapitre.

Installez le paquet :

```
make install
```

Il faut déplacer la bibliothèque partagée vers `/lib`, et donc, recréer le fichier `.so` dans `/usr/lib` :

```
mv -v /usr/lib/libacl.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libacl.so) /usr/lib/libacl.so
```

8.23.2. Contenu d'Acl

Programmes installés: chacl, getfacl et setfacl
Bibliothèques installées: libacl.so
Répertoires installés: /usr/include/acl et /usr/share/doc/acl-2.2.53

Descriptions courtes

chacl Modifie la liste de contrôle d'accès d'un fichier ou d'un répertoire

getfacl Donne les listes de contrôle des accès à un fichier

setfacl Définit les listes de contrôle d'accès à un fichier

`libacl` Contient la bibliothèque de fonction pour la manipulation de Access Control Lists

8.24. Libcap-2.48

Le paquet Libcap implémente les interfaces du niveau utilisateur avec les fonctions POSIX 1003.1e disponibles dans les noyaux Linux. Ces possibilités établissent le partage des pouvoirs avec les privilèges root dans un ensemble de droits distincts.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 45 Mo

8.24.1. Installation de Libcap

Évitez que des bibliothèques statiques ne soient installées :

```
sed -i '/install -m.*STA/d' libcap/Makefile
```

Compilez le paquet :

```
make prefix=/usr lib=lib
```

Voici la signification de l'option make :

`lib=lib`
Ce paramètre fait en sorte que la bibliothèque soit installée dans `/usr/lib` plutôt que dans `/usr/lib64` sur x86_64. Il n'a aucun effet sur x86.

Pour tester les résultats, lancez :

```
make test
```

Installez le paquet et assurez-vous que les bibliothèques essentielles sont dans le bon répertoire :

```
make prefix=/usr lib=lib install
for libname in cap psx; do
    mv -v /usr/lib/lib${libname}.so.* /lib
    ln -sfv ../../lib/lib${libname}.so.2 /usr/lib/lib${libname}.so
    chmod -v 755 /lib/lib${libname}.so.2.48
done
```

8.24.2. Contenu de Libcap

Programmes installés: capsh, getcap, getpcaps, et setcap
Bibliothèque installée: libcap.so et libpsx.so

Descriptions courtes

capsh Une enveloppe shell pour voir et contraindre la prise en charge de ces possibilités
getcap Examine les possibilités autour d'un fichier
getpcaps Affiche à la demande les possibilités d'un ou plusieurs processus
setcap Définit les possibilités d'un fichier
`libcap` Contient la bibliothèque de fonctions pour manipuler les fonctionnalités de POSIX 1003.1e
`libpsx` Contient des fonctions pour la prise en charge de la sémantique POSIX des appels systèmes associés avec la bibliothèque pthread

8.25. Shadow-4.8.1

Le paquet Shadow contient des programmes de gestion de mots de passe d'une façon sécurisée.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 45 Mo

8.25.1. Installation de Shadow

Note

Si vous souhaitez multiplier l'usage des mots de passe efficaces, reportez-vous à *http://fr.linuxfromscratch.org/blfs/../view/blfs-10.1/postlfs/cracklib.html* pour l'installation de CrackLib avant de compiler Shadow. Puis ajoutez *--with-libcrack* à la commande **configure** ci-dessous.

Désactivez l'installation du programme **groups** et de ses pages de manuel car Coreutils en fournit une meilleure version. Cela empêche aussi l'installation de pages de manuel déjà installées dans Section 8.3, « Man-pages-5.10 » :

```
sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i 's/groups\.1 / /'   {} \;
find man -name Makefile.in -exec sed -i 's/getspnam\.3 / /' {} \;
find man -name Makefile.in -exec sed -i 's/passwd\.5 / /'   {} \;
```

Au lieu d'utiliser la méthode *crypt* par défaut, utilisez la méthode *SHA-512* plus sécurisée du chiffrement de mot de passe, qui autorise aussi les mots de passe plus longs que huit caractères. Il est également nécessaire de changer l'endroit obsolète de `/var/spool/mail` pour les boîtes e-mail de l'utilisateur que Shadow utilise par défaut en l'endroit `/var/mail` utilisé actuellement :

```
sed -e 's:#ENCRYPT_METHOD DES:ENCRYPT_METHOD SHA512:' \
    -e 's:/var/spool/mail:/var/mail:'                 \
    -i etc/login.defs
```

Note

Si vous compilez Shadow avec la prise en charge de Cracklib, lancez ce qui suit :

```
sed -i 's:DICTPATH.*:DICTPATH\t/lib/cracklib/pw_dict:' etc/login.defs
```

Faites un petit changement pour que le premier numéro de groupe généré par useradd soit 1000 :

```
sed -i 's/1000/999/' etc/useradd
```

Préparez la compilation de Shadow :

```
touch /usr/bin/passwd
./configure --sysconfdir=/etc \
            --with-group-name-max-length=32
```

Voici la signification de l'option de configuration :

touch /usr/bin/passwd
Le fichier `/usr/bin/passwd` a besoin d'exister parce que son emplacement est codé en dur dans certains programmes, et l'emplacement par défaut s'il n'existe pas est incorrect.

`--with-group-name-max-length=32`
La longueur maximum d'un nom d'utilisateur est de 32 caractères. Règle un plafond similaire pour les noms de groupes.

Compilez le paquet :

```
make
```

Ce paquet n'est pas fourni avec une suite de tests.

Installez le paquet :

```
make install
```

8.25.2. Configuration de Shadow

Ce paquet contient des outils pour ajouter, modifier, supprimer des utilisateurs et des groupes, initialiser et changer leur mot de passe, et bien d'autres tâches administratives. Pour une explication complète de ce que signifie *password shadowing*, jetez un œil dans le fichier `doc/HOWTO` à l'intérieur du répertoire source. Il reste une chose à garder à l'esprit si vous décidez d'utiliser le support de Shadow : les programmes qui ont besoin de vérifier les mots de passe (gestionnaires d'affichage, programmes FTP, démons pop3 et ainsi de suite) ont besoin d'être compatibles avec shadow, c'est-à-dire qu'ils ont besoin d'être capables de fonctionner avec des mots de passe shadow.

Pour activer les mots de passe shadow, lancez la commande suivante :

```
pwconv
```

Pour activer les mots de passe shadow pour les groupes, lancez :

```
grpconv
```

La configuration fournie avec Shadow pour l'outil **useradd**présente quelques inconvénients qui appellent quelques explications. D'abord, l'action par défaut de l'outil **useradd** est de créer un utilisateur et un groupe du même nom que l'utilisateur. Par défaut les numéros d'ID utilisateur (UID) et d'ID de groupe (GID) commenceront à 1000. Cela signifie que si vous ne passez pas de paramètres à **useradd**, chaque utilisateur sera membre d'un groupe unique sur le système. Si vous ne désirez pas ce comportement, vous devrez passer le paramètre *`-g`* à **useradd**. Les paramètres par défaut sont stockés dans le fichier `/etc/default/useradd`. Il se peut que vous deviez modifier deux paramètres dans ce fichier pour satisfaire vos besoins particuliers.

Explication des paramètres de `/etc/default/useradd`

`GROUP=1000`
Ce paramètre initialise le début des numéros de groupe utilisés dans le fichier /etc/group. Vous pouvez le modifier avec ce que vous désirez. Remarquez que **useradd** ne réutilisera jamais un UID ou un GID. Si le numéro identifié dans ce paramètre est utilisé, il utilisera le numéro disponible suivant celui-ci. Remarquez aussi que si vous n'avez pas de groupe 1000 sur votre système la première fois que vous utilisez **useradd** sans le paramètre *`-g`*, vous obtiendrez un message sur le terminal qui dit : `useradd: unknown GID 1000`. Vous pouvez passer ce message et le numéro de groupe 1000 sera utilisé.

`CREATE_MAIL_SPOOL=yes`
Il résulte de ce paramètre que **useradd** crée un fichier de boîte mail pour le nouvel utilisateur créé. **useradd** rendra le groupe `mail` propriétaire de ce fichier avec les droits 0660. Si vous préférez que **useradd** ne crée pas ces fichiers de boîte mail, lancez la commande suivante :

```
sed -i 's/yes/no/' /etc/default/useradd
```

8.25.3. Configurer le mot de passe de root

Choisissez un mot de passe pour l'utilisateur *root* et configurez-le avec :

```
passwd root
```

8.25.4. Contenu de Shadow

Programmes installés: chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgidmap, newgrp, newuidmap, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (lien vers newgrp), su, useradd, userdel, usermod, vigr (lien vers vipw) et vipw

Répertoire installé: /etc/default

Descriptions courtes

chage Utilisé pour modifier le nombre maximum de jours entre des modifications obligatoires du mot de passe

chfn Utilisé pour modifier le nom complet de l'utilisateur et quelques autres informations

chgpasswd Utilisé pour mettre à jour des mots de passe en lot

chpasswd Utilisé pour mettre à jour les mots de passe utilisateurs en lot

chsh Utilisé pour modifier le shell de connexion par défaut d'un utilisateur

expiry Vérifie et renforce la politique d'expiration des mots de passe

faillog Est utilisé pour examiner les traces d'échecs de connexions, pour configurer le nombre maximum d'échecs avant qu'un compte ne soit bloqué ou pour réinitialiser le nombre d'échecs

gpasswd Est utilisé pour ajouter et supprimer des membres et des administrateurs aux groupes

groupadd Crée un groupe avec le nom donné

groupdel Supprime le groupe ayant le nom donné

groupmems Permet à un utilisateur d'administrer la liste des membres de son groupe sans avoir besoin des privilèges du super utilisateur.

groupmod Est utilisé pour modifier le nom ou le GID du groupe

grpck Vérifie l'intégrité des fichiers `/etc/group` et `/etc/gshadow`

grpconv Crée ou met à jour le fichier shadow à partir du fichier group standard

grpunconv Met à jour `/etc/group` à partir de `/etc/gshadow` puis supprime ce dernier

lastlog Indique les connexions les plus récentes de tous les utilisateurs ou d'un utilisateur donné

login Est utilisé par le système pour permettre aux utilisateurs de se connecter

logoutd Est un démon utilisé pour renforcer les restrictions sur les temps et ports de connexion

newgidmap Est utilisé pour configurer la correspondance des gid d'un espace de nom utilisateur

newgrp Est utilisé pour modifier le GID courant pendant une session de connexion

newuidmap Est utilisé pour configurer la correspondance des uid d'un espace de nom utilisateur

newusers Est utilisé pour créer ou mettre à jour toute une série de comptes utilisateur en une fois

nologin Affiche un message selon lequel un compte n'est pas disponible. Destiné à être utilisé comme shell par défaut pour des comptes qui ont été désactivés

passwd Est utilisé pour modifier le mot de passe d'un utilisateur ou d'un groupe

pwck	Vérifie l'intégrité des fichiers de mots de passe, `/etc/passwd` et `/etc/shadow`
pwconv	Crée ou met à jour le fichier de mots de passe shadow à partir du fichier password habituel
pwunconv	Met à jour `/etc/passwd` à partir de `/etc/shadow` puis supprime ce dernier
sg	Exécute une commande donnée lors de l'initialisation du GID de l'utilisateur à un groupe donné
su	Lance un shell en substituant les ID de l'utilisateur et du groupe
useradd	Crée un nouvel utilisateur avec le nom donné ou met à jour les informations par défaut du nouvel utilisateur
userdel	Supprime le compte utilisateur indiqué
usermod	Est utilisé pour modifier le nom de connexion de l'utilisateur, son UID (*User Identification*, soit Identification Utilisateur), shell, groupe initial, répertoire personnel et ainsi de suite.
vigr	Édite les fichiers `/etc/group` ou `/etc/gshadow`
vipw	Édite les fichiers `/etc/passwd` ou `/etc/shadow`

8.26. GCC-10.2.0

Le paquet GCC contient la collection de compilateurs GNU, qui inclut les compilateurs C et C++.

Temps de construction approximatif: 95 SBU (avec les tests)
Espace disque requis: 4.6 Go

8.26.1. Installation de GCC

Si vous construisez sur x86_64, changez le nom du répertoire par défaut des bibliothèques 64 bits en « lib » :

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
  ;;
esac
```

La documentation de GCC recommande de construire GCC dans un répertoire de construction dédié :

```
mkdir -v build
cd       build
```

Préparez la compilation de GCC :

```
../configure --prefix=/usr            \
             LD=ld                    \
             --enable-languages=c,c++ \
             --disable-multilib       \
             --disable-bootstrap      \
             --with-system-zlib
```

Remarquez que pour d'autres langages, il existe des prérequis qui ne sont pas encore disponibles. Voir le *la page GCC du livre BLFS* pour des instructions sur la manière de construire tous les langages pris en charge par GCC.

Voici la signification des nouvelles options de configure :

`LD=ld`
Ce paramètre assure que le script configure utilisera le ld installé par binutils, construit plus tôt dans ce chapitre, au lieu de la version compilée de manière croisée qui serait sinon utilisée.

`--with-system-zlib`
Ce paramètre dit à GCC de se lier à la copie installée sur le système de la bibliothèque zlib, plutôt qu'à sa propre copie interne.

Compilez le paquet :

```
make
```

Important

Dans cette section, la suite de tests pour GCC est considérée comme critique. Ne les sautez sous aucun prétexte.

Un ensemble de tests dans la suite de tests de GCC est connu pour utiliser toute la pile par défaut, donc augmentez la taille de la pile avant de lancer les tests :

```
ulimit -s 32768
```

Testez les résultats en tant qu'utilisateur non privilégié, mais ne vous arrêtez pas aux erreurs :

```
chown -Rv tester .
su tester -c "PATH=$PATH make -k check"
```

Pour recevoir un résumé des résultats de la suite de tests, lancez :

```
../contrib/test_summary
```

Pour n'avoir que les résumés, redirigez la sortie vers **`grep -A7 Summ`**.

Vous pouvez comparer les résultats avec ceux situés dans *http://www.linuxfromscratch.org/lfs/build-logs/10.1/* et *https://gcc.gnu.org/ml/gcc-testresults/*.

Six tests liés à la variable get_time sont connus pour échouer. Ils sont a priori liés au paramètre régional en_HK.

En plus, les tests suivantes liés aux fichiers suivants sont connus pour échouer avec glibc-2.33 : asan_test.C, co-ret-17-void-ret-coro.C, pr95519-05-gro.C, pr80166.c.

Quelques échecs inattendus sont parfois inévitables. Les développeurs de GCC connaissent généralement ces problèmes, mais ne les ont pas encore résolus. Sauf si les résultats des tests sont très différents de ceux sur l'adresse ci-dessus, vous pouvez continuer en toute sécurité.

Installez le paquet et supprimez un répertoire inutile :

```
make install
rm -rf /usr/lib/gcc/$(gcc -dumpmachine)/10.2.0/include-fixed/bits/
```

Le répertoire de construction de GCC appartient maintenant à `tester` et la propriété du répertoire des en-têtes installé (et son contenu) seront incorrectes. Donnez la propriété à l'utilisateur et au groupe `root` :

```
chown -v -R root:root \
    /usr/lib/gcc/*linux-gnu/10.2.0/include{,-fixed}
```

Créez un lien symbolique requis par le *FHS* pour des raisons « historiques ».

```
ln -sv ../usr/bin/cpp /lib
```

Ajoutez un lien symbolique de compatibilité pour permettre l'optimisation à l'édition des liens (LTO) à la construction de programmes :

```
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/10.2.0/liblto_plugin.so \
        /usr/lib/bfd-plugins/
```

Maintenant que notre chaîne d'outils est en place, il est important de s'assurer à nouveau que la compilation et l'édition de liens fonctionneront comme prévu. Cela se fait en effectuant les mêmes tests de propreté que ceux faits plus haut dans ce chapitre :

```
echo 'int main(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

Il ne devrait pas y avoir d'erreur et la sortie de la commande sera (avec des différences spécifiques aux plateformes dans le nom de l'éditeur de liens) :

```
[Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
```

Maintenant, assurez-vous que nous utilisons les bons fichiers de démarrage :

```
grep -o '/usr/lib.*/crt[1in].*succeeded' dummy.log
```

La sortie de la dernière commande sera :

```
/usr/lib/gcc/x86_64-pc-linux-gnu/10.2.0/../../../../lib/crt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/10.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/10.2.0/../../../../lib/crtn.o succeeded
```

Selon l'architecture de votre machine, le message ci-dessus peut légèrement différer. La différence porte sur le nom du répertoire après `/usr/lib/gcc`. La chose importante à chercher est que **gcc** ait trouvé les trois fichiers `crt*.o` sous le répertoire `/usr/lib`.

Vérifiez que le compilateur cherche les bons fichiers d'en-têtes :

```
grep -B4 '^ /usr/include' dummy.log
```

Cette commande devrait afficher la sortie suivante :

```
#include <...> search starts here:
 /usr/lib/gcc/x86_64-pc-linux-gnu/10.2.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/10.2.0/include-fixed
 /usr/include
```

À nouveau, le nom du répertoire après votre triplet cible peut être différent de celui ci-dessus, selon votre architecture.

Puis, vérifiez que le nouvel éditeur de liens est utilisé avec les bons chemins de recherche :

```
grep 'SEARCH.*/usr/lib' dummy.log |sed 's|; |\n|g'
```

Les références vers des emplacements qui contiennent « -linux-gnu » doivent être ignorées, sinon la sortie de la dernière commande doit être :

```
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib64")
SEARCH_DIR("/usr/local/lib64")
SEARCH_DIR("/lib64")
SEARCH_DIR("/usr/lib64")
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Il se peut qu'un système 32 bits voie différemment quelques répertoires. Par exemple, voici la sortie d'une machine i686 :

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib32")
SEARCH_DIR("/usr/local/lib32")
SEARCH_DIR("/lib32")
SEARCH_DIR("/usr/lib32")
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Ensuite, assurez-vous que nous utilisons la bonne libc :

```
grep "/lib.*/libc.so.6 " dummy.log
```

La sortie de la dernière commande sera :

```
attempt to open /lib/libc.so.6 succeeded
```

Assurez-vous que GCC utilise le bon éditeur de liens dynamiques :

```
grep found dummy.log
```

La sortie de la dernière commande devrait être (avec des différences spécifiques aux plateformes dans le nom de l'éditeur de liens) :

```
found ld-linux-x86-64.so.2 at /lib/ld-linux-x86-64.so.2
```

Si la sortie ne ressemble pas à celle montrée ci-dessus ou qu'elle n'apparaît pas du tout, alors quelque chose ne va vraiment pas. Enquêtez et retracez les étapes pour savoir d'où vient le problème et comment le corriger. Tout problème devra être résolu avant de continuer le processus.

Une fois que tout fonctionne correctement, nettoyez les fichiers tests :

```
rm -v dummy.c a.out dummy.log
```

Enfin, déplacez un fichier mal placé :

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

8.26.2. Contenu de GCC

Programmes installés: c++, cc (lien vers gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, gcov, gcov-dump et gcov-tool

Bibliothèques installées: libasan.{a,so}, libatomic.{a,so}, libcc1.so, libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, libitm.{a,so}, liblsan.{a,so}, liblto_plugin.so, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so}, libstdc++fs.a, libsupc++.a, libtsan.{a,so} et libubsan.{a,so}

Répertoires installés: /usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc et /usr/share/gcc-10.2.0

Descriptions courtes

c++ Le compilateur C++

cc Le compilateur C

cpp Le préprocesseur C ; il est utilisé par le compilateur pour l'extension des instructions #include, #define et d'autres instructions similaires dans les fichiers sources

g++ Le compilateur C++

gcc Le compilateur C

gcc-ar Une enveloppe autour de **ar** qui ajoute un greffon à la ligne de commande. Ce programme n'est utilisé que pour ajouter "l'optimisation du temps d'édition des liens" et il n'est pas utile avec les options de construction par défaut

gcc-nm Une enveloppe autour de **nm** qui ajoute un greffon à la ligne de commande. Ce programme n'est utilisé que pour ajouter « l'optimisation à l'édition des liens » et il n'est pas utile avec les options de construction par défaut

gcc-ranlib Une enveloppe autour de **ranlib** qui ajoute un greffon à la ligne de commande. Ce programme n'est utilisé que pour ajouter « l'optimisation à l'édition des liens » et il n'est pas utile avec les options de construction par défaut

gcov	Un outil de tests ; il est utilisé pour analyser les programmes et savoir où des optimisations seraient suivies du plus d'effet
gcov-dump	Outil d'affichage de profil gcda et gcno hors-ligne
gcov-tool	Outil de traitement de profils gcda hors-ligne
`libasan`	La bibliothèque Address Sanitizer à l'exécution
`libatomic`	Bibliothèque de GCC de constructions atomiques pour l'exécution
`libcc1`	La bibliothèque de pré-traitement C
`libgcc`	Contient la prise en charge de **gcc** à l'exécution
`libgcov`	Cette bibliothèque est liée à un programme où on demande à GCC d'activer le profilage
`libgomp`	Implémentation GNU de l'API OpenMP API pour la programmation en mémoire parallèle partagée pour plusieurs plateformes en C/C++ et Fortran
`liblsan`	La bibliothèque Leak Sanitizer à l'exécution
`liblto_plugin`	Greffon d'optimisation du temps d'édition des liens de GCC (LTO) permettant à GCC de pratiquer des optimisations tout au cours des unités de compilation
`libquadmath`	API de la bibliothèque de maths GCC de précision au carré
`libssp`	Contient des routines qui prennent en charge la fonctionnalité de protection de GCC contre les débordements de mémoire
`libstdc++`	La bibliothèque C++ standard
`libstdc++fs`	Bibliothèque de systèmes de fichiers ISO/IEC TS 18822:2015
`libsupc++`	Fournit des routines de prise en charge pour le langage de programmation C++
`libtsan`	La bibliothèque Thread Sanitizer à l'exécution
`libubsan`	La bibliothèque Undefined Behavior Sanitizer à l'exécution

8.27. Pkg-config-0.29.2

Le paquet pkg-config contient un outil pour passer le chemin include et les chemins des bibliothèques afin de construire les outils au moment de l'exécution de configure et de make.

Temps de construction approximatif: 0.3 SBU
Espace disque requis: 30 Mo

8.27.1. Installation de Pkg-config

Préparez la compilation de Pkg-config :

```
./configure --prefix=/usr              \
            --with-internal-glib       \
            --disable-host-tool        \
            --docdir=/usr/share/doc/pkg-config-0.29.2
```

Voici la signification des nouvelles options de configure :

`--with-internal-glib`
Cela permettra à pkg-config d'utiliser sa version interne de Glib car une version externe n'est pas disponible dans LFS.

`--disable-host-tool`
Cette option désactive la création d'un lien en dur non souhaité vers le programme pkg-config.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.27.2. Contenu de Pkg-config

Programme installé: pkg-config
Répertoire installé: /usr/share/doc/pkg-config-0.29.2

Descriptions courtes

pkg-config Retourne des méta-informations sur la bibliothèque ou le paquet spécifié

8.28. Ncurses-6.2

Le paquet Ncurses contient les bibliothèques de gestion des écrans type caractère, indépendant des terminaux.

Temps de construction approximatif: 0.4 SBU
Espace disque requis: 34 Mo

8.28.1. Installation de Ncurses

Préparez la compilation de Ncurses :

```
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            --with-shared           \
            --without-debug         \
            --without-normal        \
            --enable-pc-files       \
            --enable-widec
```

Voici la signification des nouvelles options de configure :

`--enable-widec`
Cette option amène les bibliothèques « wide-character » (comme `libncursesw.so.6.2`) à être compilée au lieu de celles normales (comme `libncurses.so.6.2`). Ces bibliothèques « wide-character » sont utilisables à la fois en locales multibyte et 8-bit traditionnelles, alors que les bibliothèques normales ne fonctionnent correctement que dans les locales 8-bit. Les bibliothèques « Wide-character » et normales sont compatibles entre leurs sources mais pas entre leurs binaires.

`--enable-pc-files`
Ce paramètre génère et installe les fichiers .pc pour pkg-config.

`--without-normal`
Ce drapeau désactive la construction et l'installation de la plupart des bibliothèques statiques.

Compilez le paquet :

```
make
```

Ce paquet a une suite de tests, mais elle ne peut être exécutée qu'après l'installation du paquet. Les tests se situent dans le répertoire `test/`. Voir le fichier `README` dans ce répertoire pour de plus amples détails.

Installez le paquet :

```
make install
```

Déplacez les bibliothèques partagées dans le répertoire `/lib`, où elles sont supposées être :

```
mv -v /usr/lib/libncursesw.so.6* /lib
```

Comme les bibliothèques ont été déplacées, un lien symbolique pointe vers un fichier inexistant. Recréez-le :

```
ln -sfv ../../lib/$(readlink /usr/lib/libncursesw.so) /usr/lib/libncursesw.so
```

Beaucoup d'applications s'attendent encore à ce que l'éditeur de liens puisse trouver les bibliothèques Ncurses non wide-character. Faites croire à de telles applications au lien vers les bibliothèques wide-character par des faux liens symboliques et des scripts d'éditeur de liens :

```
for lib in ncurses form panel menu ; do
    rm -vf                    /usr/lib/lib${lib}.so
    echo "INPUT(-l${lib}w)" > /usr/lib/lib${lib}.so
    ln -sfv ${lib}w.pc        /usr/lib/pkgconfig/${lib}.pc
done
```

Finalement, assurez-vous que les vieilles applications qui cherchent `-lcurses` lors de la compilation sont encore compilables :

```
rm -vf                     /usr/lib/libcursesw.so
echo "INPUT(-lncursesw)" > /usr/lib/libcursesw.so
ln -sfv libncurses.so      /usr/lib/libcurses.so
```

Supprimez une bibliothèque statique qui n'est pas prise en charge par le configure :

```
rm -fv /usr/lib/libncurses++w.a
```

Si désiré, installez la documentation de Ncurses :

```
mkdir -v       /usr/share/doc/ncurses-6.2
cp -v -R doc/* /usr/share/doc/ncurses-6.2
```

Note

Les instructions ci-dessus ne créent pas de bibliothèques Ncurses non-wide-character puisqu'aucun paquet installé par la compilation à partir des sources ne se lie à elles lors de l'exécution. Pour le moment, les seules applications binaires sans sources connues qui se lient à Ncurses non-wide-character exigent la version 5. Si vous devez avoir de telles bibliothèques à cause d'une application disponible uniquement en binaire ou pour vous conformer à la LSB, compilez à nouveau le paquet avec les commandes suivantes :

```
make distclean
./configure --prefix=/usr    \
            --with-shared    \
            --without-normal \
            --without-debug  \
            --without-cxx-binding \
            --with-abi-version=5
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

8.28.2. Contenu de Ncurses

Programmes installés: captoinfo (lien vers tic), clear, infocmp, infotocap (lien vers tic), ncursesw6-config, reset (lien vers tset), tabs, tic, toe, tput et tset

Bibliothèques installées: libcursesw.so (lien symbolique et script de l'éditeur de liens vers libncursesw.so), llibformw.so, libmenuw.so, libncursesw.so, libpanelw.so, ainsi que leurs équivalents non « non-wide » avec un nom identique, mais sans le « w ».

Répertoires installés: /usr/share/tabset, /usr/share/terminfo et /usr/share/doc/ncurses-6.2

Descriptions courtes

captoinfo Convertit une description termcap en description terminfo

clear	Efface l'écran si possible
infocmp	Compare ou affiche les descriptions terminfo
infotocap	Convertit une description terminfo en description termcap
ncursesw6-config	Fournit des informations de configuration de ncurses
reset	Réinitialise un terminal avec ses valeurs par défaut
tabs	Efface et initialise des taquets de tab sur un terminal
tic	Le compilateur d'entrée de description terminfo, traduisant un fichier terminfo au format source dans un format binaire nécessaire pour les routines des bibliothèques ncurses. Un fichier terminfo contient des informations sur les capacités d'un terminal particulier
toe	Liste tous les types de terminaux disponibles, donnant pour chacun d'entre eux son nom principal et sa description
tput	Rend les valeurs de capacités dépendant du terminal disponibles au shell ; il peut aussi être utilisé pour réinitialiser un terminal ou pour afficher son nom long
tset	Peut être utilisé pour initialiser des terminaux
`libcursesw`	Un lien vers `libncursesw`
`libncursesw`	Contient des fonctions pour afficher du texte de plusieurs façons compliquées sur un écran de terminal ; un bon exemple d'utilisation de ces fonctions est le menu affiché par le **make menuconfig** du noyau
`libformw`	Contient des fonctions pour implémenter des formes
`libmenuw`	Contient des fonctions pour implémenter des menus
`libpanelw`	Contient des fonctions pour implémenter des panneaux

8.29. Sed-4.8

Le paquet Sed contient un éditeur de flux.

Temps de construction approximatif:	0.5 SBU
Espace disque requis:	32 Mo

8.29.1. Installation de Sed

Préparez la compilation de Sed :

```
./configure --prefix=/usr --bindir=/bin
```

Compilez le paquet et générez la documentation HTML :

```
make
make html
```

Pour tester les résultats, lancez :

```
chown -Rv tester .
su tester -c "PATH=$PATH make check"
```

Installez le paquet et sa documentation HTML :

```
make install
install -d -m755           /usr/share/doc/sed-4.8
install -m644 doc/sed.html /usr/share/doc/sed-4.8
```

8.29.2. Contenu de Sed

Programme installé:	sed
Répertoire installé:	/usr/share/doc/sed-4.8

Description courte

sed	Filtre et transforme des fichiers texte en une seule passe

8.30. Psmisc-23.4

Le paquet Psmisc contient des programmes pour afficher des informations sur les processus en cours d'exécution.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 5.7 Mo

8.30.1. Installation de Psmisc

Préparez la compilation de Psmisc pour :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Ce paquet n'est pas fourni avec une suite de tests.

Installez le paquet :

```
make install
```

Enfin, déplacez les programmes **killall** et **fuser** à l'endroit spécifié par la FHS :

```
mv -v /usr/bin/fuser   /bin
mv -v /usr/bin/killall /bin
```

8.30.2. Contenu de Psmisc

Programmes installés: fuser, killall, peekfd, prtstat, pslog, pstree et pstree.x11 (lien vers pstree)

Descriptions courtes

fuser Indique les PID de processus utilisant les fichiers ou systèmes de fichiers donnés
killall Tue les processus suivant leur nom. Il envoie un signal à tous les processus en cours
peekfd Observe les descripteurs d'un processus en cours d'exécution, selon son PID
prtstat Affiche des informations sur un processus
pslog Rapport le chemin du journal actuel d'un processus
pstree Affiche les processus en cours hiérarchiquement
pstree.x11 Identique à **pstree**, si ce n'est qu'il attend une confirmation avant de quitter

8.31. Gettext-0.21

Le paquet Gettext contient des outils pour l'internationalisation et la localisation. Ceci permet aux programmes d'être compilés avec la prise en charge des langues natives (*Native Language Support* ou NLS), pour afficher des messages dans la langue native de l'utilisateur.

Temps de construction approximatif: 2.9 SBU
Espace disque requis: 240 Mo

8.31.1. Installation de Gettext

Préparez la compilation de Gettext :

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.21
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

8.31.2. Contenu de Gettext

Programmes installés: autopoint, envsubst, gettext, gettext.sh, gettextize, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin et xgettext
Bibliothèques installées: libasprintf.so, libgettextlib.so, libgettextpo.so, libgettextsrc.so, libtextstyle.so et preloadable_libintl.so
Répertoires installés: /usr/lib/gettext, /usr/share/doc/gettext-0.21, /usr/share/gettext et /usr/share/gettext-0.19.8

Descriptions courtes

autopoint Copie les fichiers d'infrastructure standard gettext en un paquet source

envsubst Substitue les variables d'environnement dans des chaînes de format shell

gettext Traduit un message en langue naturelle dans la langue de l'utilisateur en recherchant la traduction dans un catalogue de messages

gettext.sh Sert en priorité de bibliothèque de fonction shell pour gettext

gettextize Copie tous les fichiers standard Gettext dans le répertoire de haut niveau d'un paquet, pour commencer son internationalisation

msgattrib Filtre les messages d'un catalogue de traduction suivant leurs attributs et manipule les attributs

msgcat	Concatène et fusionne les fichiers `.po`
msgcmp	Compare deux fichiers `.po` pour vérifier que les deux contiennent le même ensemble de chaînes msgid
msgcomm	Trouve les messages qui sont communs aux fichiers `.po` donnés
msgconv	Convertit un catalogue de traduction en un autre codage de caractères
msgen	Crée un catalogue de traduction anglais
msgexec	Applique une commande pour toutes les traductions d'un catalogue de traduction
msgfilter	Applique un filtre à toutes les traductions d'un catalogue de traductions
msgfmt	Génère un catalogue binaire de messages à partir d'un catalogue de traductions
msggrep	Extrait tous les messages d'un catalogue de traductions correspondant à un modèle donné ou appartenant à d'autres sources données
msginit	Crée un nouveau fichier `.po`, initialise l'environnement de l'utilisateur
msgmerge	Combine deux traductions brutes en un seul fichier
msgunfmt	Décompile un catalogue de messages binaires en un texte brut de la traduction
msguniq	Unifie les traductions dupliquées en un catalogue de traduction
ngettext	Affiche les traductions dans la langue native d'un message texte dont la forme grammaticale dépend d'un nombre
recode-sr-latin	Recode du texte serbe de l'écrit cyrillique au latin
xgettext	Extrait les lignes de messages traduisibles à partir des fichiers source donnés pour réaliser la première traduction de modèle
`libasprintf`	définit la classe *autosprintf* qui rend les routines de sortie formatée C utilisables dans les programmes C++ pour utiliser les chaînes de <*string*> et les flux de <*iostream*>
`libgettextlib`	une bibliothèque privée contenant les routines communes utilisées par les nombreux programmes Gettext. Ils ne sont pas faits pour une utilisation générale
`libgettextpo`	Utilisé pour écrire les programmes spécialisés qui s'occupent des fichiers `.po`. Cette bibliothèque est utilisée lorsque les applications standards livrées avec Gettext ne vont pas suffire (comme **msgcomm**, **msgcmp**, **msgattrib** et **msgen**)
`libgettextsrc`	Une bibliothèque privée contenant les routines communes utilisées par les nombreux programmes gettext. Elles ne sont pas destinées à une utilisation générale
`libtextstyle`	Bibliothèque de mise en forme de texte
`preloadable_libintl`	Une bibliothèque faite pour être utilisée par LD_PRELOAD et qui aide `libintl` à archiver des messages non traduits.

8.32. Bison-3.7.5

Le paquet Bison contient un générateur d'analyseurs.

Temps de construction approximatif:	6.4 SBU
Espace disque requis:	56 Mo

8.32.1. Installation de Bison

Préparez la compilation de Bison :

```
./configure --prefix=/usr --docdir=/usr/share/doc/bison-3.7.5
```

Compilez le paquet :

```
make
```

Pour tester les résultats (environ 5,5 SBU), lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.32.2. Contenu de Bison

Programmes installés:	bison et yacc
Bibliothèque installée:	liby.a
Répertoire installé:	/usr/share/bison

Descriptions courtes

bison	Génère, à partir d'une série de règles, un programme d'analyse de structure de fichiers texte ; Bison est un remplacement pour Yacc (Yet Another Compiler Compiler)
yacc	Un emballage pour **bison**, utile pour les programmes qui appellent toujours **yacc** au lieu de **bison** ; il appelle **bison** avec l'option *-y*
`liby`	La bibliothèque Yacc contenant des implémentations, compatible Yacc, des fonctions `yyerror` et `main` ; cette bibliothèque n'est généralement pas très utile mais POSIX la réclame

8.33. Grep-3.6

Le paquet Grep contient des programmes de recherche dans le contenu de fichiers.

Temps de construction approximatif: 0.8 SBU
Espace disque requis: 38 Mo

8.33.1. Installation de Grep

Préparez la compilation de Grep :

```
./configure --prefix=/usr --bindir=/bin
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.33.2. Contenu de Grep

Programmes installés: egrep, fgrep et grep

Descriptions courtes

egrep Affiche les lignes correspondant à une expression rationnelle étendue
fgrep Affiche des lignes correspondant à une liste de chaînes fixes
grep Affiche des lignes correspondant à une expression rationnelle basique

8.34. Bash-5.1

Le paquet Bash contient le shell Bourne-Again.

Temps de construction approximatif: 1.6 SBU
Espace disque requis: 51 Mo

8.34.1. Installation de Bash

Tout d'abord, corrigez une situation de compétition si vous utilisez plusieurs cœurs :

```
sed -i  '/^bashline.o:.*shmbchar.h/a bashline.o: ${DEFDIR}/builtext.h' Makefile.
```

Préparez la compilation de Bash :

```
./configure --prefix=/usr                      \
            --docdir=/usr/share/doc/bash-5.1 \
            --without-bash-malloc              \
            --with-installed-readline
```

Voici la signification de la nouvelle option de configure :

`--with-installed-readline`
Cette option indique à Bash d'utiliser la bibliothèque `readline` sur le système plutôt que d'utiliser sa propre version de readline.

Compilez le paquet :

```
make
```

Sautez à « Installation du paquet » si vous n'exécutez pas la suite de test.

Pour préparer les tests, assurez-vous que l'utilisateur `tester` peut écrire dans l'arborescence des sources :

```
chown -Rv tester .
```

Maintenant, lancez les tests en tant qu'utilisateur `tester` :

```
su tester << EOF
PATH=$PATH make tests < $(tty)
EOF
```

Installez le paquet et déplacez l'exécutable dans `/bin`:

```
make install
mv -vf /usr/bin/bash /bin
```

Lancez le programme **bash** nouvellement compilé (en remplaçant celui en cours d'exécution) :

```
exec /bin/bash --login +h
```

Note

Les paramètres utilisés font que le processus **bash** lance un shell de connexion interactif et désactive le hachage, de façon à ce que les nouveaux programmes soient découverts au fur et à mesure de leur disponibilité.

8.34.2. Contenu de Bash

Programmes installés: bash, bashbug et sh (lien vers bash)
Répertoire installé: /usr/include/bash, /usr/lib/bash et /usr/share/doc/bash-5.1

Descriptions courtes

bash Un interpréteur de commandes largement utilisé ; il réalise un grand nombre d'expansions et de substitutions sur une ligne de commande donnée avant de l'exécuter, rendant cet interpréteur très puissant

bashbug Un script shell pour aider l'utilisateur à composer et à envoyer des courriers électroniques contenant des rapports de bogues spécialement formatés concernant **bash**

sh Un lien symbolique vers le programme **bash** ; à son appel en tant que **sh**, **bash** essaie de copier le comportement initial des versions historiques de **sh** aussi fidèlement que possible, tout en se conformant aussi au standard POSIX

8.35. Libtool-2.4.6

Le paquet Libtool contient le script de prise en charge générique des bibliothèques de GNU. Il cache la complexité d'utilisation de bibliothèques partagées dans une interface cohérente et portable.

Temps de construction approximatif: 1.6 SBU
Espace disque requis: 43 Mo

8.35.1. Installation de Libtool

Préparez la compilation de Libtool :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Note

Le temps de test de libtool peut être réduit significativement sur un système avec plusieurs cœurs. Pour ce faire, ajoutez **TESTSUITEFLAGS=-j<N>** à la ligne ci-dessus. Par exemple, avec -j4 on peut réduire le temps de test de plus de 60 pourcents.

Cinq tests sont connus pour échouer dans l'environnement de construction de LFS à cause d’une dépendance circulaire, mais tous les tests passent s'ils sont relancés après l'installation d’automake.

Installez le paquet :

```
make install
```

Supprimez une bibliothèque statique inutile :

```
rm -fv /usr/lib/libltdl.a
```

8.35.2. Contenu de Libtool

Programmes installés: libtool et libtoolize
Bibliothèques installées: libltdl.so
Répertoires installés: /usr/include/libltdl, et /usr/share/libtool

Descriptions courtes

libtool Fournit des services de prise en charge de construction généralisée de bibliothèques

libtoolize Fournit une façon standard d'ajouter la prise en charge de **libtool** dans un paquet

`libltdl` Cache les nombreuses difficultés avec dlopen sur les bibliothèques

8.36. GDBM-1.19

Le paquet GDBM contient le gestionnaire de bases de données de GNU. C'est une bibliothèque de fonctions de bases de données qui utilise du hachage extensible et qui fonctionne comme le dbm standard d'UNIX. La bibliothèque offre les bases pour stocker des paires clés/données, chercher et extraire les données avec leur clé, effacer celles-ci ainsi que leurs données associées.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 11 Mo

8.36.1. Installation de GDBM

Préparez la compilation de GDBM :

```
./configure --prefix=/usr    \
            --disable-static \
            --enable-libgdbm-compat
```

Voici la signification de l'option de configuration :

`--enable-libgdbm-compat`
Ce paramètre permet de construire la bibliothèque de compatibilité libgdbm. D'autres paquets extérieurs à LFS peuvent exiger les anciennes routines de DBM qu'elle fournit.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Un test, `VERSION`, est connu pour échouer.

Installez le paquet :

```
make install
```

8.36.2. Contenu de GDBM

Programmes installés: gdbm_dump, gdbm_load, et gdbmtool
Bibliothèques installées: libgdbm.so et libgdbm_compat.so

Descriptions courtes

gdbm_dump	Envoie une base de données GDBM vers un fichier
gdbm_load	Recrée une base de données GDBM à partir d'un fichier
gdbmtool	Règle et modifie une base de données GDBM
`libgdbm`	Contient des fonctions pour manipuler une base de données hachée
`libgdbm_compat`	Bibliothèque de compatibilité contenant les anciennes fonctions DBM

8.37. Gperf-3.1

Gperf génère une fonction de hachage parfait à partir d'un trousseau.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 6.4 Mo

8.37.1. Installation de Gperf

Préparez la compilation de Gperf :

```
./configure --prefix=/usr --docdir=/usr/share/doc/gperf-3.1
```

Compilez le paquet :

```
make
```

Les tests sont connus pour échouer lors d'une exécution parallélisée (l'option -j plus grande que 1). Pour tester le résultat lancer :

```
make -j1 check
```

Installez le paquet :

```
make install
```

8.37.2. Contenu de Gperf

Programme installé: gperf
Répertoire installé: /usr/share/doc/gperf-3.1

Descriptions courtes

gperf Génère un hachage parfait à partir d'un trousseau

8.38. Expat-2.2.10

Le paquet Expat contient une bibliothèque C orientée flux pour analyser de l'XML.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 14 Mo

8.38.1. Installation d'Expat

Préparez la compilation d'Expat :

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.2.10
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

Si vous le désirez, installez la documentation :

```
install -v -m644 doc/*.{html,png,css} /usr/share/doc/expat-2.2.10
```

8.38.2. Contenu d'Expat

Programme installé: xmlwf
Bibliothèques installées: libexpat.so
Répertoire installé: /usr/share/doc/expat-2.2.10

Descriptions courtes

xmlwf Est un outil de validation pour vérifier si des documents XML sont bien formés

`libexpat` Contient les fonctions de l'API de l'analyse XML

8.39. Inetutils-2.0

Le paquet Inetutils contient des programmes réseaux basiques.

Temps de construction approximatif: 0.3 SBU
Espace disque requis: 31 Mo

8.39.1. Installation de Inetutils

Préparez la compilation d'Inetutils :

```
./configure --prefix=/usr        \
            --localstatedir=/var \
            --disable-logger     \
            --disable-whois      \
            --disable-rcp        \
            --disable-rexec      \
            --disable-rlogin     \
            --disable-rsh        \
            --disable-servers
```

Voici la signification des options de configure :

`--disable-logger`
Cette option empêche l'installation du programme **logger** par Inetutils. Ce programme est utilisé par les scripts pour passer des messages au démon des traces système. Nous ne l'installons pas car Util-linux livre une version plus récente.

`--disable-whois`
Cette option désactive la construction du client **whois** d'Inetutils qui est vraiment obsolète. Les instructions pour un meilleur client **whois** sont dans le livre BLFS.

`--disable-r*`
Ces paramètres désactivent la construction de programmes obsolètes qui ne doivent pas être utilisés pour des raisons de sécurité. Les fonctions fournies pas ces programmes peuvent être fournies par le paquet openssh du livre BLFS.

`--disable-servers`
Ceci désactive l'installation des différents serveurs réseau inclus dans le paquet Inetutils. Ces serveurs semblent inappropriés dans un système LFS de base. Certains ne sont pas sécurisés et ne sont considérés comme sûrs que sur des réseaux de confiance. Remarquez que de meilleurs remplacements sont disponibles pour certains de ces serveurs.

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Note

Un test, libls.sh, peut échouer dans l'environnement chroot initial mais fonctionnera si le test est relancé après que le système LFS a été installé. Un test, ping-localhost.sh, échouera si le système hôte ne gère pas IPv6.

Installez le paquet :

```
make install
```

Déplacez certains programmes pour qu'ils soient disponibles si `/usr` n'est pas accessible :

```
mv -v /usr/bin/{hostname,ping,ping6,traceroute} /bin
mv -v /usr/bin/ifconfig /sbin
```

8.39.2. Contenu de Inetutils

Programmes installés: dnsdomainname, ftp, ifconfig, hostname, ping, ping6, talk, telnet, tftp et traceroute

Descriptions courtes

dnsdomainname Affiche le nom de domaine du système

ftp Est un programme de transfert de fichier

hostname Affiche ou règle le nom de l'hôte

ifconfig Gère des interfaces réseaux

ping Envoie des paquets echo-request et affiche le temps mis pour que la réponse arrive

ping6 Une version de **ping** pour les réseaux IPv6

talk Est utilisé pour discuter avec un autre utilisateur

telnet Une interface du protocole TELNET

tftp Un programme de transfert trivial de fichiers

traceroute Trace le trajet que prennent vos paquets depuis l'endroit où vous travaillez jusqu'à un hôte sur un réseau, en montrant tous les hops (passerelles) intermédiaires pendant le chemin

8.40. Perl-5.32.1

Le paquet Perl contient le langage pratique d'extraction et de rapport (*Practical Extraction and Report Language*).

Temps de construction approximatif: 10 SBU
Espace disque requis: 222 Mo

8.40.1. Installation de Perl

Cette version de Perl compile maintenant les modules Compress::Raw::Zlib et Compress::Raw::BZip2. Par défaut Perl utilisera une copie interne du code source Zlib pour la compilation. Lancez la commande suivante afin que Perl utilise les bibliothèques Zlib installées sur le système :

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

Si vous voulez avoir un contrôle total sur la façon dont Perl est configuré, vous pouvez supprimer les options « -des » de la commande suivante et contrôler à la main la façon dont ce paquet est construit. Alternativement, utilisez exactement la commande ci-dessous pour utiliser les paramètres par défaut que détecte Perl automatiquement :

```
sh Configure -des                                         \
             -Dprefix=/usr                                \
             -Dvendorprefix=/usr                          \
             -Dprivlib=/usr/lib/perl5/5.32/core_perl      \
             -Darchlib=/usr/lib/perl5/5.32/core_perl      \
             -Dsitelib=/usr/lib/perl5/5.32/site_perl      \
             -Dsitearch=/usr/lib/perl5/5.32/site_perl     \
             -Dvendorlib=/usr/lib/perl5/5.32/vendor_perl  \
             -Dvendorarch=/usr/lib/perl5/5.32/vendor_perl \
             -Dman1dir=/usr/share/man/man1                \
             -Dman3dir=/usr/share/man/man3                \
             -Dpager="/usr/bin/less -isR"                 \
             -Duseshrplib                                 \
             -Dusethreads
```

Voici la signification de l'option de configure :

`-Dvendorprefix=/usr`
Ceci s'assure que **perl** sait comment dire aux paquets où ils devraient installer leurs modules Perl.

`-Dpager="/usr/bin/less -isR"`
Ceci assure que **`less`** est utilisé au lieu de **`more`**.

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`
Comme Groff n'est pas installé, **Configure** pense que nous ne voulons pas les pages de manuel de Perl. Ces paramètres changent cette décision.

`-Duseshrplib`
Construit une bibliothèque partagée dont certains modules perl ont besoin.

`-Dusethreads`
Construisez perl avec la prise en charge des threads.

`-Dprivlib,-Darchlib,-Dsitelib,...`
Ces paramètres définissent où Perl cherche les modules installés. Les éditeurs de LFS ont choisi de les mettre dans une structure de répertoires basée sur la version Major.Minor de Per (5.32) qui permet de mettre à jour Perl vers ne nouvelles version Patch (5.32.1) sans avoir besoin de réinstaller tous les modules.

Compilez le paquet :

```
make
```

Pour tester les résultats (approximativement 11 SBU), lancez :

```
make test
```

Installez le paquet et faites le ménage :

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

8.40.2. Contenu de Perl

Programmes installés: corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.32.1 (lien en dur vers perl), perlbug, perldoc, perlivp, perlthanks (lien en dur vers perlbug), piconv, pl2pm, pod2html, pod2man, pod2text, pod2usage, podchecker, podselect, prove, ptar, ptardiff, ptargrep, shasum, splain, xsubpp et zipdetails
Bibliothèques installées: Plusieurs qui ne peuvent pas être listés ici
Répertoire installé: /usr/lib/perl5

Descriptions courtes

corelist Une interface en ligne de commande pour Module::CoreList

cpan Interagit avec le réseau d'archive Perl global (*Comprehensive Perl Archive Network*, CPAN) à partir de la ligne de commande

enc2xs Construit une extension Perl pour le module Encode, soit à partir de *Unicode Character Mappings* soit à partir de *Tcl Encoding Files*

encguess Devine le type d'encodage d'un ou plusieurs fichiers

h2ph Convertit les fichiers d'en-têtes C `.h` en fichiers d'en-têtes Perl `.ph`

h2xs Convertit les fichiers d'en-têtes C `.h` en extensions Perl

instmodsh Script shell pour examiner les modules Perl installés, et pouvant créer une archive tar à partir d'un module installé

json_pp Convertit des données entre certains formats d'entrée et de sortie

libnetcfg Peut être utilisé pour configurer le module Perl `libnet`

perl Combine quelques-unes des meilleures fonctionnalités de C, **sed**, **awk** et **sh** en un langage style couteau suisse

perl5.32.1 Un lien vers **perl**

perlbug Utilisé pour générer des rapports de bogues sur Perl ou les modules l'accompagnant et pour les envoyer par courrier électronique

perldoc Affiche une partie de la documentation au format pod, embarquée dans le répertoire d'installation de Perl ou dans un script Perl

perlivp La procédure de vérification d'installation de Perl. Il peut être utilisé pour vérifier que Perl et ses bibliothèques ont été installés correctement

perlthanks Utilisé pour générer des messages de remerciements par mail aux développeurs de Perl

piconv Une version Perl du convertisseur de codage des caractères **iconv**

pl2pm Un outil simple pour la conversion des fichiers Perl4 `.pl` en modules Perl5 `.pm`

pod2html	Convertit des fichiers à partir du format pod vers le format HTML
pod2man	Convertit des fichiers à partir du format pod vers une entrée formatée *roff
pod2text	Convertit des fichiers à partir du format pod vers du texte ANSI
pod2usage	Affiche les messages d'usage à partir des documents embarqués pod
podchecker	Vérifie la syntaxe du format pod des fichiers de documentation
podselect	Affiche les sections sélectionnées de la documentation pod
prove	Outil en ligne de commande pour lancer des tests liés au module Test::Harness
ptar	Un programme du genre **tar** écrit en Perl
ptardiff	Un programme Perl qui compare une archive extraite et une non extraite
ptargrep	Un programme Perl qui applique des modèles correspondant au contenu des fichiers d'une archive tar
shasum	Affiche ou vérifie des sommes de contrôle SHA
splain	Utilisé pour forcer la verbosité des messages d'avertissement avec Perl
xsubpp	Convertit le code Perl XS en code C
zipdetails	Affiche des détails sur la structure interne d'un fichier Zip

8.41. XML::Parser-2.46

Le module XML::Parser est une interface Perl avec l'analyseur Expat de James Clark.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 2.4 Mo

8.41.1. Installation de XML::Parser

Préparez la compilation de XML::Parser :

```
perl Makefile.PL
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make test
```

Installez le paquet :

```
make install
```

8.41.2. Contenu de XML::Parser

Module installé: Expat.so

Descriptions courtes

`Expat` fournit l'interface Perl avec Expat

8.42. Intltool-0.51.0

Le paquet Intltool est un outil d'internationalisation utilisé pour extraire des chaînes traduisibles à partir de fichiers sources.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 1.5 Mo

8.42.1. Installation d'Intltool

Corrigez un avertissement causé par perl-5.22 et les versions ultérieures :

```
sed -i 's:\\\${:\\\$\\{:' intltool-update.in
```

Note

L'expression régulière ci-dessus a l'air inhabituelle à cause des antislashs. Elle ajoute un antislash avant l'accolade ouvrante dans la séquence « \${ » ce qui donne « \&\{ ».

Préparez la compilation d'Intltool :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
install -v -Dm644 doc/I18N-HOWTO /usr/share/doc/intltool-0.51.0/I18N-HOWTO
```

8.42.2. Contenu d'Intltool

Programmes installés: intltool-extract, intltool-merge, intltool-prepare, intltool-update et intltoolize
Répertoires installés: /usr/share/doc/intltool-0.51.0 et /usr/share/intltool

Descriptions courtes

intltoolize	Prépare l'utilisation d'intltool par un paquet
intltool-extract	Génère des fichiers d'en-tête lisibles par **gettext**.
intltool-merge	Rassemble les chaînes traduites dans divers types de fichiers.
intltool-prepare	Met à jour les fichiers pot et les synchronise avec les fichiers de traduction.
intltool-update	Met à jour les modèles po et les synchronise avec les traductions

8.43. Autoconf-2.71

Le paquet Autoconf contient des programmes produisant des scripts shell qui configurent automatiquement le code source.

Temps de construction approximatif: moins de 0.1 SBU (environ 7.2 SBU avec les tests)
Espace disque requis: 24 Mo

8.43.1. Installation de Autoconf

Préparez la compilation d'Autoconf :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.43.2. Contenu de Autoconf

Programmes installés: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate et ifnames
Répertoire installé: /usr/share/autoconf

Descriptions courtes

autoconf Produit des scripts shell configurant automatiquement des paquets de code source, permettant ainsi de les adapter à tous les types de systèmes Unix. Les scripts de configuration qu'il produit sont indépendants. Les exécuter ne nécessite pas le programme **autoconf**

autoheader Un outil pour créer des fichiers modèle d'instructions C *#define* que configure utilise

autom4te Un emballage pour le processeur de macro M4

autoreconf Exécute automatiquement **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize**, et **libtoolize** dans le bon ordre pour gagner du temps lorsque des modifications ont eu lieu sur les fichiers modèles d'**autoconf** et d'**automake**

autoscan Aide à la création de fichiers `configure.in` pour un paquet logiciel. Il examine les fichiers source d'un répertoire et crée un fichier `configure.scan` servant de fichier `configure.in` préliminaire pour le paquet

autoupdate Modifie un fichier `configure.in` qui appelle toujours les macros **autoconf** par leurs anciens noms pour qu'il utilise les noms de macros actuels

ifnames Sert à écrire les fichiers `configure.in` pour un paquet logiciel. Il affiche les identifieurs que le paquet utilise dans des conditions du préprocesseur C. Si un paquet a déjà été initialisé pour avoir une certaine portabilité, ce programme aide à déterminer ce que **configure** doit vérifier. Il peut aussi remplir les blancs dans un fichier `configure.in` généré par **autoscan**

8.44. Automake-1.16.3

Le paquet Automake contient des programmes de génération de Makefile à utiliser avec Autoconf.

Temps de construction approximatif: moins de 0.1 SBU (environ 9.1 SBU avec les tests)
Espace disque requis: 115 Mo

8.44.1. Installation de Automake

Corrigez un test qui échoue :

```
sed -i "s/''/etags/" t/tags-lisp-space.sh
```

Préparez la compilation d'Automake :

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.16.3
```

Compilez le paquet :

```
make
```

L'option -j4 de make accélère la vitesse des tests, même sur les processeurs à un seul cœur en raison de délais internes de chaque test. Pour tester les résultats, lancez :

```
make -j4 check
```

Les tesst t/subobj.sh, t/deprecated-acinit.sh et t/init.sh sont connus pour échouer dans l'environnement LFS.

Installez le paquet :

```
make install
```

8.44.2. Contenu de Automake

Programmes installés: aclocal, aclocal-1.16 (lié en dur avec aclocal), automake, et automake-1.16 (lié en dur avec automake)
Répertoires installés: /usr/share/aclocal-1.16, /usr/share/automake-1.16, et /usr/share/doc/automake-1.16.3

Descriptions courtes

aclocal Génère des fichiers `aclocal.m4` basés sur le contenu du fichier `configure.in`

aclocal-1.16 Un lien vers **aclocal**

automake Un outil pour générer automatiquement des fichiers `Makefile.in` à partir de fichiers `Makefile.am`. Pour créer tous les fichiers `Makefile.in` d'un paquet, lancez ce programme dans le répertoire de haut niveau. En parcourant le fichier `configure.in`, il trouve automatiquement chaque fichier `Makefile.am` approprié et génère le fichier `Makefile.in` correspondant.

automake-1.16 Un lien vers **automake**

8.45. Kmod-28

Le paquet Kmod contient des bibliothèques et des outils pour charger des modules du noyau

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 13 Mo

8.45.1. Installation de Kmod

Préparez la compilation de Kmod :

```
./configure --prefix=/usr          \
            --bindir=/bin          \
            --sysconfdir=/etc      \
            --with-rootlibdir=/lib \
            --with-xz              \
            --with-zstd            \
            --with-zlib
```

Voici la signification des options de configure :

`--with-xz, --with-zlib, --with-zstd`
Ces options permettent à Kmod de gérer des modules noyau compressés.

`--with-rootlibdir=/lib`
Cette option garantit que la bibliothèque et les fichiers liés seront au bon endroit.

Compilez le paquet :

```
make
```

Ce paquet ne contient pas de suite de tests qui peut être lancée dans l'environnement chroot de LFS. Git est requis et plusieurs tests ne vont pas fonctionner en dehors d'un répertoire git.

Installez le paquet et créez des liens symboliques à des fins de compatibilité avec Module-Init-Tools (le paquet qui gérait auparavant les modules du noyau Linux) :

```
make install

for target in depmod insmod lsmod modinfo modprobe rmmod; do
  ln -sfv ../bin/kmod /sbin/$target
done

ln -sfv kmod /bin/lsmod
```

8.45.2. Contenu de Kmod

Programmes installés: depmod (lien vers kmod), insmod (lien vers kmod), kmod, kmod-nolib, lsmod (lien vers kmod), modinfo (liien vers kmod), modprobe (lien vers kmod), et rmmod (liien vers kmod)
Bibliothèque installée: libkmod.so

Descriptions courtes

depmod Crée un fichier de dépendances basé sur les symboles qu'il trouve dans l'ensemble de modules existant ; ce fichier de dépendance est utilisé par **modprobe** pour charger automatiquement les modules requis

insmod Installe un module chargeable dans le noyau en cours d'exécution

kmod Charge et décharge des modules du noyau

lsmod Cette bibliothèque est utilisée par d'autres programmes pour charger et décharger des modules noyau

lsmod Liste les modules actuellement chargés

modinfo Utilise un fichier de dépendance, créé par **depmod**, pour charger automatiquement les modules adéquats

modprobe Décharge les modules du noyau en cours d'exécution.

`libkmod` Décharge des modules du noyau en cours d'exécution

8.46. Libelf de Elfutils-0.183

Libelf est une bibliothèque pour gérer les fichiers ELF (Executable and Linkable Format).

Temps de construction approximatif: 0.9 SBU
Espace disque requis: 121 Mo

8.46.1. Installation de Libelf

Libelf fait partit du paquet elfutils-0.183. Utilisez elfutils-0.183.tar.bz2 comme archive des sources.

Préparez Libelf pour la compilation :

```
./configure --prefix=/usr                \
            --disable-debuginfod         \
            --enable-libdebuginfod=dummy \
            --libdir=/lib
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez seulement Libelf :

```
make -C libelf install
install -vm644 config/libelf.pc /usr/lib/pkgconfig
rm /lib/libelf.a
```

8.46.2. Contenu de Libelf

Bibliothèques installées: libelf.so (lien symbolique) et libelf-0.183.so
Répertoire installé: /usr/include/elfutils

Descriptions courtes

`libelf` Contient les fonction de l'API pour gérer les fichiers objet ELF

8.47. Libffi-3.3

La bibliothèque Libffi fournit une interface portable et haut-niveau pour diverses conventions d'appel. Cela permet au programmeur d'appeler des fonctions spécifiées par une interface d'appel décrite à l'exécution.

Temps de construction approximatif: 1.9 SBU
Espace disque requis: 10 Mo

8.47.1. Installation de Libffi

Note

Comme pour GMP, libffi est construit avec des optimisations spécifiques au processeur utilisé. Si vous construisez pour un autre système, exportez CFLAGS et CXXFLAGS pour spécifier une construction générique pour votre architecture. Si vous ne faites pas cela, des erreurs de type opération illégale apparaîtront à l'édition des liens.

Préparez libffi à la compilation :

```
./configure --prefix=/usr --disable-static --with-gcc-arch=native
```

Voici la signification des options de configuration :

`--with-gcc-arch=native`
S'assure que GCC active les optimisations pour le système actuel. Si l'option n'est pas spécifiée, il essaiera de deviner le système et le code généré peut ne pas être correct pour certains systèmes. Si le code généré est copié du système actuel vers un système avec moins de fonctionnalités, utilisez le système avec moins de fonctionnalités dans le paramètre. Pour des détails sur les types de systèmes alternatifs, voyez *les options x86 dans le manuel de GCC*.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.47.2. Contenu de Libffi

Bibliothèque installée: libffi.so

Descriptions courtes

`libffi` contient les fonctions de l'API de l'interface pour les fonctions externes.

8.48. OpenSSL-1.1.1j

Le paquet OpenSSL contient des outils et des bibliothèques de gestion en matière de cryptographie. Ils servent à fournir des fonctions cryptographiques à d'autres paquets, comme OpenSSH, des applications de messagerie électronique et des navigateurs Internet (pour accéder à des sites HTTPS).

Temps de construction approximatif: 2.2 SBU
Espace disque requis: 154 Mo

8.48.1. Installation d'OpenSSL

Préparez OpenSSL à la compilation :

```
./config --prefix=/usr         \
         --openssldir=/etc/ssl \
         --libdir=lib          \
         shared                \
         zlib-dynamic
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make test
```

Un test, 30-test_afalg.t, est connu pour échouer sur certaines configuration du noyau (il suppose apparemment qu'une certaine option cryptographique non spécifiée a été choisie).

Installez le paquet :

```
sed -i '/INSTALL_LIBS/s/libcrypto.a libssl.a//' Makefile
make MANSUFFIX=ssl install
```

Ajoutez la version au nom de répertoire de la documentation, pour rester cohérent avec d'autres paquets :

```
mv -v /usr/share/doc/openssl /usr/share/doc/openssl-1.1.1j
```

Si vous le souhaitez, installez de la documentation supplémentainre :

```
cp -vfr doc/* /usr/share/doc/openssl-1.1.1j
```

8.48.2. Contenu d'OpenSSL

Programmes installés: c_rehash et openssl
Bibliothèques installées: libcrypto.so et libssl.so
Répertoires installés: /etc/ssl, /usr/include/openssl, /usr/lib/engines et /usr/share/doc/openssl-1.1.1j

Descriptions courtes

c_rehash est un script Perl qui scanne tous les fichiers dans un répertoire et ajoute des liens symboliques vers leur valeur hashée

openssl est un outil en ligne de commande qui permet d'utiliser les diverses fonctions cryptographiques de la bibliothèque crypto d'OpenSSL depuis le shell. Il peut être utilisé pour diverses fonctions documentées dans **man 1 openssl**

`libcrypto.so` implémente un large éventail d'algorithmes cryptographiques utilisés dans divers standards Internet. Les services fournis par cette bibliothèque sont utilisés par les implémentations OpenSSL de SSL, TLS et S/MIME. Ils ont aussi été utilisés pour implémenter OpenSSH, OpenPGP et d'autres standards de cryptographie

`libssl.so` implémente le protocole *Transport Layer Security* (TLS v1). Elle fournit une API riche, et sa documentation peut être trouvée en lançant **man 3 ssl**

8.49. Python-3.9.2

Le paquet Python 3 contient l'environnement de développement Python. Il est utile pour programmer en orienté-objet, écrire des scripts, prototyper de plus grands programmes ou pour développer des applications complètes.

Temps de construction approximatif: 2.8 SBU
Espace disque requis: 294 Mo

8.49.1. Installation de Python 3

Préparez Python pour la compilation :

```
./configure --prefix=/usr       \
            --enable-shared     \
            --with-system-expat \
            --with-system-ffi   \
            --with-ensurepip=yes
```

Voici la signification des options de configure :

`--with-system-expat`
Ce paramètre active la liaison avec la version du système de Expat.

`--with-system-ffi`
Ce paramètre active la liaison avec la version du système de libffi.

`--with-ensurepip=yes`
Ce paramètre active la construction des gestionnaires de paquets **pip** et **setuptools**.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make test
```

Certains tests requièrent une connexion réseau ou des paquets supplémentaires et sont passés. Le test nommé test_normalization est connu pour échouer. Pour des résultats plus complets, vous pouvez relancer les tests après la réinstallation de Python 3 dans BLFS.

Installez le paquet :

```
make install
```

Si vous le souhaitez, installez la documentation préformatée :

```
install -v -dm755 /usr/share/doc/python-3.9.2/html

tar --strip-components=1  \
    --no-same-owner       \
    --no-same-permissions \
    -C /usr/share/doc/python-3.9.2/html \
    -xvf ../python-3.9.2-docs-html.tar.bz2
```

Voici la signification des commandes d'installation de la documentation :

`--no-same-owner` et `--no-same-permissions`
S'assure que les fichiers installés ont la bonne appartenance et les bonnes permissions. Sans ces options, utiliser tar installera les fichiers du paquet avec les valeurs du créateur en amont.

8.49.2. Contenu de Python 3

Programmes installés: 2to3, idle3, pip3, pydoc3, python3 et python3-config
Bibliothèques installées: libpython3.9.so et libpython3.so
Répertoires installés: /usr/include/python3.9, /usr/lib/python3 et /usr/share/doc/python-3.9.2

Descriptions courtes

2to3 est un programme Python qui lit du code source Python 2.x et applique une série de corrections pour le transformer en code Python 3.x valide

idle3 est un script enveloppe qui ouvre un éditeur en GUI qui connait Python. Pour que ce script puisse tourner, vous devez avoir installé Tk avant Python pour que le module python Tkinter soit construit

pip3 L'installateur de paquets pour Python. Vous pouvez utiliser pip pour installer des paquets de Python Package Index et d'autres répertoires

pydoc3 est l'outil de documentation de Python

python3 est un langage de programmation interprété, interactif et orienté objet

8.50. Ninja-1.10.2

Ninja est un petit système de construction qui met l'accent sur la rapidité.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 79 Mo

8.50.1. Installation de Ninja

Lorsque lancé, ninja lance un nombre maximum de processus en parallèle. Par défaut c'est le nombre de cœurs du système plus deux. Dans certains cas, cela peut surchauffer le CPU ou épuiser la mémoire. S'il est lancé depuis la ligne de commande, passer le paramètre -jN limitera le nombre de processus en parallèle, mais certains paquets incluent l'exécution de ninja et ne passent pas de paramètre -j.

Utiliser la procédure *facultative* ci-dessous permet à l'utilisateur de limiter le nombre de processus en parallèle via une variable d'environnement, NINJAJOBS. **Par exemple** initialiser :

```
export NINJAJOBS=4
```

limitera ninja à 4 processus en parallèle.

Si vous le souhaitez, ajoutez la possibilité d'utiliser la variable d'environnement NINJAJOBS en lançant :

```
sed -i '/int Guess/a \
  int   j = 0;\
  char* jobs = getenv( "NINJAJOBS" );\
  if ( jobs != NULL ) j = atoi( jobs );\
  if ( j > 0 ) return j;\
' src/ninja.cc
```

Construisez Ninja avec :

```
python3 configure.py --bootstrap
```

Voici la signification des options de construction :

`--bootstrap`
Ce paramètre force ninja à se reconstruire pour le système actuel.

Pour tester les résultats, lancez :

```
./ninja ninja_test
./ninja_test --gtest_filter=-SubprocessTest.SetWithLots
```

Installez le paquet :

```
install -vm755 ninja /usr/bin/
install -vDm644 misc/bash-completion /usr/share/bash-completion/completions/ninj
install -vDm644 misc/zsh-completion  /usr/share/zsh/site-functions/_ninja
```

8.50.2. Contenu de Ninja

Programmes installés: ninja

Descriptions courtes

ninja est le système de construction Ninja

8.51. Meson-0.57.1

Meson est un système de construction libre destiné à être très rapide et aussi facile à utiliser que possible.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 37 Mo

8.51.1. Installation de Meson

Compilez Meson avec la commande suivante :

```
python3 setup.py build
```

Ce paquet n'a pas de suite de tests.

Installez le paquet :

```
python3 setup.py install --root=dest
cp -rv dest/* /
```

Voici la signification des paramètres d'installation :

`--root=dest`
Par défaut **python3 setup.py install** installe divers fichiers (comme les pages de manuel) dans des Python Eggs. Avec un emplacement racine spécifié, **setup.py** installe ces fichiers dans la hiérarchie standard. Ensuite on peut simplement copier la hiérarchie vers l'emplacement standard.

8.51.2. Contenu de Meson

Programmes installés: meson
Répertoire installé: /usr/lib/python3.9/site-packages/meson-0.57.1-py3.9.egg-info et /usr/lib/python3.9/site-packages/mesonbuild

Descriptions courtes

meson Un système de construction pour une plus grande productivité

8.52. Coreutils-8.32

Le paquet Coreutils contient des outils pour afficher et configurer les caractéristiques basiques d'un système.

Temps de construction approximatif: 2.5 SBU
Espace disque requis: 164 Mo

8.52.1. Installation de Coreutils

POSIX exige que les programmes de Coreutils reconnaissent les limites des caractères correctement même dans des locales multibyte. Le correctif suivant corrige cette rigidité et d'autres bogues liés à l'internationalisation.

```
patch -Np1 -i ../coreutils-8.32-i18n-1.patch
```

Note

Autrefois, on a trouvé beaucoup de bogues dans ce correctif. Lorsque vous signalez aux mainteneurs de Coreutils de nouveaux bogues, merci de vérifier d'abord qu'ils sont reproductibles sans ce correctif.

Supprimez un test qui peut boucler indéfiniment sur certaines machines :

```
sed -i '/test.lock/s/^/#/' gnulib-tests/gnulib.mk
```

Maintenant, préparez la compilation de Coreutils :

```
autoreconf -fiv
FORCE_UNSAFE_CONFIGURE=1 ./configure \
            --prefix=/usr            \
            --enable-no-install-program=kill,uptime
```

Voici la signification des options de configuration

autoreconf
Le correctif pour l'internationalisation a modifié le système de construction du paquet, donc les fichiers de configuration doivent être régénérés.

`FORCE_UNSAFE_CONFIGURE=1`
Cette variable d'environnement permet au paquet d'être compilé avec l'utilisateur `root`.

`--enable-no-install-program=kill,uptime`
Le but de ce paramètre est d'empêcher Coreutils d'installer des binaires qui seront installés plus tard par d'autres paquets.

Compilez le paquet :

```
make
```

Passez à « Installez le paquet » si vous n'exécutez pas la suite de test.

Maintenant, la suite de tests peut être lancée. Tout d'abord, lancez les quelques tests qui ont besoin d'être lancés en tant que `root` :

```
make NON_ROOT_USERNAME=tester check-root
```

Nous allons exécuter le reste des tests en tant qu'utilisateur `tester`. Certains tests exigent cependant que l'utilisateur soit membre de plus d'un groupe. Afin que ces tests ne soient pas sautés, nous allons ajouter un groupe temporaire et y ajouter l'utilisateur `tester` :

```
echo "dummy:x:102:tester" >> /etc/group
```

Corrigez des droits afin qu'un utilisateur non-root puisse compiler et exécuter les tests :

```
chown -Rv tester .
```

Maintenant lancez les tests :

```
su tester -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes check"
```

Le test test-getlogin est connu pour échouer dans l'environnement chroot de LFS.

Supprimez le groupe temporaire :

```
sed -i '/dummy/d' /etc/group
```

Installez le paquet :

```
make install
```

Déplacez quelques programmes aux emplacements spécifiés par le FHS :

```
mv -v /usr/bin/{cat,chgrp,chmod,chown,cp,date,dd,df,echo} /bin
mv -v /usr/bin/{false,ln,ls,mkdir,mknod,mv,pwd,rm} /bin
mv -v /usr/bin/{rmdir,stty,sync,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
mv -v /usr/share/man/man1/chroot.1 /usr/share/man/man8/chroot.8
sed -i 's/"1"/"8"/' /usr/share/man/man8/chroot.8
```

```
mv -v /usr/bin/{head,nice,sleep,touch} /bin
```

8.52.2. Contenu de Coreutils

Programmes installés: [, b2sum, base32, base64, basename, basenc, cat, chcon, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, numfmt, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami et yes
Bibliothèque installée: libstdbuf.so (dans /usr/libexec/coreutils)
Répertoire installé: /usr/libexec/coreutils

Descriptions courtes

[Est une vrai commande, /usr/bin/[, qui est synonyme de la commande **test**.

base32 Encode et décode des données selon la spécification de la base32 (RFC 4648)

base64 Encode et décode des données selon la spécification de la base64 (RFC 4648)

b2sum Affiche ou vérifie des sommes de contrôle 512-bit BLAKE2

basename Supprime tout le chemin et un suffixe donné à partir du nom de fichier donné

basenc Encode ou décode des données avec divers algorithmes

cat Concatène des fichiers sur la sortie standard

chcon Modifie le contexte de sécurité des fichiers et des dossiers

chgrp Change le groupe propriétaire de certains fichiers et répertoires

chmod Change les droits de chaque fichier donné avec le mode indiqué. Le mode peut être soit une représentation symbolique des modifications à faire soit un nombre octal représentant les nouveaux droits

chown Modifie le propriétaire utilisateur et le groupe de certains fichiers et répertoires

chroot Lance une commande avec le répertoire spécifié `/` comme répertoire racine

cksum Affiche la somme de vérification CRC (Cyclic Redundancy Check) et le nombre d'octets de chaque fichier

comm Compare deux fichiers triés, affichant sur trois colonnes, les lignes uniques et les lignes communes

cp Copie des fichiers

csplit Divise un fichier donné sur plusieurs fichiers indiqués, les séparant par des modèles donnés ou des numéros de lignes. Il affiche le nombre total d'octets pour chaque nouveau fichier

cut Affiche des parties de lignes, sélectionnant ces parties suivant des champs ou positions donnés

date Affiche l'heure actuelle dans le format donné ou initialise la date système

dd Copie un fichier en utilisant la taille et le nombre de blocs donnés tout en réalisant des conversions optionnelles

df Affiche l'espace disque disponible (et utilisé) sur tous les systèmes de fichiers montés, ou seulement sur les systèmes de fichiers contenant les fichiers donnés

dir Liste le contenu de chaque répertoire donné (identique à la commande **ls**)

dircolors Affiche les commandes pour initialiser la variable d'environnement `LS_COLOR` ce qui permet de changer le schéma de couleurs utilisé par **ls**

dirname Supprime le suffixe qui ne représente pas le répertoire dans un nom de fichier donné

du Affiche le total de l'espace disque utilisé par le répertoire actuel, ou par chacun des répertoires donnés incluant tous les sous-répertoires, ou par chacun des fichiers donnés

echo Affiche les chaînes données

env Lance une commande dans un environnement modifié

expand Convertit les tabulations en espaces

expr Évalue des expressions

factor Affiche les facteurs premiers de tous les entiers spécifiés

false Ne fait rien. Il renvoie toujours un code d'erreur indiquant l'échec

fmt Reformate les paragraphes dans les fichiers donnés

fold Emballe les lignes des fichiers donnés

groups Affiche les groupes auxquels appartient un utilisateur

head Affiche les dix premières lignes (ou le nombre demandé de lignes) pour chaque fichier précisé

hostid Affiche l'identifieur numérique de l'hôte (en hexadécimal)

id Affiche l'identifieur effectif de l'utilisateur courant ou de l'utilisateur précisé, l'identifieur du groupe et les groupes auxquels appartient cet utilisateur

install Copie les fichiers en initialisant leurs droits et, si possible, leur propriétaire et groupe

join Joint à partir de deux fichiers les lignes qui ont des champs de jointure identiques

link Crée un lien physique avec le nom de donné vers le fichier indiqué

ln Crée des liens symboliques ou physiques entre des fichiers

logname Indique le nom de connexion de l'utilisateur actuel

ls Liste le contenu de chaque répertoire donné

md5sum Affiche ou vérifie les sommes de vérification MD5 (Message Digest 5)

mkdir Crée des répertoires avec les noms donnés

mkfifo Crée des fichiers FIFO (First-In, First-Out, un « tube nommé » dans le vocable d'Unix) avec les noms donnés

mknod Crée des nœuds de périphériques avec les noms donnés. Un nœud périphérique est de type caractère ou bloc, ou encore un FIFO

mktemp Crée des fichiers temporaires de manière sécurisée, il est utilisé dans des scripts

mv Déplace ou renomme des fichiers ou répertoires

nice Lance un programme avec une priorité modifiée

nl Numérote les lignes de fichiers donnés

nohup Lance une commande immune aux arrêts brutaux, dont la sortie est redirigée vers le journal de traces

nproc Affiche le nombre d'unités d'action disponibles pour un processus

numfmt Convertit des numéros en chaînes lisibles par un humain ou vice-versa

od Affiche les fichiers en octal ou sous d'autres formes

paste Joint les fichiers donnés en plaçant les lignes correspondantes l'une à côté de l'autre, en les séparant par des caractères de tabulation

pathchk Vérifie que les noms de fichier sont valides ou portables

pinky Un client finger léger. Il affiche quelques informations sur les utilisateurs indiqués

pr Fait de la pagination, principalement en colonne, des fichiers pour une impression

printenv Affiche l'environnement

printf Affiche les arguments donnés suivant le format demandé, un peu comme la fonction C printf

ptx Produit un index permuté à partir du contenu des fichiers indiqués, avec chaque mot dans son contexte

pwd Indique le nom du répertoire courant

readlink Indique la valeur du lien symbolique

realpath Affiche le chemin résolu

rm Supprime des fichiers ou des répertoires

rmdir Supprime des répertoires s'ils sont vides

runcon Lance une commande avec le contexte de sécurité spécifié

seq Affiche une séquence de nombres, à l'intérieur d'un intervalle et avec un incrément spécifié

sha1sum Affiche ou vérifie des sommes de contrôle 160-bit Secure Hash Algorithm (SHA1)

sha224sum Affiche ou vérifie des sommes de contrôle 224-bit Secure Hash Algorithm (SHA1)

sha256sum Affiche ou vérifie des sommes de contrôle 256-bit Secure Hash Algorithm (SHA1)

sha384sum Affiche ou vérifie des sommes de contrôle 384-bit Secure Hash Algorithm (SHA1)

sha512sum Affiche ou vérifie des sommes de contrôle 512-bit Secure Hash Algorithm (SHA1)

shred Efface les fichiers indiqués en écrivant dessus des modèles aléatoires pour rendre la récupération des données très difficile

shuf Mélange des lignes de texte

sleep Fait une pause d'un certain temps

sort Trie les lignes des fichiers donnés

split Divise les fichiers donnés en plusieurs pièces, par taille ou par nombre de lignes

stat Affiche le statut du fichier ou du système de fichiers

stdbuf Lance des commandes avec des opérations de mise en tampon différentes pour ses flux standards

stty Initialise ou affiche les paramètres de la ligne de terminal

sum Affiche la somme de contrôle et le nombre de blocs pour chacun des fichiers donnés

sync Vide les tampons du système de fichiers. Cela force l'enregistrement sur disque des blocs modifiés et met à jour le superbloc

tac Concatène les fichiers donnés à l'envers

tail Affiche les dix dernières lignes (ou le nombre de lignes indiqué) pour chaque fichier précisé

tee Lit à partir de l'entrée standard en écrivant à la fois sur la sortie standard et sur les fichiers indiqués

test Compare des valeurs et vérifie les types de fichiers

timeout Lance une commande avec une limite de temps

touch Modifie l'horodatage d'un fichier, initialise les dates/heures d'accès et de modification des fichiers indiqués à l'heure actuelle. Les fichiers inexistants sont créés avec une longueur nulle

tr Convertit, compresse et supprime les caractères lus depuis l'entrée standard

true Ne fait rien mais avec succès. Il quitte avec un code de sortie indiquant une réussite

truncate Réduit ou augmente un fichier selon la taille spécifiée

tsort Réalise un tri topologique. Il écrit une liste totalement ordonnée suivant un fichier donné partiellement ordonné

tty Indique le nom du fichier du terminal connecté à l'entrée standard

uname Affiche des informations système

unexpand Convertit les espaces en tabulations

uniq Ne conserve qu'une seule ligne parmi plusieurs lignes successives identiques

unlink Supprime le fichier donné

users Indique les noms des utilisateurs actuellement connectés

vdir Est identique à **ls -l**

wc Indique le nombre de lignes, mots et octets de chaque fichier indiqué ainsi que le total de lignes lorsque plus d'un fichier est donné

who Indique qui est connecté

whoami Indique le nom de l'utilisateur associé avec l'identifieur utilisateur effectif

yes Affiche indéfiniment « y » ou la chaîne précisée jusqu'à ce que le processus soit tué

`libstdbuf` Bibliothèque utilisée par **stdbuf**

8.53. Check-0.15.2

Check est un environnement de tests unitaires pour C.

Temps de construction approximatif:	0.1 SBU (environ 4.2 SBU avec les tests)
Espace disque requis:	12 Mo

8.53.1. Installation de Check

PrÃ©parez Check pour la compilation :

```
./configure --prefix=/usr --disable-static
```

Construisez le paquet :

```
make
```

La compilation est maintenant terminÃ©e. Pour lancer la suite de tests de Check, lancez la commande suivante :

```
make check
```

Remarquez que la suite de tests de Check peut prendre un temps relativement long (jusqu'Ã 4 SBU).

Installez le paquet :

```
make docdir=/usr/share/doc/check-0.15.2 install
```

8.53.2. Contenu de Check

Programme installÃ©:	checkmk
BibliothÃ¨que installÃ©e:	libcheck.so

Descriptions courtes

checkmk	Script awk pour gÃ©nÃ©rer des tests d'unitÃ© C Ã utiliser avec l'environnement de tests d'unitÃ© de Check
`libcheck.{a,so}`	Contient les fonctions permettant Ã Check d'Ãªtre appelÃ© depuis un programme de test

8.54. Diffutils-3.7

Le paquet Diffutils contient les programmes montrant les différences entre fichiers ou répertoires.

Temps de construction approximatif: 0.4 SBU
Espace disque requis: 33 Mo

8.54.1. Installation de Diffutils

Préparez la compilation de Diffutils :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez ce paquet :

```
make install
```

8.54.2. Contenu de Diffutils

Programmes installés: cmp, diff, diff3 et sdiff

Descriptions courtes

cmp Compare deux fichiers et rapporte si ou à quels endroits ils diffèrent
diff Compare deux fichiers ou répertoires et rapporte les lignes où les fichiers diffèrent
diff3 Compare trois fichiers ligne par ligne
sdiff Assemble deux fichiers et affiche le résultat de façon interactive

8.55. Gawk-5.1.0

Le paquet Gawk contient des programmes de manipulation de fichiers texte.

Temps de construction approximatif:	0.4 SBU
Espace disque requis:	45 Mo

8.55.1. Installation de Gawk

Tout d'abord, assurez-vous que certains fichiers inutiles ne sont pas installés :

```
sed -i 's/extras//' Makefile.in
```

Préparez la compilation de Gawk :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

Si désiré, installez la documentation :

```
mkdir -v /usr/share/doc/gawk-5.1.0
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-5.1.0
```

8.55.2. Contenu de Gawk

Programmes installés:	awk (lien vers gawk), gawk et awk-5.1.0
Bibliothèques installées:	filefuncs.so, fnmatch.so, fork.so, inplace.so, intdiv.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoway.so, rwarray.so, et time.so (toutes dans /usr/lib/gawk)
Répertoires installés:	/usr/lib/gawk, /usr/libexec/awk, /usr/share/awk, et /usr/share/doc/gawk-5.1.0

Descriptions courtes

awk	Un lien vers **gawk**
gawk	Un programme de manipulation de fichiers texte. C'est l'implémentation GNU d'**awk**
gawk-5.1.0	Un lien vers **gawk**

8.56. Findutils-4.8.0

Le paquet Findutils contient des programmes de recherche de fichiers. Ces programmes sont fournis pour rechercher récursivement dans une hiérarchie de répertoires et pour créer, maintenir et chercher dans une base de données (souvent plus rapide que la recherche récursive mais moins fiable si la base de données n'a pas été mise à jour récemment).

Temps de construction approximatif: 0.9 SBU
Espace disque requis: 55 Mo

8.56.1. Installation de Findutils

Préparez la compilation de Findutils :

```
./configure --prefix=/usr --localstatedir=/var/lib/locate
```

Voici la signification de l'option de configure :

`--localstatedir`
Cette option modifie l'emplacement de la base de données **locate** pour qu'elle soit dans `/var/lib/locate`, pour être compatible avec FHS.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
chown -Rv tester .
su tester -c "PATH=$PATH make check"
```

Installez le paquet :

```
make install
```

Certains paquets dans BLFS et au-delà attendent le programme **find** dans `/bin`, assurez-vous donc qu'il y est placé :

```
mv -v /usr/bin/find /bin
sed -i 's|find:=${BINDIR}|find:=/bin|' /usr/bin/updatedb
```

8.56.2. Contenu de Findutils

Programmes installés: find, locate, updatedb et xargs
Répertoire installé: /var/lib/locate

Descriptions courtes

find Cherche dans les hiérarchies de répertoires donnés les fichiers correspondant à un critère spécifié

locate Recherche à travers la base de données des noms de fichiers et renvoie les noms contenant une certaine chaîne ou correspondant à un certain modèle

updatedb Met à jour la base de données **locate** ; Il parcourt le système de fichiers entier (en incluant les autres systèmes de fichiers actuellement montés, sauf si le contraire est spécifié) et place tous les noms de fichiers qu'ils trouvent dans la base de données

xargs Peut être utilisé pour lancer une commande donnée sur une liste de fichiers

8.57. Groff-1.22.4

Le paquet Groff contient des programmes de formatage de texte.

Temps de construction approximatif: 0.5 SBU
Espace disque requis: 96 Mo

8.57.1. Installation de Groff

Groff s'attend à ce que la variable d'environnement PAGE contienne la taille du papier par défaut. Pour les utilisateurs américains, *PAGE=letter* est adéquate. *PAGE=A4* pourrait aller mieux ailleurs. Si la taille du papier par défaut est configurée lors de la compilation, elle peut être réécrite plus tard en écrivant « A4 » ou « letter » dans le fichier `/etc/papersize`.

Maintenant, préparez la compilation de Groff :

```
PAGE=<taille_papier> ./configure --prefix=/usr
```

Ce paquet ne prend pas en charge la compilation en parallèle. Compilez le paquet :

```
make -j1
```

Ce paquet n'est pas fourni avec une suite de test.

Installez le paquet :

```
make install
```

8.57.2. Contenu de Groff

Programmes installés: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, glilypond, gperl, gpinyin, grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, gropdf, grops, grotty, hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfmom, pdfroff, pfbtops, pic, pic2graph, post-grohtml, preconv, pre-grohtml, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit, et troff
Répertoires installés: /usr/lib/groff and /usr/share/doc/groff-1.22.4, /usr/share/groff

Descriptions courtes

addftinfo Lit un fichier de polices troff et ajoute quelques informations métriques supplémentaires sur la police qui est utilisée par le système **groff**

afmtodit Crée un fichier de police à utiliser avec **groff** et **grops**

chem Préprocesseur Groff pour produire des diagrammes de structure chimique

eqn Compile les descriptions d'équations imbriquées dans les fichiers d'entrée de troff pour obtenir des commandes comprises par **troff**

eqn2graph Convertit une équation EQN troff en une image améliorée

gdiffmk Marque les différences entre des fichiers groff/nroff/troff

glilypond Transforme les partitions de musiques du langage lilypond en langage groff

gperl Préprocesser groff, permettant l'ajout de code perl dans les fichiers groff

gpinyin Préprocesseur groff, permettant d'ajouter du Pinyin (chinois en alphabet européen) dans les fichiers groff.

grap2graph Convertit diagramme grap en image bitmap exploitable

grn Un préprocesseur **groff** pour les fichiers gremlin

grodvi Un pilote pour **groff** qui produit un format dvi TeX

groff Une interface au système de formatage de document groff. Normalement, il lance le programme **troff** et un post-processeur approprié au périphérique sélectionné

groffer Affiche des fichiers groff et des pages man sur des terminaux X et tty

grog Lit des fichiers et devine les options `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s`, et `-t` de **groff** requises pour l'impression des fichiers. Il indique la commande **groff** incluant ces options

grolbp Pilote **groff** pour les imprimantes Canon CAPSL (imprimantes laser de la série LBP-4 et LBP-8

grolj4 Un pilote pour **groff** produisant une sortie au format PCL5, intéressant les imprimantes HP Laserjet 4

gropdf Traduit la sortie de GNU **troff** en PDF

grops Traduit la sortie de GNU **troff** en PostScript

grotty Traduit la sortie de GNU **troff** en un format compatible pour les périphériques de type machine à écrire

hpftodit Crée un fichier de polices à utiliser avec **groff -Tlj4** à partir d'un fichier métrique de police HP

indxbib Crée un index inversé d'un fichier spécifié, index utilisé par les bases de données bibliographiques avec **refer**, **lookbib** et **lkbib**

lkbib Recherche dans les bases de données bibliographiques des références contenant certaines clés et indique toute référence trouvée

lookbib Affiche une invite sur la sortie des erreurs (sauf si l'entrée standard n'est pas un terminal), lit à partir de l'entrée standard une ligne contenant un ensemble de mots clés, recherche dans les bases de données bibliographiques dans un fichier spécifié les références contenant ces mots clés, affiche toute référence trouvée sur la sortie standard et répère ce processus jusqu'à la fin de l'entrée

mmroff Un pré-processeur pour **groff**

neqn Formate les équations pour une sortie ASCII (*American Standard Code for Information Interchange*)

nroff Un script qui émule la commande **nroff** en utilisant **groff**

pdfmom Enveloppe autour de groff qui facilite la production de documents PDF depuis des fichiers formattés avec les macros parentes.

pdfroff Crée des documents pdf en utilisant groff

pfbtops Traduit une police Postscript au format `.pfb`

pic Compile les descriptions d'images embarquées à l'intérieur de fichiers d'entrées troff ou TeX en des commandes comprises par TeX ou **troff**

pic2graph Convertit un diagramme PIC en une image améliorée

post-grohtml Traduit la sortie de GNU **troff** en HTML

preconv Convertit l'encodage de fichiers en entrée vers quelque chose que comprend GNU **troff**

pre-grohtml Traduit la sortie de GNU **troff** en HTML

refer Copie le contenu d'un fichier sur la sortie standard, sauf pour les lignes entre les symboles *.[* et *.]* interprétées comme des citations, et les lignes entre *.R1* et *.R2* interprétées comme des commandes sur la façon de gérer les citations

roff2dvi Transforme des fichiers roff au format DVI

roff2html	Transforme des fichiers roff au format HTML
roff2pdf	Transforme des fichiers roff au format PDF
roff2ps	Transforme des fichiers roff au format ps
roff2text	Transforme des fichiers roff en fichiers textes
roff2x	Transforme des fichiers roff dans d'autres formats
soelim	Lit des fichiers et remplace les lignes de la forme *.so fichier* par le contenu du *fichier* mentionné
tbl	Compile les descriptions des tables imbriquées dans les fichiers d'entrées troff en commandes comprises par **troff**
tfmtodit	Crée un fichier de police à utiliser avec **groff -Tdvi**
troff	Est hautement compatible avec la commande Unix **troff**. Habituellement, il devrait être appelé en utilisant la commande **groff** qui lance aussi les pré-processeurs et post-processeurs dans l'ordre approprié et avec les options appropriées

8.58. GRUB-2.04

Le paquet Grub contient un chargeur de démarrage, le *GRand Unified Bootloader*.

Temps de construction approximatif: 0.7 SBU
Espace disque requis: 154 Mo

8.58.1. Installation de GRUB

Tout d'abord, corrigez un problème introduit par binutils-2.36 :

```
sed "s/gold-version/& -R .note.gnu.property/" \
    -i Makefile.in grub-core/Makefile.in
```

Préparez la compilation de GRUB :

```
./configure --prefix=/usr          \
            --sbindir=/sbin        \
            --sysconfdir=/etc      \
            --disable-efiemu       \
            --disable-werror
```

La signification des nouvelles options de configurations :

`--disable-werror`
Cela permet de terminer la compilation avec les avertissements introduits dans des versions récentes de Flex.

`--disable-efiemu`
Cette option minimise ce qui est construit en désactivant des fonctionnalités et des programmes de tests non nécessaires pour LFS.

Compilez le paquet :

```
make
```

Ce paquet n'est pas fourni avec une suite de tests.

Installez le paquet :

```
make install
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

L'utilisation de GRUB pour rendre un système LFS amorçable sera traitée au Section 10.4, « Utiliser GRUB pour paramétrer le processus de démarrage ».

8.58.2. Contenu de GRUB

Programmes installés: grub-bios-setup, grub-editenv, grub-file, grub-fstest, grub-glue-efi, grub-install, grub-kbdcomp, grub-macbless, grub-menulst2cfg, grub-mkconfig, grub-mkimage, grub-mklayout, grub-mknetdir, grub-mkpasswd-pbkdf2, grub-mkrelpath, grub-mkrescue, grub-mkstandalone, grub-ofpathname, grub-probe, grub-reboot, grub-render-label, grub-script-check, grub-set-default, grub-sparc64-setup et grub-syslinux2cfg
Répertoires installés: /usr/lib/grub, /etc/grub.d, /usr/share/grub et boot/grub (après avoir lancé grub-install pour la première fois)

Descriptions courtes

grub-bios-setup Est un programme d'aide pour grub-install

grub-editenv	Un outil pour éditer l'ensemble d'environnement
grub-file	Vérifie que FILE est du type spécifié.
grub-fstest	Outil de débogage du pilote d'un système de fichiers
grub-glue-efi	Traite les images EFI ia32 et amd64 et les colle ensemble suivant le format d'Apple.
grub-install	Installe GRUB sur votre lecteur
grub-kbdcomp	Script qui convertit un plan xkb dans un plan reconnu par GRUB
grub-macbless	bless dans le style de mac pour les fichiers HFS ou HFS+
grub-menulst2cfg	Convertit un `menu.lst` du GRUB de base en fichier `grub.cfg` utilisable avec GRUB 2
grub-mkconfig	Génère un fichier de configuration grub
grub-mkimage	Crée une image amorçable de GRUB
grub-mklayout	Génère un fichier de plan de clavier pour GRUB
grub-mknetdir	Prépare un répertoire GRUB d'amorçage par le réseau
grub-mkpasswd-pbkdf2	Génère un mot de passe PBKDF2 chiffré pour une utilisation dans le menu de démarrage
grub-mkrelpath	Rend relatif le nom de chemin vers la racine d'un système
grub-mkrescue	Fabrique une image amorçable de GRUB adaptée à une disquette ou à CDROM/DVD
grub-mkstandalone	Génère une image autonome
grub-ofpathname	Est un programme d'aide qui affiche le chemin d'un périphérique GRUB
grub-probe	Teste les informations de périphérique pour un chemin ou un périphérique donné
grub-reboot	Règle l'entrée d'amorçage par défaut pour GRUB uniquement pour le prochain démarrage
grub-render-label	Produit des .disk_label Apple pour les Macs Apple
grub-script-check	Vérifie les erreurs de syntaxe du script de configuration de GRUB
grub-set-default	Règle l'entrée d'amorçage par défaut pour GRUB
grub-sparc64-setup	Est un programme d'aide pour grub-setup
grub-syslinux2cfg	Transforme un fichier de configuration syslinux vers le format de grub.cfg

8.59. Less-563

Le paquet Less contient un visualiseur de fichiers texte.

Temps de construction approximatif: moins de 0.1 SBU
Espace disque requis: 4.1 Mo

8.59.1. Installation de Less

Préparez la compilation de Less :

```
./configure --prefix=/usr --sysconfdir=/etc
```

Voici la signification de l'option de configure :

`--sysconfdir=/etc`
Cette option indique aux programmes créés par le paquet de chercher leurs fichiers de configuration dans `/etc`.

Compilez le paquet :

```
make
```

Ce paquet n'est pas fourni avec une suite de test.

Installez le paquet :

```
make install
```

8.59.2. Contenu de Less

Programmes installés: less, lessecho et lesskey

Descriptions courtes

less Un visualiseur de fichiers. Il affiche le contenu du fichier donné, vous permettant d'aller vers le haut et vers le bas, de chercher des chaînes et de sauter vers des repères

lessecho Nécessaire pour étendre les méta-caractères, comme * et *?*, dans les noms de fichiers de systèmes Unix

lesskey Utilisé pour spécifier les associations de touches pour **less**

8.60. Gzip-1.10

Le paquet Gzip contient des programmes de compression et décompression de fichiers.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 19 Mo

8.60.1. Installation de Gzip

Préparez la compilation de Gzip :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

Déplacez un programme qui a besoin d'être sur le système de fichiers racine :

```
mv -v /usr/bin/gzip /bin
```

8.60.2. Contenu de Gzip

Programmes installés: gunzip, gzexe, gzip, uncompress (lien dur avec gunzip), zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore et znew

Descriptions courtes

gunzip Décompresse les fichiers gzip

gzexe Crée des fichiers exécutables auto-extractibles

gzip Compresse les fichiers donnés en utilisant le codage Lempel-Ziv (LZ77)

uncompress Décompresse les fichiers compressés

zcat Décompresse les fichiers gzip sur la sortie standard

zcmp Lance **cmp** sur des fichiers compressés avec gzip

zdiff Lance **diff** sur des fichiers compressés avec gzip

zegrep Lance **egrep** sur des fichiers compressés avec gzip

zfgrep Lance **fgrep** sur des fichiers compressés avec gzip

zforce Force une extension `.gz` sur tous les fichiers donnés qui sont au format gzip, pour que **gzip** ne les compresse pas de nouveau ; ceci est utile quand les noms de fichiers sont tronqués lors d'un transfert de fichiers

zgrep Lance **grep** sur des fichiers compressés avec gzip

zless Lance **less** sur des fichiers compressés avec gzip

zmore Lance **more** sur des fichiers compressés avec gzip

znew Recompresse les fichiers formatés avec **compress** au format **gzip**— de `.Z` vers `.gz`

8.61. IPRoute2-5.10.0

Le paquet IPRoute2 contient des programmes pour le réseau, basique ou avancé, basé sur IPV4.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 15 Mo

8.61.1. Installation de IPRoute2

Le programme **arpd** inclus dans ce paquet ne sera pas construit car il dépend de Berkeley DB, qui n'est pas installé dans LFS. Cependant, un dossier pour **arpd** et une page de manuel seront tout de même installés. Empêchez-le en lançant la commande ci-dessous. Si vous avez besoin du binaire **arpd**, vous pouvez trouver des instructions pour la compilation de Berkeley DB dans le livre BLFS sur *http://fr.linuxfromscratch.org/blfs/../view/blfs-10.1/server/db.html*.

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

Il est aussi nécessaire de désactiver la construction de deux modules qui nécessitent *http://fr.linuxfromscratch.org/blfs/../view/blfs-10.1/postlfs/iptables.html*.

```
sed -i 's/.m_ipt.o//' tc/Makefile
```

Compilez le paquet :

```
make
```

Ce paquet n'a pas de suite de tests fonctionnelle.

Installez le paquet :

```
make DOCDIR=/usr/share/doc/iproute2-5.10.0 install
```

8.61.2. Contenu de IPRoute2

Programmes installés: bridge, ctstat (link to lnstat), genl, ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (lien vers lnstat), ss et tc
Répertoires installés: /etc/iproute2, /usr/lib/tc et /usr/share/doc/iproute2-5.10.0

Descriptions courtes

bridge Configure des ponts réseaux

ctstat Outil donnant le statut de la connexion

genl Interface netlink générique

ifcfg Un emballage en script shell pour la commande **ip**. Remarquez qu'il a besoin des programmes **arping** et **rdisk** du paquet iputils que vous pouvez trouver sur *http://www.skbuff.net/iputils/*.

ifstat Affiche les statistiques des interfaces, incluant le nombre de paquets émis et transmis par l'interface

ip L'exécutable principal. Il a plusieurs fonctions :
ip link *<périphérique>* autorise les utilisateurs à regarder l'état des périphériques et à faire des changements
ip addr autorise les utilisateurs à regarder les adresses et leurs propriétés, à ajouter de nouvelles adresses et à supprimer les anciennes

ip neighbor autorise les utilisateurs à regarder dans les liens des voisins et dans leurs propriétés, à ajouter de nouvelles entrées et à supprimer les anciennes
ip rule autorise les utilisateurs à regarder les politiques de routage et à les modifier
ip route autorise les utilisateurs à regarder la table de routage et à modifier les règles de routage
ip tunnel autorise les utilisateurs à regarder les tunnels IP et leurs propriétés, et à les modifier
ip maddr autorise les utilisateurs à regarder les adresses multicast et leurs propriétés, et à les changer
ip mroute autorise les utilisateurs à configurer, modifier ou supprimer le routage multicast
ip monitor autorise les utilisateurs à surveiller en continu l'état des périphériques, des adresses et des routes

lnstat Fournit les statistiques réseau Linux. C'est un remplacement plus généraliste et plus complet de l'ancien programme **rtstat**

nstat Affiche les statistiques réseau

routef Un composant de **ip route** pour vider les tables de routage

routel Un composant de **ip route** pour afficher les tables de routage

rtacct Affiche le contenu de `/proc/net/rt_acct`

rtmon Outil de surveillance de routes

rtpr Convertit la sortie de **ip -o** en un format lisible

rtstat Outil de statut de routes

ss Similaire à la commande **netstat** ; affiche les connexions actives

tc Exécutable de contrôle du trafic ; utile pour l'implémentation de la qualité de service (QOS) et de la classe de service (COS)
tc qdisc autorise les utilisateurs à configurer la discipline de queues
tc class autorise les utilisateurs à configurer les classes suivant la planification de la discipline de queues
tc estimator autorise les utilisateurs à estimer le flux réseau dans un réseau
tc filter autorise les utilisateurs à configurer les filtres de paquets pour QOS/COS
tc policy autorise les utilisateurs à configurer les politiques QOS/COS

8.62. Kbd-2.4.0

Le paquet Kbd contient les fichiers de tables de caractères, les polices de la console et des outils pour le clavier.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 33 Mo

8.62.1. Installation de Kbd

Le comportement des touches Effacement et Supprimer n'est pas cohérent dans toutes les tables de correspondance du clavier du paquet Kbd. Le correctif suivant répare ce problème pour les tables de correspondance i386 du clavier :

```
patch -Np1 -i ../kbd-2.4.0-backspace-1.patch
```

Après la correction, la touche Effacement génère le caractère de code 127, et la touche Supprimer génère une séquence d'échappement bien connue.

Supprimez le programme **resizecons** redondant (il exige feu svgalib pour fournir les fichiers du mode graphique - pour une utilisation normale, **setfont** redimensionne correctement la console) ainsi que sa page de man.

```
sed -i '/RESIZECONS_PROGS=/s/yes/no/' configure
sed -i 's/resizecons.8 //' docs/man/man8/Makefile.in
```

Préparez la compilation de Kbd :

```
./configure --prefix=/usr --disable-vlock
```

Voici la signification des options de configuration :

`--disable-vlock`
Cette option empêche la construction de l'utilitaire vlock car il requiert la bibliothèque PAM qui n'est pas disponible dans l'environnement chroot.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

Note

Pour certaines langues (comme le biélorusse), le paquet Kbd ne fournit pas une table de correspondance utile, puisque le contenu de la table « by » suppose l'encodage ISO-8859-5, et la table CP1251 est normalement utilisée. Les utilisateurs de telles langues doivent télécharger les tables de correspondance qui conviennent séparément.

Si désiré, installez la documentation :

```
mkdir -v            /usr/share/doc/kbd-2.4.0
cp -R -v docs/doc/* /usr/share/doc/kbd-2.4.0
```

8.62.2. Contenu de Kbd

Programmes installés: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (lien vers psfxtable), psfgettable (lien vers psfxtable), psfstriptable (lien vers psfxtable), psfxtable, setfont, setkeycodes, setleds, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start et unicode_stop

Répertoires installés: /usr/share/consolefonts, /usr/share/consoletrans, /usr/share/keymaps, /usr/share/doc/kbd-2.4.0 et /usr/share/unimaps

Descriptions courtes

chvt	Change le terminal virtuel en avant plan
deallocvt	Désalloue les terminaux virtuels inutilisés
dumpkeys	Affiche la table de traduction du clavier
fgconsole	Affiche le numéro du terminal virtuel actif
getkeycodes	Affiche la table de correspondance des « scancode » avec les « keycode »
kbdinfo	Obtient des informations sur l'état d'une console
kbd_mode	Affiche ou initialise le mode du clavier
kbdrate	Initialise les taux de répétition et de délai du clavier
loadkeys	Charge les tables de traduction du clavier
loadunimap	Charge la table de correspondance du noyau unicode-police
mapscrn	Un programme obsolète utilisé pour charger une table de correspondance des caractères de sortie définie par l'utilisateur dans le pilote de la console. Ceci est maintenant fait par **setfont**
openvt	Lance un programme sur un nouveau terminal virtuel (VT)
psfaddtable	Ajoute une table de caractères a Unicode à la police d'une console
psfgettable	Extrait la table de caractères Unicode embarquée dans la police de la console
psfstriptable	Supprime la table de caractères Unicode embarquée dans la police de la console
psfxtable	Gère les tables de caractères Unicode pour les polices de la console
setfont	Modifie les polices EGA/VGA (*Enhanced Graphic Adapter-Video Graphics Array*) sur la console
setkeycodes	Charge les entrées de la table de correspondance entre scancode et keycode, utile si vous avez des touches inhabituelles sur votre clavier
setleds	Initialise les drapeaux et LED du clavier
setmetamode	Définit la gestion de la touche méta du clavier
setvtrgb	Défini la table de couleur console dans tous les terminaux virtuels
showconsolefont	Affiche la police de l'écran pour la console EGA/VGA
showkey	Affiche les scancodes, keycodes et codes ASCII des touches appuyées sur le clavier
unicode_start	Met le clavier et la console en mode UNICODE. N'utilisez pas ce programme sauf si votre fichier de correspondance est encodé en ISO-8859-1. Pour les autres encodages, cet utilitaire donne de mauvais résultats.
unicode_stop	Ramène le clavier et la console dans le mode avant UNICODE

8.63. Libpipeline-1.5.3

Le paquet Libpipeline contient une bibliothèque pour manipuler des pipelines (tuyaux) de sous-processus de façon flexible et commode.

Temps de construction approximatif: 0.1 SBU
Espace disque requis: 9.3 Mo

8.63.1. Installation de Libpipeline

Préparez la compilation de Libpipeline :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.63.2. Contenu de Libpipeline

Bibliothèque installée: libpipeline.so

Descriptions courtes

`libpipeline` Cette bibliothèque est utilisée pour construire de façon sécurisée des pipelines (tuyaux) entre des sous-processus

8.64. Make-4.3

Le paquet Make contient un programme pour contrôler la génération d'exécutables et d'autres fichiers non-sources d'un paquet à partir des fichiers sources.

Temps de construction approximatif: 0.6 SBU
Espace disque requis: 14 Mo

8.64.1. Installation de Make

Préparez la compilation de Make :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.64.2. Contenu de Make

Programme installé: make

Description courte

make Détermine automatiquement quelles pièces d'un paquet doivent être (re)compilées. Puis, il lance les commandes adéquates

8.65. Patch-2.7.6

Le paquet Patch contient un programme permettant de modifier et de créer des fichiers en appliquant un fichier correctif (appelé habituellement « patch ») créé généralement par le programme **diff**.

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 12 Mo

8.65.1. Installation de Patch

Préparez la compilation de Patch :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, exécutez :

```
make check
```

Installez le paquet :

```
make install
```

8.65.2. Contenu de Patch

Programme installé: patch

Description courte

patch Modifie des fichiers suivant les indications d'un fichier patch, aussi appelé correctif. Un fichier patch est généralement une liste de différences créée par le programme **diff**. En appliquant ces différences sur les fichiers originaux, **patch** crée les versions corrigées.

8.66. Man-DB-2.9.4

Le paquet Man-DB contient des programmes pour trouver et voir des pages de manuel.

Temps de construction approximatif: 0.4 SBU
Espace disque requis: 40 Mo

8.66.1. Installation de Man-DB

Préparez la compilation de man-DB :

```
sed -i '/find/s@/usr@@' init/systemd/man-db.service.in

./configure --prefix=/usr                         \
            --docdir=/usr/share/doc/man-db-2.9.4 \
            --sysconfdir=/etc                     \
            --disable-setuid                      \
            --enable-cache-owner=bin              \
            --with-browser=/usr/bin/lynx          \
            --with-vgrind=/usr/bin/vgrind         \
            --with-grap=/usr/bin/grap
```

Voici la signification des options de configuration :

sed -i '/find/s@/usr@@' init/systemd/man-db.service.in
Cela change un chemin codé en dur vers l'utilitaire **find**, que nous installons dans `/bin`.

`--disable-setuid`
Ceci empêche que le programme **man** se voit attribué l'ID de l'utilisateur `man`.

`--enable-cache-owner=bin`
Cela donne les fichiers de cache du système à l'utilisateur `bin`.

`--with-...`
Ces trois paramètres sont utilisés pour initialiser quelques programmes par défaut. **lynx** est un navigateur Web en mode console (voir BLFS pour les instructions d'installation), **vgrind** convertit du code source de programme en entrée Groff et **grap** est utile pour la composition de texte de graphes dans les documents Groff. Les programmes **vgrind** et **grap** ne sont normalement pas nécessaires pour la visualisation des pages de manuel. Ils ne font pas partie de LFS ou de BLFS, mais vous devriez être capable de les installer vous-même après avoir fini LFS si vous souhaitez faire cela.

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

8.66.2. Pages de manuel non anglaises dans LFS

Le tableau suivant montre l'encodage présumé avec lequel Man-DB encodera les pages de manuel installées dans `/usr/share/man/<ll>`. En outre, Man-DB détermine correctement si les pages de manuel installées dans ce répertoire seront encodées en UTF-8.

Tableau 8.1. Encodage de caractère attendu des pages de manuel 8-bit de base

Langue (code)	Encodage	Langue (code)	Encodage
Danois (da)	ISO-8859-1	Croate (hr)	ISO-8859-2
Allemand (de)	ISO-8859-1	Hongrois (hu)	ISO-8859-2
Anglais (en)	ISO-8859-1	Japonais (ja)	EUC-JP
Espagnol (es)	ISO-8859-1	Coréen (ko)	EUC-KR
Estonien (et)	ISO-8859-1	Lituanien (lt)	ISO-8859-13
Finnois (fi)	ISO-8859-1	Letton (lv)	ISO-8859-13
Français (fr)	ISO-8859-1	Macédonien (mk)	ISO-8859-5
Irlandais (ga)	ISO-8859-1	Polonais (pl)	ISO-8859-2
Galicien (gl)	ISO-8859-1	Roumain (ro)	ISO-8859-2
Indonésien (id)	ISO-8859-1	Russe (ru)	KOI8-R
Islandais (is)	ISO-8859-1	Slovaque (sk)	ISO-8859-2
Italien (it)	ISO-8859-1	Slovène (sl)	ISO-8859-2
Bokmal Norvégien (nb)	ISO-8859-1	Latin serbe (sr@latin)	ISO-8859-2
Hollandais (nl)	ISO-8859-1	Serbe (sr)	ISO-8859-5
Norvégien Nynorsk (nn)	ISO-8859-1	Turc (tr)	ISO-8859-9
Norvégien (no)	ISO-8859-1	Ukrainien (uk)	KOI8-U
Portugais (pt)	ISO-8859-1	Vietnamien (vi)	TCVN5712-1
Suédois (sv)	ISO-8859-1	Chinois simplifié (zh_CN)	GBK
Biélorusse (be)	CP1251	Chinois, Singapour (zh_SG)	GBK
Bulgare (bg)	CP1251	Chinois traditionnel, Hong Kong (zh_HK)	BIG5HKSCS
Tchèque (cs)	ISO-8859-2	Chinois traditionnel (zh_TW)	BIG5
Grec (el)	ISO-8859-7		

Note

Les pages de manuel dans des langues non comprises dans la liste ne sont pas prises en charge.

8.66.3. Contenu de Man-DB

Programmes installés: accessdb, apropos (lien vers whatis), catman, lexgrog, man, mandb, manpath et whatis
Bibliothèques installées: libman.so et libmandb.so (tous deux dans /usr/lib/man-db)
Répertoires installés: /usr/lib/man-db, /usr/libexec/man-db et /usr/share/doc/man-db-2.9.4

Descriptions courtes

accessdb Transforme le contenu de la base de données **whatis** en format lisible par un humain

apropos Recherche la base de données **whatis** et affiche les descriptions courtes des commandes système qui contiennent une chaîne donnée

catman Crée ou met à jour les pages de manuel préformatées

lexgrog Affiche des informations en résumé d'une ligne à propos d'une page de manuel donnée

man Formate et affiche les pages de manuel demandées

mandb Crée ou met à jour la base de données **whatis**

manpath Affiche le contenu de $MANPATH ou (si $MANPATH n'est pas paramétré) d'un chemin de recherche convenable basé sur les paramètres de l'environnement de l'utilisateur

whatis Recherche la base de données **whatis** et affiche les descriptions courtes des commandes système qui contiennent le mot-clé donné sous la forme d'un mot séparé

`libman` Contient le support au moment de l'exécution de **man**

`libmandb` Contient le support au moment de l'exécution de **man**

8.67. Tar-1.34

Le paquet Tar fournit la possibilité de créer des archives tar et effectuer diverses manipulations d'archives. Tar peut être utilisé sur des archives précédemment créées pour extraire des fichiers, ajouter des fichiers supplémentaires, mettre à jour ou lister les fichiers qui étaient déjà stockés.

Temps de construction approximatif: 2.0 SBU
Espace disque requis: 41 Mo

8.67.1. Installation de Tar

Préparez la compilation de Tar :

```
FORCE_UNSAFE_CONFIGURE=1  \
./configure --prefix=/usr \
            --bindir=/bin
```

Voici la signification de l'option de configure :

`FORCE_UNSAFE_CONFIGURE=1`
Ceci oblige le test de `mknod` à se lancer en tant que `root`. On considère généralement que lancer ce test en tant qu'utilisateur `root` est dangereux, mais comme on ne l'exécute que sur un système qui n'a été construit que partiellement, ce remplacement est acceptable.

Compilez le paquet :

```
make
```

Pour tester les résultats (environ 3 SBU), lancez :

```
make check
```

Un test, « capabilities: binary store/restore », est connu pour échouer.

Installez le paquet :

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.34
```

8.67.2. Contenu de Tar

Programmes installés: tar
Répertoire installé: /usr/share/doc/tar-1.34

Descriptions courtes

tar Crée, extrait des fichiers à partir d'archives et liste le contenu d'archives, connues sous le nom d'archives tar

8.68. Texinfo-6.7

Le paquet Texinfo contient des programmes de lecture, écriture et conversion des pages Info.

Temps de construction approximatif: 0.7 SBU
Espace disque requis: 105 Mo

8.68.1. Installation de Texinfo

Préparez la compilation de Texinfo :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour tester les résultats, lancez :

```
make check
```

Installez le paquet :

```
make install
```

De manière optionnelle, installez les composants appartenant à une installation TeX :

```
make TEXMF=/usr/share/texmf install-tex
```

Voici la signification du paramètre de make :

`TEXMF=/usr/share/texmf`
La variable `TEXMF` du Makefile contient l'emplacement de la racine de votre répertoire TeX si, par exemple, un paquet TeX est installé plus tard.

Le système de documentation Info utilise un fichier texte pour contenir sa liste des entrées de menu. Le fichier est situé dans `/usr/share/info/dir`. Malheureusement, à cause de problèmes occasionnels dans les Makefile de différents paquets, il peut être non synchronisé avec les pages info. Si le fichier `/usr/share/info/dir` a besoin d'être recréé, les commandes suivantes accompliront cette tâche :

```
pushd /usr/share/info
  rm -v dir
  for f in *
    do install-info $f dir 2>/dev/null
  done
popd
```

8.68.2. Contenu de Texinfo

Programmes installés: info, install-info, makeinfo (lien vers texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf et texindex
Bibliothèque installée: MiscXS.so, Parsetexi.so et XSParagraph.so (le tout dans /usr/lib/texinfo)
Répertoires installés: /usr/share/texinfo et /usr/lib/texinfo

Descriptions courtes

info Utilisé pour lire des pages info similaires aux pages man mais qui vont souvent plus loin que la simple explication des arguments disponibles. Par exemple, comparez **man bison** et **info bison**.

install-info Utilisé pour installer les pages info ; il met à jour les entrées dans le fichier index d'**info**

makeinfo Traduit les sources Texinfo données dans différents autres langages : pages info, texte ou HTML

pdftexi2dvi Utilisé pour formater le document Texinfo indiqué en un fichier PDF (*Portable Document Format*)

pod2texi Convertit des documents Pod vers le format Texinfo

texi2any Traduit des documentations source Texinfo vers différents autres formats

texi2dvi Utilisé pour formater le document Texinfo indiqué en un fichier indépendant des périphériques, pouvant être imprimé

texi2pdf Utilisé pour formater le document Texinfo indiqué en un fichier PDF (*Portable Document Format*)

texindex Utilisé pour trier les fichiers d'index de Texinfo

8.69. Vim-8.2.2433

Le paquet Vim contient un puissant éditeur de texte.

Temps de construction approximatif: 2.0 SBU
Espace disque requis: 208 Mo

Alternatives à Vim

Si vous préférez un autre éditeur—comme Emacs, Joe, ou Nano—merci de vous référer à *http://fr.linuxfromscratch.org/blfs/../view/blfs-10.1/postlfs/editors.html* pour des instructions d'installation.

8.69.1. Installation de Vim

Tout d'abord, modifiez l'emplacement par défaut du fichier de configuration `vimrc` en `/etc` :

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Préparez la compilation de Vim :

```
./configure --prefix=/usr
```

Compilez le paquet :

```
make
```

Pour préparer les tests, assurez-vous que l'utilisateur `tester` puisse écrire dans l'arborescence des sources :

```
chown -Rv tester .
```

Maintenant lancez les tests en tant qu'utilisateur `tester` :

```
su tester -c "LANG=en_US.UTF-8 make -j1 test" &> vim-test.log
```

La suite de tests affiche à l'écran beaucoup de caractères binaires. Ils peuvent causer des soucis avec les paramètres de votre terminal actuel. Le problème peut se résoudre en redirigeant la sortie vers un journal de traces comme montré ci-dessus. Un test réussi donnera les mots « ALL DONE » à la fin.

Installez le paquet :

```
make install
```

Beaucoup d'utilisateurs sont habitués à utiliser **vi** au lieu de **vim**. Pour permettre l'exécution de **vim** quand les utilisateurs saisissent habituellement **vi**, créez un lien symbolique vers les binaires et vers les pages de man dans les langues fournies :

```
ln -sv vim /usr/bin/vi
for L in  /usr/share/man/{,*/}man1/vim.1; do
    ln -sv vim.1 $(dirname $L)/vi.1
done
```

Par défaut, la documentation de Vim s'installe dans `/usr/share/vim`. Le lien symbolique suivant permet d'accéder à la documentation via `/usr/share/doc/vim-8.2.2433`, en cohérence avec l'emplacement de la documentation d'autres paquets :

```
ln -sv ../vim/vim82/doc /usr/share/doc/vim-8.2.2433
```

Si vous allez installer le système de fenêtrage X sur votre système LFS, il pourrait être nécessaire de recompiler Vim après avoir installé X. Vim fournit une version graphique de l'éditeur qui requiert X et quelques autres bibliothèques pour s'installer. Pour plus d'informations sur ce processus, référez-vous à la documentation de Vim et à la page d'installation de Vim dans le livre BLFS sur *http://fr.linuxfromscratch.org/blfs/../view/blfs-10.1/postlfs/vim.html*.

8.69.2. Configuration de Vim

Par défaut, **vim** est lancé en mode compatible vi. Ceci pourrait être nouveau pour les personnes qui ont utilisé d'autres éditeurs dans le passé. Le paramètre « nocompatible » est inclus ci-dessous pour surligner le fait qu'un nouveau comportement est en cours d'utilisation. Il rappelle aussi à ceux qui voudraient le changer en mode « compatible » qu'il devrait être le premier paramètre dans le fichier de configuration. Ceci est nécessaire car il modifie d'autres paramètres et la surcharge doit survenir après ce paramètre. Créez un fichier de configuration **vim** par défaut en lançant ce qui suit :

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc

" Ensure defaults are set before customizing settings, not after
source $VIMRUNTIME/defaults.vim
let skip_defaults_vim=1

set nocompatible
set backspace=2
set mouse=
syntax on
if (&term == "xterm") || (&term == "putty")
  set background=dark
endif

" End /etc/vimrc
EOF
```

L'option *set nocompatible* change le comportement de **vim** d'une façon plus utile (par défaut) que le comportement compatible vi. Supprimez « no » pour conserver l'ancien comportement de **vi**. Le paramètre *set backspace=2* permet le retour en arrière après des sauts de ligne, l'indentation automatique et le début de l'insertion. L'instruction *syntax on* active la coloration syntaxique. Le paramètre *set mouse=r* permet de coller du texte avec la souris correctement dans un environnement chroot ou au travers d'une connexion à distance. Enfin, l'instruction *if* avec *set background=dark* corrige l'estimation de **vim** concernant la couleur du fond de certains émulateurs de terminaux. Ceci permet d'utiliser de meilleures gammes de couleurs pour la coloration syntaxique, notamment avec les fonds noirs de ces programmes.

La documentation pour les autres options disponibles peut être obtenue en lançant la commande suivante :

```
vim -c ':options'
```

Note

Par défaut, Vim installe uniques les fichiers de dictionnaires pour l'anglais. Pour installer des fichiers de dictionnaires pour votre langue, téléchargez les fichiers `*.spl` et éventuellement, les `*.sug` pour votre langue et votre encodage sur *ftp://ftp.vim.org/pub/vim/runtime/spell/* et enregistrez-les dans `/usr/share/vim/vim82/spell/`.

Pour utiliser ces fichiers dictionnaire, il faut une configuration dans `/etc/vimrc`, comme :

```
set spelllang=en,ru
set spell
```

Pour plus d'informations, voir le fichier README approprié situé sur la page ci-dessus.

8.69.3. Contenu de Vim

Programmes installés: ex (lien vers vim), rview (lien vers vim), rvim (lien vers vim), vi (lien vers vim), view (lien vers vim), vim, vimdiff (lien vers vim), vimtutor, et xxd
Répertoire installé: /usr/share/vim

Descriptions courtes

ex Démarre **vim** en mode ex

rview Une version restreinte de **view** : aucune commande shell ne peut être lancée et **view** ne peut pas être suspendu

rvim Une version restreinte de **vim** : aucune commande shell ne peut être lancée et **vim** ne peut pas être suspendu

vi Lien vers **vim**

view Démarre **vim** en mode lecture seule

vim L'éditeur

vimdiff Édite deux ou trois versions d'un fichier avec **vim** et montre les différences

vimtutor Vous apprend les touches et les commandes basiques de **vim**

xxd Fait un affichage hexa du fichier donné. Il peut aussi faire l'inverse pour une correspondance binaire

8.70. Systemd-247

Le paquet Systemd contient des programmes pour contrôler le démarrage, l'exécution et l'arrêt du système.

Temps de construction approximatif: 1.8 SBU
Espace disque requis: 264 Mo

8.70.1. Installation de Systemd

Tout d'abord, appliquez un correctif pour corriger des régressions :

```
patch -Np1 -i ../systemd-247-upstream_fixes-1.patch
```

Créez un lien symbolique pour éviter un problème avec l’absence de xsltproc :

```
ln -sf /bin/true /usr/bin/xsltproc
```

Mettez en place les pages de manuel :

```
tar -xf ../systemd-man-pages-247.tar.xz
```

Supprimez des tests qui ne peuvent pas être construits en chroot :

```
sed '181,$ d' -i src/resolve/meson.build
```

Supprimez un groupe inutile, `render` des règles udev par défaut :

```
sed -i 's/GROUP="render"/GROUP="video"/' rules.d/50-udev-default.rules.in
```

Préparez la compilation de Systemd :

```
mkdir -p build
cd       build

LANG=en_US.UTF-8                  \
meson --prefix=/usr               \
      --sysconfdir=/etc           \
      --localstatedir=/var        \
      -Dblkid=true                \
      -Dbuildtype=release         \
      -Ddefault-dnssec=no         \
      -Dfirstboot=false           \
      -Dinstall-tests=false       \
      -Dkmod-path=/bin/kmod       \
      -Dldconfig=false            \
      -Dmount-path=/bin/mount     \
      -Drootprefix=               \
      -Drootlibdir=/lib           \
      -Dsplit-usr=true            \
      -Dsulogin-path=/sbin/sulogin  \
      -Dsysusers=false            \
      -Dumount-path=/bin/umount   \
      -Db_lto=false               \
      -Drpmmacrosdir=no           \
      -Dhomed=false               \
      -Duserdb=false              \
      -Dman=true                  \
      -Dmode=release              \
      -Ddocdir=/usr/share/doc/systemd-247 \
      ..
```

Voici la signification des options de meson :

`-D*-path=*`

Ces paramètres fournissent l'emplacement des binaires requis par systemd à l'exécution qui n'ont pas encore été installés.

`-Ddefault-dnssec=no`

Cet argument désactive la prise en charge expérimentale de DNSSEC.

`-Dfirstboot=false`

Ce paramètre empêche l'installation d'un service de systemd qui est responsable de la mise en place du système pour la première fois. Ceux-ci ne sont pas utiles pour LFS car tout est fait manuellement.

`-Dinstall-tests=false`

Ce paramètre évite l'installation des tests compilés.

`-Dldconfig=false`

Ce paramètre empêche l'installation d'une unité de systemd qui lance **ldconfig** au démarrage, inutile pour les distributions source comme LFS et qui rend celui-ci plus long. Retirez ceci si la fonction est désirée.

`-Droot*`

Ces paramètres garantissent que les programmes et les bibliothèques partagées centrales s'installent dans les sous-répertoires de la partition racine.

-Dsplit-usr=true

Ce paramètre garantit que systemd fonctionnera sur des systèmes où les répertoires /bin, /lib et /sbin ne sont pas des liens symboliques vers leurs répertoires /usr correspondant.

-Dsysusers=false

Ce paramètre empêche l'installation de services de systemd responsables de la mise en place des fichiers `/etc/group` et `/etc/passwd`. Ces deux fichiers sont créés plus tôt dans ce chapitre.

-Drpmmacrosdir=no

Ce paramètre désactive l'installation des macros RPM pour systemd parce que LFS ne prend pas en charge RPM.

-D{userdb,homed}=false

Supprimez deux démons qui ont des dépendances qui ne rentrent pas dans le cadre de LFS.

-Dmode=release

Désactivez certaines fonctionnalités considérées comme expérimentales en amont.

Compilez le paquet :

```
LANG=en_US.UTF-8 ninja
```

Installez le paquet :

```
LANG=en_US.UTF-8 ninja install
```

Supprimez un lien symbolique inutile :

```
rm -f /usr/bin/xsltproc
```

Supprimez un répertoire inutile :

```
rm -rf /usr/lib/pam.d
```

Créez le fichier `/etc/machine-id` dont a besoin **systemd-journald** :

```
systemd-machine-id-setup
```

Mettez en place la structure cible de base :

```
systemctl preset-all
```

Désactivez un service connu pour causer des problèmes avec les systèmes qui utilisent une configuration réseau autre que celle fournie par systemd-networkd :

```
systemctl disable systemd-time-wait-sync.service
```

8.70.2. Contenu de systemd

Programmes installés: bootctl, busctl, coredumpctl, halt (lien symbolique vers systemctl), hostnamectl, init, journalctl, kernel-install, localectl, loginctl, machinectl, networkctl, portablectl, poweroff (lien symbolique vers systemctl), reboot (lien symbolique vers systemctl), resolvconf (lien symbolique vers resolvectl), resolvectl, runlevel (lien symbolique vers systemctl), shutdown (lien symbolique vers systemctl), systemctl, systemd-analyze, systemd-ask-password, systemd-cat, systemd-cgls, systemd-cgtop, systemd-delta, systemd-detect-virt, systemd-dissect, systemd-escape, systemd-hwdb, systemd-id128, systemd-inhibit, systemd-machine-id-setup, systemd-mount, systemd-notify, systemd-nspawn, systemd-path, systemd-repart, systemd-resolve (lien symbolique vers resolvectl), systemd-run, systemd-socket-activate, systemd-stdio-bridge, systemd-tmpfiles, systemd-tty-ask-password-agent, systemd-umount (lien symbolique vers systemd-mount), telinit (lien symbolique vers systemctl), timedatectl, and udevadm

Bibliothèques installées: libnss_myhostname.so.2, libnss_mymachines.so.2, libnss_resolve.so.2, libnss_systemd.so.2, libsystemd.so, libsystemd-shared-247.so (dans /lib/systemd) et libudev.so

Répertoires installés: /etc/binfmt.d, /etc/init.d, /etc/kernel, /etc/modules-load.d, /etc/sysctl.d, /etc/systemd, /etc/tmpfiles.d, /etc/udev, /etc/xdg/systemd, /lib/systemd, /lib/udev, /usr/include/systemd, /usr/lib/binfmt.d, /usr/lib/environment.d, /usr/lib/kernel, /usr/lib/modules-load.d, /usr/lib/sysctl.d, /usr/lib/systemd, /usr/lib/tmpfiles.d, /usr/share/doc/systemd-247, /usr/share/factory, /usr/share/systemd, /var/lib/systemd et /var/log/journal

Descriptions courtes

bootctl Est utilisé pour voir les paramètres du microprogramme et du gestionnaire de démarrage

busctl Est utilisé pour examiner et surveiller le bus D-Bus

coredumpctl Est utilisé pour récupérer les fichiers coredumps du journal de Systemd

halt Appelle en principe **shutdown** avec le paramètre *-h*, sauf si on est déjà au niveau d'exécution 0, où il dit au noyau d'arrêter le système ; il inscrit dans le fichier `/var/log/wtmp` que le système va s'éteindre

hostnamectl Est utilisé pour voir et changer le nom d'hôte du système et les paramètres afférents

init Est le premier processus qui démarre lorsque le noyau a initialisé le matériel, qui poursuit le processus de démarrage et lance les processus qu'on lui a demandés. Dans ce cas, il démarre systemd

journalctl Est utilisé pour voir le contenu du journal de systemd

kernel-install Est utilisé pour ajouter et supprimer des images de noyau et des initramfs dans /boot. Dans LFS, on le fait manuellement

localectl Est utilisé pour voir et modifier les paramètres linguistiques et de la disposition clavier du système

loginctl Est utilisé pour examiner et contrôler l'état du gestionnaire d'identification de systemd

machinectl Est utilisé pour examiner et contrôler l'état du gestionnaire des enregistrements de machines virtuelles et de conteneurs de systemd

networkctl Est utilisé pour examiner et configurer l'état des liens réseaux configurés par systemd-networkd

portablectl Est utilisé pour attacher ou détacher des services portables d'un système local

poweroff Dit au noyau d'arrêter le système et d'éteindre l'ordinateur (voir **halt**)

reboot Dit au noyau de redémarrer le système (voir **halt**)

resolvconf Enregistre la configuration des serveurs DNS et des domaines avec **systemd-resolved**

resolvectl Envoie des commandes de contrôle au gestionnaire de résolution de noms réseaux ou résout des noms de domaines, des adresses IPv4 ou IPv6, des enregistrements DNS et des services

runlevel Renvoie le niveau d'exécution actuel et précédent indiqué dans le journal du dernier niveau d'exécution situé dans `/run/utmp`

shutdown Éteint le système de manière sécurisée en envoyant à tous les processus un signal et en notifiant tous les utilisateurs connectés

systemctl Est utilisé pour examiner et contrôler l'état du gestionnaire des services du système de systemd

systemd-analyze Est utilisé pour déterminer les performances de démarrage du système lors du dernier démarrage et pour identifier les unités problématiques

systemd-ask-password Est utilisé pour afficher le mot ou la phrase de passe d'un utilisateur en utilisant une question définie en ligne de commande

systemd-cat Est utilisé pour relier les STDOUT et STDERR d'un processus au journal de systemd

systemd-cgls Affiche de manière récursive le contenu de la hiérarchie des groupes de Linux sélectionnée dans une arborescence

systemd-cgtop Affiche les groupes de contrôle dans la hiérarchie des groupes du Linux local, triés par processus, charge mémoire et d'E/S disque

systemd-delta Est utilisé pour identifier et comparer des fichiers de configuration de `/etc`, qui ont leur correspondance par défaut dans `/usr`

systemd-detect-virt Détecte si le système est lancé dans un environnement virtuel, et ajuste udev en fonction

systemd-dissect est utilisé pour inspecter les images disques de systèmes d'exploitations

systemd-escape Est utilisé pour échapper les chaînes de caractères pour les utiliser dans les noms d'unités de systemd

systemd-hwdb Est utilisé pour gérer la base de données matérielle (hwdb)

systemd-id128 Génère et afficher des chaînes id128

systemd-inhibit Est utilisé pour exécuter un programme avec un marqueur d'extinction, de veille ou d'inactivité qui le verrouille, pour éviter une action comme l'arrêt du système jusqu'à ce que le processus termine

systemd-machine-id-setup Est utilisé par les outils d'installation de systemd pour initialiser l'ID de la machine contenu dans `/etc/machine-id` lors de l'installation, avec un ID généré de façon aléatoire

systemd-mount Est utilisé pour monter ou monter automatiquement un périphérique temporairement

systemd-notify Est utilisé par les démons pour indiquer au système de démarrage les changements d'état

systemd-nspawn Est utilisé pour lancer une commande ou un OS dans un conteneur léger (espace de nom séparé)

systemd-path Est utilisé pour voir les path système et utilisateur

systemd-repart Est utilisé pour agrandir et ajouter des partitions à une table de partition quand systemd est utilisé dans un image d'OS (p. ex. dans un conteneur)

systemd-resolve Est utilisé pour résoudre les noms de domaines, les adresses IPV4 et IPV6, les enregistrements DNS et les services

systemd-run Est utilisé pour créer et démarrer une unité transitoire .service ou .scope et pour y lancer la commande spécifiée. C'est utile pour valider des unités systemd

systemd-socket-activate Est utilisé pour écouter sur des périphériques de socket et lancer un processus lors d'une connexion

systemd-tmpfiles Crée, efface et nettoie les fichiers et les répertoires volatiles et temporaires sur la base d'un format et d'un emplacement du fichier de configuration indiqué dans les répertoires `tmpfiles.d`

systemd-umount Démonte des points de montage

systemd-tty-ask-password-agent Est utilisé pour lister ou traiter les requêtes de mots de passe en attente

telinit Dit à **init** quel niveau d'exécution adopter

timedatectl Est utilisé pour voir et modifier l'horloge du système et ses paramètres

udevadm Est un outil générique d'administration d'Udev qui contrôle le démon udevd, donne des infos à partir de la base de données matérielle d'Udev, surveille les uevents, attend la fin des uevents, teste la configuration d'Udev et déclenche des événements pour un périphérique donné

`libsystemd` Est la bibliothèque d'outils principale de systemd

`libudev` Est une bibliothèque pour accéder aux informations d'Udev

8.71. D-Bus-1.12.20

D-Bus est un système de bus de messages, une manière simple pour les applications de se parler. D-Bus fournit un démon système (pour les événements comme « l'ajout de nouveaux matériels » ou le « changement de la file d'impression ») et un démon individuel à chaque utilisateur connecté (pour les besoins généraux de communication entre les processus des applications de l'utilisateur). De plus, le bus des messages est construit sur la base d'un environnement de circulation des messages par communication directe, ce qui peut être utilisé par deux applications pour communiquer directement (sans passer par le démon de bus de messages).

Temps de construction approximatif: 0.2 SBU
Espace disque requis: 18 Mo

8.71.1. Installation de D-Bus

Préparez la compilation de D-Bus :

```
./configure --prefix=/usr                       \
            --sysconfdir=/etc                   \
            --localstatedir=/var                \
            --disable-static                    \
            --disable-doxygen-docs              \
            --disable-xml-docs                  \
            --docdir=/usr/share/doc/dbus-1.12.20 \
            --with-console-auth-dir=/run/console \
            --with-system-pid-file=/run/dbus/pid \
            --with-system-socket=/run/dbus/system_bus_socket
```

Voici la signification des options de configuration :

`--with-console-auth-dir=/run/console`
Cela indique l'emplacement du répertoire d'autorisation de ConsoleKit.

`--with-system-pid-file=/run/dbus/pid` et *`--with-system-socket=/run/dbus/system_bus_socket`*
Ces paramètres indiquent que l'emplacement du fichier de PID et du socket du bus système sera dans `/run` au lieu du répertoire obsolète `/var/run`.

Compilez le paquet :

```
make
```

Ce paquet est fourni avec une suite de tests, mais elle exige des paquets non inclus dans LFS. Vous pouvez trouver des instructions pour lancer la suite de tests dans BLFS, à *http://fr.linuxfromscratch.org/blfs/../view/blfs-10.1/general/dbus.html*.

Installez le paquet :

```
make install
```

Il faut déplacer la bibliothèque partagée dans `/lib` et, par conséquent, le fichier `.so` de `/usr/lib` devra être recréé :

```
mv -v /usr/lib/libdbus-1.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libdbus-1.so) /usr/lib/libdbus-1.so
```

Créez un lien symbolique pour que D-Bus et systemd puissent utiliser le même fichier `machine-id` :

```
ln -sv /etc/machine-id /var/lib/dbus
```

8.71.2. Contenu de D-Bus

Programmes installés: dbus-cleanup-sockets, dbus-daemon, dbus-launch, dbus-monitor, dbus-run-session, dbus-send, dbus-test-tool, dbus-update-activation-environment et dbus-uuidgen
Bibliothèques installées: libdbus-1.{a,so}
Répertoires installés: /etc/dbus-1, /usr/include/dbus-1.0, /usr/lib/dbus-1.0, /usr/share/dbus-1, /usr/share/doc/dbus-1.12.20, et /var/lib/dbus

Descriptions courtes

dbus-cleanup-sockets	est utilisé pour nettoyer les sockets fantômes d'un répertoire
dbus-daemon	Est le démon de bus de messages D-Bus
dbus-launch	Démarre **dbus-daemon** à partir d'un script shell
dbus-monitor	Surveille les messages circulant dans un bus de message D-Bus
dbus-run-session	Démarre une instance de bus de session de **dbus-daemon** à partir d'un scipt shell et démarre un programme spécifié dans cette session
dbus-send	Envoie un message à un bus de message D-Bus
dbus-test-tool	Est un outil pour aider les paquets à tester D-Bus
dbus-update-activation-environment	Met à jour les variables d'environnement qui seront initialisées pour les services de session de D-Bus
dbus-uuidgen	Génère un ID universel et unique
`libdbus-1`	Contient les fonctions de l'API utilisée pour communiquer avec le bus de message de D-Bus

8.72. Procps-3.3.17

Le paquet Procps-ng contient des programmes pour surveiller les processus.

Note

Ce paquet s'extrait vers le répertoire `procps-3.3.17` et non le répertoire attendu `procps-ng-3.3.17`.

Temps de construction approximatif: 0.5 SBU
Espace disque requis: 20 Mo

8.72.1. Installation de Procps-ng

Préparez maintenant la compilation de procps-ng :

```
./configure --prefix=/usr                            \
            --exec-prefix=                           \
            --libdir=/usr/lib                        \
            --docdir=/usr/share/doc/procps-ng-3.3.17 \
            --disable-static                         \
            --disable-kill                           \
            --with-systemd
```

Voici la signification de l'option de configure :

`--disable-kill`
Cette option désactive la construction de la commande **kill** installée dans le paquet util-linux.

Compilez le paquet :

```
make
```

Pour lancer la suite de tests, lancez :

```
make check
```

Cinq tests liés à pkill sont connus pour échouer à cause d'un problème avec les tests qui n'ont pas été mis à jour.

Installez le paquet :

```
make install
```

Enfin, déplacez les bibliothèques essentielles à un endroit où elles seront trouvables si `/usr` n'est pas monté.

```
mv -v /usr/lib/libprocps.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libprocps.so) /usr/lib/libprocps.so
```

8.72.2. Contenu de Procps-ng

Programmes installés: free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w et watch
Bibliothèque installée: libprocps.so
Répertoires installés: /usr/include/proc et /usr/share/doc/procps-ng-3.3.17

Descriptions courtes

free	Indique le total de mémoire libre et utilisé sur le système à la fois pour la mémoire physique et pour la mémoire swap
pgrep	Recherche les processus suivant leur nom et autres attributs
pidof	Indique le PID des programmes précisés
pkill	Envoie des signaux aux processus suivant leur nom et autres attributs
pmap	Affiche le plan mémoire du processus désigné
ps	Donne un aperçu des processus en cours d'exécution
pwait	Attend qu'un processus termine avant de s'exécuter.
pwdx	Indique le répertoire d'exécution courant d'un processus
slabtop	Affiche des informations détaillées sur le cache slab du noyau en temps réel
sysctl	Modifie les paramètres du noyau en cours d'exécution
tload	Affiche un graphe de la charge système actuelle
top	Affiche une liste des processus demandant le maximum de ressources CPU ; il fournit un affichage agréable sur l'activité du processeur en temps réel
uptime	Affiche le temps d'exécution du système, le nombre d'utilisateurs connectés et les moyennes de charge système
vmstat	Affiche les statistiques de mémoire virtuelle, donne des informations sur les processus, la mémoire, la pagination, le nombre de blocs en entrées/sorties, les échappements et l'activité CPU
w	Affiche les utilisateurs actuellement connectés, où et depuis quand
watch	Lance une commande de manière répétée, affichant le premier écran de sa sortie ; ceci vous permet de surveiller la sortie
`libprocps`	Contient les fonctions utilisées par la plupart des programmes de ce paquet

8.73. Util-linux-2.36.2

Le paquet Util-linux contient différents outils. Parmi eux se trouvent des outils de gestion des systèmes de fichiers, de consoles, de partitions et des messages.

Temps de construction approximatif: 1.1 SBU
Espace disque requis: 262 Mo

8.73.1. Installation d'Util-linux

Préparez la compilation d'Util-linux :

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime   \
            --docdir=/usr/share/doc/util-linux-2.36.2 \
            --disable-chfn-chsh  \
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
            --disable-runuser    \
            --disable-pylibmount \
            --disable-static     \
            --without-python     \
            runstatedir=/run
```

Les options --disable et --without préviennent des avertissements à propos d'éléments de construction qui requièrent des paquets non compris dans LFS ou incohérents avec les programmes installés par d'autres paquets.

Compilez le paquet :

```
make
```

Si vous le souhaitez, lancez la suite de tests en tant qu'utilisateur non root :

Avertissement

L'exécution de la suite de tests en tant qu'utilisateur `root` peut être dangereuse pour votre système. Pour la lancer, l'option CONFIG_SCSI_DEBUG du noyau doit être disponible sur le système en cours d'exécution et doit être construite en tant que module. Si elle est compilée en dur dans le noyau, cela empêchera de démarrer. Pour une exécution complète, il faut installer d'autres paquets de BLFS. Si vous le souhaitez, vous pouvez lancer ce test après le redémarrage dans le système LFS terminé, en exécutant :

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

```
chown -Rv tester .
su tester -c "make -k check"
```

Installez le paquet :

```
make install
```

8.73.2. Contenu d'Util-linux

Programmes installés: addpart, agetty, blkdiscard, blkid, blkzone, blockdev, cal, cfdisk, chcpu, chmem, choom, chrt, col, colcrt, colrm, column, ctrlaltdel, delpart, dmesg, eject, fallocate, fdformat, fdisk, fincore, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipcs, isosize, kill, last, lastb (link to last), ldattach, linux32, linux64, logger, look, losetup, lsblk, lscpu, lsipc, lslocks, lslogins, lsmem, lsns, mcookie, mesg, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, nsenter, partx, pivot_root, prlimit, raw, readprofile, rename, renice, resizepart, rev, rfkill, rtcwake, script, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff (link to swapon), swapon, switch_root, taskset, ul, umount, uname26, unshare, utmpdump, uuidd, uuidgen, uuidparse, wall, wdctl, whereis, wipefs, x86_64 et zramctl
Bibliothèques installées: libblkid.so, libfdisk.so, libmount.so, libsmartcols.so et libuuid.so
Répertoires installés: /usr/include/blkid, /usr/include/libfdisk, /usr/include/libmount, /usr/include/libsmartcols, /usr/include/uuid, /usr/share/doc/util-linux-2.36.2 et /var/lib/hwclock

Descriptions courtes

addpart Informe le noyau Linux de nouvelles partitions

agetty Ouvre un port tty, demande un nom de connexion puis appelle le programme **login**

blkdiscard Désactive des secteurs d'un périphérique

blkid Un outil en ligne de commande pour trouver et afficher les attributs d'un périphérique bloc

blkzone Lance des commandes de zone sur le périphérique bloc donné

blockdev Permet aux utilisateurs d'appeler les ioctl d'un périphérique bloc à partir de la ligne de commande

cal Affiche un calendrier simple

cfdisk Manipule la table des partitions du périphérique donné

chcpu Modifie l'état des processeurs

chmem Configure la mémoire

choom Affiche et ajuste les scores de l'OOM-killer

chrt Manipule les attributs d'un processus en temps réel

col Filtre les retours de chariot inversés

colcrt Filtre la sortie de **nroff** pour les terminaux manquant de capacités comme le texte barré ou les demi-lignes

colrm Filtre les colonnes données

column Formate un fichier donné en plusieurs colonnes

ctrlaltdel Initialise la combinaison des touches Ctrl+Alt+Del pour une réinitialisation matérielle ou logicielle

delpart Demande au noyau Linux de supprimer une partition

dmesg Affiche les messages du noyau lors du démarrage

eject Éjecte un média amovible

fallocate Pré-alloue de l'espace à un fichier

fdformat Réalise un formatage de bas niveau sur un disque amovible

fdisk Manipule la table des partitions du périphérique donné

fincore Compte les pages de contenu d'un fichier en mémoire

findfs Trouve un système de fichiers par étiquette ou UUID (*Universally Unique Identifier*, soit Identifiant Unique Universel)

findmnt Est une interface en ligne de commande avec la bibliothèque libmount pour du travail avec les fichiers mountinfo, fstab et mtab

flock Acquiert le verrouillage d'un fichier puis exécute une commande en maintenant le verrouillage

fsck Est utilisé pour vérifier, et parfois réparer, les systèmes de fichiers

fsck.cramfs Réalise un test de cohérence sur le système de fichiers Cramfs du périphérique donné

fsck.minix Réalise un test de cohérence sur le système de fichiers Minix du périphérique donné

fsfreeze Est une enveloppe très simple autour des opérations du pilote noyau FIFREEZE/FITHAW ioctl

fstrim Écarte les blocs inutilisés sur un système de fichiers monté

getopt Analyse les options sur la ligne de commande donnée

hexdump Affiche le fichier indiqué en hexadécimal ou dans un autre format donné

hwclock Lit ou initialise l'horloge du matériel, aussi appelée horloge RTC (*Real-Time Clock*, horloge à temps réel) ou horloge du BIOS (*Basic Input-Output System*)

i386 Un lien symbolique vers setarch

ionice Obtient ou initialise la classe de planification IO (ES) et la priorité pour un programme

ipcmk Crée diverses ressources IPC

ipcrm Supprime la ressource IPC (inter-process communication) donnée

ipcs Fournit l'information de statut IPC

isosize Affiche la taille d'un système de fichiers iso9660

kill Envoie des signaux aux processus

last Affiche les utilisateurs connectés (et déconnectés) dernièrement en s'appuyant sur le fichier `/var/log/wtmp` ; il affiche également les démarrages du système, les extinctions et les changements de niveau d'exécution

lastb Affiche les tentatives de connexions enregistrées dans `/var/log/btmp`

ldattach Attache une discipline de ligne à une ligne série

linux32 Un lien symbolique vers setarch

linux64 Un lien symbolique vers setarch

logger Enregistre le message donné dans les traces système

look Affiche les lignes commençant par la chaîne donnée

losetup Initialise et contrôle les périphériques loop

lsblk Liste les informations sur tous les périphériques blocs ou ceux sélectionnés dans un format semblable à une arborescence

lscpu Affiche des informations sur l'architecture du processeur

lsipc Affiche les informations sur les fonctions IPC actuellement utilisées sur le système

lslocks Liste les verrous du système local

lslogins Liste les informations sur les comptes utilisateurs, groupes et systèmes

lsmem Liste les intervalles de mémoire disponibles avec leur statut en ligne

lsns Liste les espaces de noms

mcookie	Génère des cookies magiques, nombres hexadécimaux aléatoires sur 128 bits, pour **xauth**
mesg	Contrôle si d'autres utilisateurs peuvent envoyer des messages au terminal de l'utilisateur actuel
mkfs	Construit un système de fichiers sur un périphérique (habituellement une partition du disque dur)
mkfs.bfs	Crée un système de fichiers bfs de SCO (Santa Cruz Operations)
mkfs.cramfs	Crée un système de fichiers cramfs
mkfs.minix	Crée un système de fichiers Minix
mkswap	Initialise le périphérique ou le fichier à utiliser comme swap
more	Est un filtre pour visualiser un texte un écran à la fois
mount	Attache le système de fichiers du périphérique donné sur un répertoire spécifié dans le système de fichiers
mountpoint	Vérifie si le répertoire est un point de montage
namei	Affiche les liens symboliques dans les chemins donnés
nsenter	Lance un programme avec un nom espacé des autres processus
partx	Signale au noyau la présence et le nombre de partitions sur un disque
pivot_root	Fait en sorte que le système de fichiers donné soit le nouveau système de fichiers racine du processus actuel
prlimit	Récupère et envoie la limite des ressources d'un processus
raw	Envoie un périphérique de caractère de base Linux vers un périphérique de bloc
readprofile	Lit les informations de profilage du noyau
rename	Renomme les fichiers donnés, remplaçant une chaîne donnée par une autre
renice	Modifie la priorité des processus exécutés
resizepart	Demande au noyau Linux de redimensionner une partition
rev	Inverse les lignes d'un fichier donné
rkfill	Outil pour activer et désactiver les périphériques sans fil
rtcwake	Utilisé pour mettre un système en sommeil jusqu'à un moment de réveil spécifié
script	Crée un script type à partir d'une session du terminal
scriptreplay	Rejoue des scripts type en utilisant les informations de temps
setarch	Change d'architecture signalée dans un nouvel environnement de programme et initialise les commutateurs adéquats
setsid	Lance le programme donné dans une nouvelle session
setterm	Initialise les attributs du terminal
sfdisk	Est un manipulateur de table de partitions disque
sulogin	Permet la connexion de `root`. Il est normalement appelé par **init** lorsque le système passe en mono-utilisateur
swaplabel	Permet de modifier l'UUID et l'étiquette d'un espace d'échange
swapoff	Désactive des périphériques et des fichiers pour la pagination et l'échange
swapon	Active les périphériques et fichiers de pagination et d'échange et liste les périphériques et fichiers en cours d'utilisation
switch_root	Change de système de fichiers racine pour une arborescence montée

tailf	Observe la croissance d'un fichier journal. Affiche les 10 dernières lignes d'un fichier journal, puis continue à afficher toute nouvelle entrée dans le fichier journal dès qu'elle est créée
taskset	Récupère ou initialise un processus vis-à-vis du processeur
ul	Un filtre pour traduire les soulignements en séquences d'échappement indiquant un soulignement pour le terminal utilisé
umount	Déconnecte un système de fichiers à partir de la hiérarchie de fichiers du système
uname26	Un lien symbolique vers setarch
unshare	Lance un programme avec quelques espaces de nom non partagés avec le parent
utmpdump	Affiche le contenu du fichier de connexion donné dans un format convivial
uuidd	Un démon utilisé par la bibliothèque UUID pour générer des UUIDs basés sur l'heure de manière sécurisée et avec une garantie d'unicité
uuidgen	Crée un nouvel UUID. Chaque nouvel UUID peut être raisonnablement considéré unique parmi tous les UUID créés, sur le système local mais aussi sur les autres, dans le passé et dans le futur.
uuidparse	Un utilitaire pour analyser des identifiants uniques
wall	Affiche le contenu d'un fichier ou, par défaut, son entrée standard, sur les terminaux de tous les utilisateurs actuellement connectés
wdctl	Affiche l'état du watchdog matériel
whereis	Affiche l'emplacement du binaire, les sources et la page de manuel de la commande donnée
wipefs	Nettoie la signature d'un système de fichiers à partir du périphérique
x86_64	Un lien symbolique vers setarch
zramctl	Un programme pour configurer et contrôler les périphériques zram (mémoire ram compressée)
`libblkid`	Contient des routines pour l'identification des périphériques et l'extraction des modèles
`libfdisk`	Contient des routines pour la manipulation de table de partition
`libmount`	Contient les routines pour le montage et le démontage des périphériques de bloc
`libsmartcols`	Contient les routines pour la sortie d'écran d'aide sous forme de tableau
`libuuid`	Contient des routines pour la génération d'identifiants uniques pour des objets qui peuvent être accessibles en dehors du système local

8.74. E2fsprogs-1.46.1

Le paquet e2fsprogs contient les outils de gestion du système de fichiers `ext2`. Il prend aussi en charge les systèmes de fichiers journalisés `ext3` et `ext4`.

Temps de construction approximatif: 4.4 SBU on a spinning disk, 1.5 SBU on an SSD
Espace disque requis: 102 Mo

8.74.1. Installation de E2fsprogs

La documentation recommande de construire e2fsprogs dans un sous-répertoire du répertoire source :

```
mkdir -v build
cd       build
```

Préparez la compilation d'e2fsprogs :

```
../configure --prefix=/usr           \
             --bindir=/bin           \
             --with-root-prefix=""   \
             --enable-elf-shlibs     \
             --disable-libblkid      \
             --disable-libuuid       \
             --disable-uuidd         \
             --disable-fsck
```

Voici la signification des options de configure :

`--with-root-prefix=""` et *`--bindir=/bin`*
Certains programmes (comme **e2fsck**) sont considérés comme essentiels. Quand, par exemple, `/usr` n'est pas monté, ces programmes essentiels doivent encore être disponibles. Ils appartiennent aux répertoires comme `/lib` et `/sbin`. Si cette option n'est pas passée au configure, les programmes sont placés dans le répertoire `/usr`.

`--enable-elf-shlibs`
Ceci crée les bibliothèques partagées que certains programmes de ce paquet utilisent.

`--disable-*`
Ceci empêche e2fsprogs de construire et d'installer les bibliothèques `libuuid` et `libblkid`, le démon `uuidd` et le script enveloppe **fsck**, car util-Linux installe des versions plus récentes.

Compilez le paquet :

```
make
```

Pour lancer les tests, lancez :

```
make check
```

Un test, m_rootdir_acl, est connu pour échouer.

Sur un disque à plateaux, les tests prennent un peu plus de 4 SBU. Ils peuvent être bien plus courts sur un SSD (jusqu'à 1,5 SBU).

Installez le paquet :

```
make install
```

Supprimez des bibliothèques statiques inutiles :

```
rm -fv /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Ce paquet installe un fichier `.info` gzippé mais ne met pas à jour le fichier `dir` du système. Dézippez ce fichier puis mettez à jour le fichier `dir` du système en utilisant les commandes suivantes :

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir /usr/share/info/libext2fs.info
```

Si vous le désirez, créez et installez de la documentation supplémentaire en lançant les commandes suivantes :

```
makeinfo -o      doc/com_err.info ../lib/et/com_err.texinfo
install -v -m644 doc/com_err.info /usr/share/info
install-info --dir-file=/usr/share/info/dir /usr/share/info/com_err.info
```

8.74.2. Contenu de E2fsprogs

Programmes installés: badblocks, chattr, compile_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2label, e2mmpstatus, e2scrub, e2scrub_all, e2undo, e4crypt, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mklost+found, resize2fs et tune2fs

Bibliothèques installées: libcom_err.so, libe2p.so, libext2fs.so, et libss.so

Répertoires installés: /usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/ss, /usr/lib/e2fsprogs, /usr/share/et et /usr/share/ss

Descriptions courtes

badblocks Recherche les blocs défectueux sur un périphérique (habituellement une partition d'un disque)

chattr Modifie les attributs de fichiers sur un système de fichiers `ext2` et `ext3`, la version journalisée d'`ext2`

compile_et Un compilateur de table d'erreurs. Il convertit une table de noms d'erreurs et des messages associés en un fichier source C à utiliser avec la bibliothèque `com_err`

debugfs Un débogueur de système de fichiers. Il est utilisé pour examiner et modifier l'état d'un système de fichiers `ext2`

dumpe2fs Affiche le superbloc et les informations de groupes de blocs sur le système de fichiers présent sur un périphérique donné

e2freefrag Raporte les informations de fragmentation de l'espace libre

e2fsck Est utilisé pour vérifier et, si demandé, réparer les systèmes de fichiers `ext2` et `ext3`

e2image Est utilisé pour sauver les données critiques d'un système de fichiers `ext2` dans un fichier

e2label Affiche ou modifie le label d'un système de fichiers `ext2` présent sur un périphérique donné

e2mmpstatus Vérifie le statut MMP d'un système de fichier ext4

e2scrub Vérifie le contenu d'un système de fichiers ext[234] monté

e2scrub_all Vérifie qu'aucun système de fichiers ext[234] monté n'a d'erreur

e2undo Rejoue le journal d'annulation undo_log pour un système de fichiers ext2/ext3/ext4 trouvé sur un périphérique. Il peut être utilisé pour annuler une opération échouée par un programme e2fsprogs.

e4crypt Utilitaire de chiffrement de système de fichiers ext4

e4defrag Défragmenteur en ligne des systèmes de fichiers ext4

filefrag	Signale le niveau de fragmentation que pourrait atteindre un fichier en particulier
fsck.ext2	Vérifie par défaut les systèmes de fichiers `ext2`. C'est un lien vers **e2fsck**
fsck.ext3	Vérifie par défaut les systèmes de fichiers `ext3`. C'est un lien vers **e2fsck**
fsck.ext4	Vérifie par défaut les systèmes de fichiers `ext4`. C'est un lien vers **e2fsck**
logsave	Sauvegarde la sortie d'une commande dans un journal applicatif
lsattr	Liste les attributs de fichiers sur un système de fichiers ext2 (second extended file system)
mk_cmds	Convertit une table de noms de commandes et de messages d'aide en un fichier source C bon à utiliser avec la bibliothèque sous-système `libss`
mke2fs	Crée un système de fichiers `ext2` ou `ext3` sur le périphérique donné
mkfs.ext2	Crée par défaut un système de fichiers `ext2`. C'est un lien vers **mke2fs**
mkfs.ext3	Crée par défaut un système de fichiers `ext3`. C'est un lien vers **mke2fs**
mkfs.ext4	Crée par défaut un système de fichiers `ext4`. C'est un lien vers **mke2fs**
mklost+found	Utilisé pour créer un répertoire `lost+found` sur un système de fichiers `ext2` ; il pré-alloue des blocs de disque à ce répertoire pour alléger la tâche d'**e2fsck**
resize2fs	Utilisé pour agrandir ou réduire un système de fichiers `ext2`
tune2fs	Ajuste les paramètres d'un système de fichiers `ext2`
`libcom_err`	La routine d'affichage d'erreurs
`libe2p`	Est utilisé par **dumpe2fs**, **chattr**, et **lsattr**
`libext2fs`	Contient des routines pour permettre aux programmes de niveau utilisateur de manipuler un système de fichiers `ext2`
`libss`	Est utilisé par **debugfs**

8.75. À propos des symboles de débogage

La plupart des programmes et des bibliothèques sont compilés, par défaut, en incluant les symboles de débogage (avec l'option *-g* de **gcc**). Ceci signifie que, lors du débogage d'un programme ou d'une bibliothèque compilés avec les informations de débogage, le débogueur peut vous donner non seulement les adresses mémoire mais aussi le nom des routines et des variables.

Néanmoins, l'intégration de ces symboles de débogage font grossir le programme ou la bibliothèque de façon significative. Ce qui suit est un exemple de l'espace occupé par ces symboles :

- Un binaire **bash** avec les symboles de débogage : 1 200 Ko
- Un binaire **bash** sans les symboles de débogage : 480 Ko
- Les fichiers Glibc et GCC (`/lib` et `/usr/lib`) avec les symboles de débogage : 87 Mo
- Les fichiers Glibc et GCC sans les symboles de débogage : 16 Mo

Les tailles peuvent varier suivant le compilateur et la bibliothèque C utilisée mais, lors d'une comparaison de programmes avec et sans symboles de débogages, la différence sera généralement d'un facteur de deux à cinq.

Comme la plupart des gens n'utiliseront jamais un débogueur sur leur système, beaucoup d'espace disque peut être gagné en supprimant ces symboles. La prochaine section montre comment supprimer tous les symboles de débogage des programmes et bibliothèques.

8.76. Supprimer de nouveau les symboles des fichiers objets

Cette section est facultative. Si l'utilisateur initial n'est pas un développeur et ne pense pas faire de débogage sur les logiciels du système, la taille du système peut être diminué d'environ 2 Go en supprimant les symboles de débogage contenus dans les binaires et dans les bibliothèques. Ceci ne pose pas de problème autre que le fait de ne plus pouvoir les déboguer.

La plupart des personnes qui utilisent la commande mentionnée ci-dessous ne rencontrent aucune difficulté. Néanmoins, il est facile de faire une erreur de saisie et rendre le nouveau système complètement inutilisable, donc avant d'exécuter la commande **strip**, il est recommandé de faire une sauvegarde de l'état actuel du système LFS.

Tout d'abord placez les symboles des bibliothèques choisies dans des fichiers séparés. Ces informations sont requises si vous lancez des tests de régression qui utilisent *valgrind* ou *gdb* plus tard dans BLFS.

```
save_lib="ld-2.33.so libc-2.33.so libpthread-2.33.so libthread_db-1.0.so"

cd /lib

for LIB in $save_lib; do
    objcopy --only-keep-debug $LIB $LIB.dbg
    strip --strip-unneeded $LIB
    objcopy --add-gnu-debuglink=$LIB.dbg $LIB
done

save_usrlib="libquadmath.so.0.0.0 libstdc++.so.6.0.28
             libitm.so.1.0.0 libatomic.so.1.2.0"

cd /usr/lib

for LIB in $save_usrlib; do
    objcopy --only-keep-debug $LIB $LIB.dbg
    strip --strip-unneeded $LIB
    objcopy --add-gnu-debuglink=$LIB.dbg $LIB
done

unset LIB save_lib save_usrlib
```

Maintenant, les binaires et les bibliothèques peuvent être nettoyées :

```
find /usr/lib -type f -name \*.a \
   -exec strip --strip-debug {} ';'

find /lib /usr/lib -type f -name \*.so* ! -name \*dbg \
   -exec strip --strip-unneeded {} ';'

find /{bin,sbin} /usr/{bin,sbin,libexec} -type f \
    -exec strip --strip-all {} ';'
```

Un grand nombre de fichiers seront rapportés comme ayant un format non reconnu. Ces messages d'avertissement indiquent que ces fichiers sont des scripts et non pas des binaires.

8.77. Nettoyer

Enfin, nettoyez des fichiers résultant des tests :

```
rm -rf /tmp/*
```

Maintenant déconnectez-vous et entrez de nouveau dans l'environnement chroot avec une nouvelle commande chroot. À partir de maintenant, utilisez cette nouvelle commande chroot à chaque fois que vous devrez entrer dans l'environnement chroot après l'avoir quitté :

```
logout

chroot "$LFS" /usr/bin/env -i          \
    HOME=/root TERM="$TERM"            \
    PS1='(lfs chroot) \u:\w\$ '        \
    PATH=/bin:/usr/bin:/sbin:/usr/sbin \
    /bin/bash --login
```

Ici l'option *+h* n'est plus utilisée, comme tous les programmes précédents ont été remplacés : le hashage est donc possible.

Si les systèmes de fichiers virtuels du noyau ont été démontés, manuellement ou suite à un redémarrage, assurez-vous que les systèmes de fichiers virtuels du noyau seront montés lorsque vous entrerez à nouveau dans le chroot. On a expliqué cette procédure dans Section 7.3.2, « Monter et peupler /dev » et Section 7.3.3, « Monter les systèmes de fichiers virtuels du noyau ».

Il y a aussi de nombreux fichiers installés dans les répertoires /usr/lib et /usr/libexec dont l'extension est .la. Ce sont les fichiers « d'archive libtool ». Comme on l'a déjà dit, ils ne sont utiles que pour lier des bibliothèques statiques. Ils sont inutiles, et potentiellement dangereux, lorsqu'on utilise des bibliothèque partagée, surtout avec les systèmes de construction non autotools. Pour les supprimer, lancez :

```
find /usr/lib /usr/libexec -name \*.la -delete
```

Pour plus d'informations sur les fichiers d'archive libtool, voir la *section BLFS « À propos des fichiers d'archive libtool (.la) »*.

Le compilateur construit dans Chapitre 6 et Chapitre 7 est toujours partiellement installé et n'est plus requis. Supprimez-le avec :

```
find /usr -depth -name $(uname -m)-lfs-linux-gnu\* | xargs rm -rf
```

Le répertoire `/tools` peut aussi être supprimé pour gagner un peu plus de place :

```
rm -rf /tools
```

Enfin, supprimez le compte utilisateur temporaire « tester » créé au début du chapitre précédent.

```
userdel -r tester
```

Chapitre 9. Configuration du système

9.1. Introduction

Ce chapitre parle des fichiers de configuration et des services systemd. Tout d'abord, les fichiers de configuration générale nécessaire à la mise en place du réseau sont présentés.

- Section 9.2, « Configuration générale du réseau. »
- Section 9.2.3, « Configurer le nom d'hôte du système. »
- Section 9.2.4, « Personnaliser le fichier /etc/hosts. »

Ensuite, Les problèmes qui affectent la bonne configuration des périphériques sont décrits.

- Section 9.3, « Manipulation des périphériques et modules. »
- Section 9.4, « Gérer les périphériques. »

Puis, la configuration de l'horloge du système et de l'agencement du clavier est expliquée.

- Section 9.5, « Configuration de l'horloge système. »
- Section 9.6, « Configurer la Console Linux. »

De plus, une brève introduction aux scripts et aux fichiers de configuration utilisés quand l'utilisateur se connecte au système est faite.

- Section 9.7, « Configuration des paramètres régionaux du système. »
- Section 9.8, « Créer le fichier /etc/inputrc. »

Et enfin, la configuration du comportement de systemd est expliquée.

- Section 9.10, « Utilisation et configuration de Systemd. »

9.2. Configuration générale du réseau

Cette section s'applique seulement si une carte réseau doit être configurée.

9.2.1. Fichier de configuration d'interface réseau

À partir de la version 209, systemd contient un démon de configuration réseau nommé **systemd-networkd** qui peut être utilisé pour la configuration basique du réseau. De plus, depuis la version 213, la résolution de nom DNS peut être prise en charge par **systemd-resolved** au lieu d'un fichier `/etc/resolv.conf` statique. Ces deux services sont activés par défaut.

Les fichiers de configuration pour **systemd-networkd** (et **systemd-resolved**) peuvent être placés dans `/usr/lib/systemd/network` ou `/etc/systemd/network`. Les fichiers dans `/etc/systemd/network` ont une priorité supérieure à ceux dans `/usr/lib/systemd/network`. Il existe trois types de fichiers de configuration : `.link`, `.netdev` et `.network`. Pour une description détaillée et des exemples de contenu de ces fichiers de configuration, consultez des pages de manuel `systemd-link(5)`, `systemd-netdev(5)` et `systemd-network(5)`.

9.2.1.1. Nommage des périphériques réseau

Udev assigne normalement des noms d'interface basés sur les caractéristiques physiques comme enp2s1. Si vous n'êtes pas sûr de votre nom d'interface, vous pouvez toujours lancer **ip link** après avoir lancé votre système.

Pour la plupart des systèmes, il n'y a qu'une interface réseau pour chaque type de connexion. Par exemple, le nom classique pour une connexion filaire est eth0. Une connexion sans fil aura souvant pour nom wifi0 ou wlan0.

Si vous préférez utiliser les noms d'interfaces réseau classiques ou personnalisés, il y a trois manières de procéder :

- Masquez le fichier .link d'udev pour la politique par défaut :

```
ln -s /dev/null /etc/systemd/network/99-default.link
```

- Créez une convention de nommage manuelle, par exemple en nommant les interface comme « internet0 », « dmz0 » ou « lan0 ». Pour cela, créez des fichiers .link dans /etc/systemd/network/, qui choisissent un nom explicite ou une meilleure convention de nommage pour vos interfaces. Par exemple :

```
cat > /etc/systemd/network/10-ether0.link << "EOF"
[Match]
# Changez l'adresse MAC comme il faut pour votre périphérique réseau
MACAddress=12:34:45:78:90:AB

[Link]
Name=ether0
EOF
```

  Voir la page de manuel de systemd.link(5) pour plus d'informations.

- Dans /boot/grub/grub.cfg, passez l'option net.ifnames=0 sur la ligne de commande du noyau.

9.2.1.2. Configuration en IP statique

La commande suivante crée un fichier de configuration basique pour la configuration d'une adresse ip statique (en utilisant systemd-networkd et systemd-resolved) :

```
cat > /etc/systemd/network/10-eth-static.network << "EOF"
[Match]
Name=<network-device-name>

[Network]
Address=192.168.0.2/24
Gateway=192.168.0.1
DNS=192.168.0.1
Domains=<Your Domain Name>
EOF
```

Vous pouvez ajouter plusieurs entrées DNS si vous avez plus d’un serveur DNS. Ne rajoutez pas d’entrée DNS ou Domains si vous voulez utiliser un fichier `/etc/resolv.conf` statique.

9.2.1.3. Configuration du DHCP

La commande suivante crée un fichier de configuration basique pour la configuration de DHCP en IPv4 :

```
cat > /etc/systemd/network/10-eth-dhcp.network << "EOF"
[Match]
Name=<network-device-name>

[Network]
DHCP=ipv4

[DHCP]
UseDomains=true
EOF
```

9.2.2. Créer le fichier /etc/resolv.conf

Si le système a besoin d'être connecté à Internet, il aura besoin d'un DNS pour résoudre les noms de domaines Internet en adresse IP, et vice-versa. Ceci se fait en plaçant les adresses IP des serveurs DNS, disponibles auprès du FAI ou de l'administrateur système, dans `/etc/resolv.conf`.

9.2.2.1. Configuration de systemd-resolved

Note

Si vous utilisez un autre moyen pour configurer vos interfaces réseaux (par exemple ppp, network-manager, etc), ou si vous utilisez un résolveur local (par exemple bind, dnsmasq, unbound, etc), ou tout autre logiciel générant un fichier `/etc/resolv.conf` (par exemple resolvconf), le service **systemd-resolved** ne devrait pas être utilisé.

Lorsque vous utilisez **systemd-resolved** pour configurer le DNS, il crée le fichier `/run/systemd/resolve/resolv.conf`. Créez un lien symbolique dans `/etc` pour utiliser le fichier généré :

```
ln -sfv /run/systemd/resolve/resolv.conf /etc/resolv.conf
```

9.2.2.2. Configuration de resolv.conf statique

Si un fichier statique `/etc/resolv.conf` est désiré, créez-le en lançant la commande suivante :

```
cat > /etc/resolv.conf << "EOF"
# Début de /etc/resolv.conf

domain <Votre nom de domaine>
nameserver <Adresse IP du DNS primaire>
nameserver <Adresse IP du DNS secondaire>

# Fin de /etc/resolv.conf
EOF
```

Le paramètre `domain` peut être omis ou remplacé par un paramètre `search`. Voir la page de manuel de resolv.conf pour plus de détails.

Remplacez *<Adresse IP du DNS>* par l'adresse IP du DNS le plus approprié pour votre configuration. Il y aura souvent plus d'une entrée (les serveurs secondaires sont utiles en cas d'indisponibilité du premier). Si vous avez seulement besoin ou si vous voulez seulement un serveur DNS, supprimez la seconde ligne *nameserver* du fichier. L'adresse IP pourrait aussi être un routeur sur le réseau local. Une autre possibilité est d'utiliser le service DNS public de Google avec les adresse IP suivantes comme serveurs de noms.

Note

Les adresses des DNS publiques de Google sont *8.8.8.8* et *8.8.4.4* en IPv4 et *2001:4860:4860::8888* et *2001:4860:4860::8844* en IPv6. Autrement, vous pouvez aussi utiliser les résolveurs de FDN: *80.67.169.12* et *80.67.169.40* en IPv4 et *2001:910:800::12* et *2001:910:800::40* en IPv6.

9.2.3. Configurer le nom d'hôte du système

Pendant le processus de démarrage, le fichier `/etc/hostname` est utilisé pour donner un nom d'hôte au système.

Créez le fichier /etc/network et saisissez le nom du système en lançant :

```
echo "<lfs>" > /etc/hostname
```

<lfs> doit être remplacé par le nom de l'ordinateur. Ne saisissez pas le FQDN ici. Cette information sera saisie dans le fichier /etc/hosts.

9.2.4. Personnaliser le fichier /etc/hosts

Choisissez un nom de domaine pleinement qualifié (*fully-qualified domain name*, ou FQDN) et les alias possibles à déclarer dans le fichier /etc/hosts. Si vous utilisez des adresses statiques, vous devrez aussi choisir une adresse IP. La syntaxe d'une entrée du fichier hosts est :

```
IP_address myhost.example.org aliases
```

Sauf si votre ordinateur doit être visible à partir d'Internet (c-à-d que c'est un domaine enregistré et un bloc d'adresses IP valide—la plupart des utilisateurs n'ont pas ceci), assurez-vous que l'adresse IP se trouve dans la plage d'adresses réservée aux réseaux privés. Les plages valides sont :

```
Plage d'adresses réseau privés      Préfixe normal
     10.0.0.1 - 10.255.255.254                8
    172.x.0.1 - 172.x.255.254                16
  192.168.y.1 - 192.168.y.254                24
```

x peut être un nombre compris entre 16 et 31. y peut être un nombre compris entre 0 et 255.

Une adresse IP privée valide pourrait être 192.168.1.1. Un FQDN valide pour cette IP pourrait être lfs.example.org.

Même si vous ne possédez pas de carte réseau, un FQDN valide est toujours requis. Certains programmes, comme les MTA, en ont besoin pour fonctionner correctement.

Créez le fichier /etc/hosts avec la commande suivante :

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.0.2> <FQDN> <HOSTNAME> [alias1] [alias2] ...
::1       localhost ip6-localhost ip6-loopback
ff02::1   ip6-allnodes
ff02::2   ip6-allrouters

# End /etc/hosts
EOF
```

Les valeurs <192.168.0.2>, <FQDN> et <HOSTNAME> doivent être remplacées suivant les contraintes et les besoins spécifiques (si la machine se voit affecter une adresse IP par un administrateur réseau/système et que cette machine est connectée à un réseau existant). Vous pouvez omettre le ou les noms d'alias facultatifs, et la ligne <192.168.0.2 si vous utilisez une connexion configurée avec DHCP ou l'autoconfiguration IPv6.

L'adresse ::1 est l'équivalent en IPv6 à 127.0.0.1 et représente l'interface de rebouclage IPv6. 127.0.1.1 est une entrée de rebouclage réservée spécifiquement pour le FQDN.

9.3. Manipulation des périphériques et modules

Au Chapitre 8, nous avons installé le paquet udev en construisant systemd. Avant d'entrer dans les détails concernant son fonctionnement, un bref historique des méthodes précédentes de gestion des périphériques est nécessaire.

Traditionnellement, les systèmes Linux utilisaient une méthode de création de périphériques statiques avec laquelle un grand nombre de nœuds de périphériques étaient créés sous `/dev` (quelques fois littéralement des milliers de nœuds), que le matériel correspondant existait ou non. Ceci était fait le plus souvent avec un script **MAKEDEV**, qui contient des appels au programme **mknod** avec les numéros de périphériques majeurs et mineurs pour chaque périphérique possible qui pourrait exister dans le monde.

En utilisant la méthode udev, seuls les nœuds des périphériques détectés par le noyau sont créés. Comme ces nœuds de périphériques seront créés à chaque lancement du système, ils seront stockés dans un système de fichiers `devtmpfs` (un système de fichiers virtuel qui réside entièrement dans la mémoire du système). Les nœuds de périphériques ne requièrent pas beaucoup d'espace, donc la mémoire utilisée est négligeable.

9.3.1. Historique

En février 2000, un nouveau système de fichiers appelé `devfs` a été intégré au noyau 2.3.46 et rendu disponible pour la série 2.4 des noyaux stables. Bien qu'il soit présent dans les sources du noyau, cette méthode de création dynamique des périphériques n'a jamais reçu un support inconditionnel des développeurs du noyau.

Le principal problème de l'approche adopté par `devfs` était la façon dont il gérait la détection, la création et le nommage des périphériques. Ce dernier problème, le nommage des périphériques, était peut-être le plus critique. Il est généralement accepté que s'il est possible de configurer les noms des périphériques, alors la politique de nommage des périphériques revient à l'administrateur du système, et non imposée par quelque développeur. Le système de fichiers `devfs` souffrait aussi de restrictions particulières inhérentes à sa conception et qui ne pouvaient être corrigées sans une revue importante du noyau. Il a aussi été marqué comme obsolète pendant une longue période — à cause d'un manque de maintenance — et a finalement été supprimé du noyau en juin 2006.

Avec le développement de la branche instable 2.5 du noyau, sortie ensuite avec la série 2.6 des noyaux stables, un nouveau système de fichiers virtuel appelé `sysfs` est arrivé. Le rôle de `sysfs` est d'exporter une vue de la configuration matérielle du système pour les processus en espace utilisateur. Avec cette représentation visible en espace utilisateur, la possibilité de développer un remplacement en espace utilisateur de `devfs` est devenu beaucoup plus réaliste.

9.3.2. Implémentation d'Udev

9.3.2.1. Sysfs

Le système de fichier `sysfs` a été brièvement mentionné ci-dessus. On pourrait se demander comment `sysfs` connaît les périphériques présents sur un système et quels numéros de périphériques devraient être utilisés. Les pilotes qui ont été compilés directement dans le noyau enregistrent leurs objets avec le `sysfs` (en interne, devtmpfs) quand ils sont détectés par le noyau. Pour les pilotes compilés en tant que modules, cet enregistrement surviendra quand le module sera chargé. Une fois que le système de fichier `sysfs` est monté (sur /sys), les données enregistrées par les pilotes avec `sysfs` sont disponibles pour les processus en espace utilisateur ainsi que pour udevd pour traitement (y compris des modifications aux nœuds de périphériques).

9.3.2.2. Création de nœuds de périphérique

Les fichiers de périphérique sont créés par le noyau avec le système de fichiers `devtmpfs`. Tout pilote souhaitant enregistrer un nœud de périphérique ira dans le `devtmpfs` (par le cœur du pilote) pour le faire. Quand une instance `devtmpfs` est montée sur `/dev`, le nœud de périphérique sera créé dès le départ avec un nom, des droits et un propriétaire figés.

Peu de temps après, le noyau enverra un uevent à **udevd**. À partir des règles indiquées dans les fichiers contenus dans les répertoires `/etc/udev/rules.d`, `/lib/udev/rules.d` et `/run/udev/rules.d`, **udevd** créera les liens symboliques supplémentaires vers le nœud de périphérique, ou bien il modifiera ses droits, son propriétaire ou son groupe, ou l'entrée dans la base de données interne d'**udevd** concernant cet objet.

Les règles de ces trois répertoires sont numérotées et les trois répertoires sont fusionnés. Si **udevd** ne peut pas trouver de règles pour le périphérique qu'il crée, il en donnera la propriété et les droits à n'importe quel `devtmpfs` utilisé au départ.

9.3.2.3. Chargement d'un module

Il se peut que les pilotes des périphériques compilés en module aient aussi des alias compilés. Les alias sont visibles dans la sortie du programme **modinfo** et sont souvent liés aux identifiants de bus spécifiques des périphériques pris en charge par un module. Par exemple, le pilote *snd-fm801* prend en charge les périphériques PCI ayant l'ID fabricant 0x1319 et l'ID de périphérique 0x0801 a aussi un alias « pci:v00001319d00000801sv*sd*bc04sc01i* ». Pour la plupart des périphériques, le pilote du bus définit l'alias du pilote qui gérerait le périphérique via `sysfs`. Par exemple, le fichier `/sys/bus/pci/devices/0000:00:0d.0/modalias` pourrait contenir la chaîne « pci:v00001319d00000801sv00001319sd00001319bc04sc01i00 ». Les règles par défaut fournies par Udev feront que **udevd** appellera **/sbin/modprobe** avec le contenu de la variable d'environnement de l'uevent `MODALIAS` (qui devrait être la même que le contenu du fichier `modalias` dans sysfs), donc chargera tous les modules dont les alias correspondent à cette chaîne après les expansions génériques.

Dans cet exemple, cela signifie que, outre *snd-fm801*, le pilote obsolète (et non désiré) *forte* sera chargé s'il est disponible. Voir ci-dessous les moyens d'empêcher le chargement des modules indésirables.

Le noyau lui-même est aussi capable de charger des modules de protocole réseau, de prise en charge pour des systèmes de fichiers et de prise en charge native des langues sur demande.

9.3.2.4. Gestion des périphériques dynamiques ou montables à chaud

Quand vous connectez un périphérique, comme un lecteur MP3 USB, le noyau reconnaît que le périphérique est maintenant connecté et génère un uevent. Cet uevent est alors géré par **udevd** comme décrit ci-dessus.

9.3.3. Problèmes avec le chargement des modules et la création des périphériques

Il existe quelques problèmes connus pour la création automatique des nœuds de périphériques.

9.3.3.1. Un module noyau n'est pas chargé automatiquement

Udev ne chargera un module que s'il a un alias spécifique au bus et que le pilote du bus envoie correctement les alias nécessaires vers `sysfs`. Sinon, il faut organiser le chargement des modules par d'autres moyens. Avec Linux-5.10.17, udev est connu pour charger les pilotes correctement écrits pour les périphériques INPUT, IDE, PCI, USB, SCSI, SERIO et FireWire.

Pour déterminer si le pilote du périphérique dont vous avez besoin prend en charge udev, lancez **modinfo** avec le nom du module en argument. Puis, essayez de localiser le répertoire du périphérique sous `/sys/bus` et vérifiez s'il y a un fichier `modalias`.

Si le fichier `modalias` existe dans `sysfs`, alors le pilote prend en charge le périphérique et peut lui parler directement, mais s'il n'a pas d'alias, c'est un bogue dans le pilote. Chargez le pilote sans l'aide d'udev et attendez que le problème soit corrigé plus tard.

S'il n'y a pas de fichier `modalias` dans le bon répertoire sous `/sys/bus`, cela signifie que les développeurs du noyau n'ont pas encore ajouté de prise en charge de modalias à ce type de bus. Avec Linux-5.10.17, c'est le cas pour les bus ISA. Attendez que ce problème soit réparé dans les versions ultérieures du noyau.

Udev n'a pas du tout pour but de charger des pilotes « wrapper » (qui emballent un autre pilote) comme *snd-pcm-oss* et des pilotes non matériels comme *loop*.

9.3.3.2. Un module du noyau n'est pas chargé automatiquement et udev n'est pas prévu pour le charger

Si le module « enveloppe » n'améliore que la fonctionnalité fournie par un autre module (comme *snd-pcm-oss* améliore la fonctionnalité de *snd-pcm* en rendant les cartes son disponibles pour les applications OSS), configurez **modprobe** pour charger l'enveloppe après qu'udev a chargé le module enveloppé. Pour cela, ajoutez une ligne « softdep » dans tous les fichiers `/etc/modprobe.d/<filename>.conf`. Par exemple :

```
softdep snd-pcm post: snd-pcm-oss
```

Remarquez que la commande « softdep » autorise aussi les dépendances `pre:`, ou un mélange de `pre:` et de `post:`. Voir la page de manuel de `modprobe.d(5)` pour plus d'informations sur la syntaxe et les possibilités de « softdep ».

9.3.3.3. Udev charge un module indésirable

Ne compilez pas le module, ou mettez-le en liste noire dans un fichier `/etc/modprobe.d/blacklist.conf` comme nous l'avons fait avec le module *forte* dans l'exemple ci-dessous :

```
blacklist forte
```

Les modules en liste noire peuvent toujours être chargés manuellement avec la commande explicite **modprobe**.

9.3.3.4. Udev crée mal un périphérique, ou crée un mauvais lien symbolique

Cela se produit habituellement si une règle correspond à un périphérique de façon imprévue. Par exemple, une règle lacunaire peut correspondre à un disque SCSI (comme désiré) et au périphérique SCSI générique du même fabricant (de façon incorrecte). Trouvez la règle défectueuse et affinez-la, à l'aide de la commande **udevadm info**.

9.3.3.5. Une règle Udev fonctionne de manière non fiable

Cela peut être une autre manifestation du problème précédent. Sinon, et si votre règle utilise les attributs de `sysfs`, il se peut que ce soit un problème de timing du noyau, sur le point d'être corrigé dans les noyaux ultérieurs. Pour le moment, vous pouvez contourner en créant une règle qui attend l'attribut `sysfs` utilisé et en le mettant dans le fichier `/etc/udev/rules.d/10-wait_for_sysfs.rules` (créez ce fichier s'il n'existe pas). Merci d'informer la liste de développement de LFS si vous faites ainsi et que cela vous aide.

9.3.3.6. Udev ne crée pas un périphérique

Les textes suivants supposent que le pilote est compilé statiquement dans le noyau ou bien sont déjà chargés comme modules et que vous avez déjà vérifié que udev ne crée pas un périphérique mal nommé.

Udev n'a pas les informations pour créer un nœud si un pilote noyau n'exporte pas ses informations vers `sysfs`. C'est le plus souvent le cas des pilotes tiers ne provenant pas du noyau. Créez un nœud de périphérique statique dans `/lib/udev/devices` avec les numéros majeurs/mineurs appropriés (regardez le fichier `devices.txt` dans la documentation du noyau du vendeur du pilote tiers). Le nœud statique sera copié dans `/dev` par **udev**.

9.3.3.7. L'ordre de nommage des périphériques change de manière aléatoire après le redémarrage

Cela est dû au fait qu'udev, par nature, gère les uevents et charge les modules en parallèle, donc dans un ordre imprévisible. Cela ne sera jamais « corrigé ». Vous ne devriez pas supposer que les noms des périphériques du noyau sont stables. Créez plutôt vos propres règles qui rendent les liens symboliques stables basés sur des attributs stables du périphérique, comme une série de nombres ou la sortie de divers utilitaires *_id installés par udev. Voir la Section 9.4, « Gérer les périphériques » et la Section 9.2, « Configuration générale du réseau » pour des exemples.

9.3.4. Lecture utile

Des documentations supplémentaires sont disponibles sur les sites suivants :

- A Userspace Implementation of `devfs` *http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf* (NdT : Une implémentation en espace utilisateur de devfs)
- The `sysfs` Filesystem *http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf* (NdT : Le système de fichiers `sysfs`)

9.4. Gérer les périphériques

9.4.1. Gestion des périphériques dupliqués

Comme expliqué à la Section 9.3, « Manipulation des périphériques et modules », l'ordre dans lequel les périphériques ayant la même fonction apparaissent dans `/dev` est essentiellement aléatoire. Par exemple si vous avez une webcam USB et un tunner TV, parfois `/dev/video0` renvoie à la webcam, et `/dev/video1` renvoie au tuner, et parfois après un redémarrage l'ordre s'inverse. Pour toutes les classes de matériel sauf les cartes son et les cartes réseau, ceci peut se corriger en créant des règles udev pour des liens symboliques constants personnalisés. Le cas des cartes réseau est couvert de façon séparée dans Section 9.2, « Configuration générale du réseau », et vous pouvez trouver la configuration des cartes son dans *BLFS*.

Pour chacun des périphériques susceptibles d'avoir ce problème (même si le problème n'apparaît pas dans votre distribution Linux actuelle), trouvez le répertoire correspondant sous `/sys/class` ou `/sys/block`. Pour les périphériques vidéo, cela peut être `/sys/class/video4linux/videoX`. Calculez les attributs qui identifient de façon unique un périphérique (normalement basé sur l'ID du fabricant et du produit ou les numéros de série) :

```
udevadm info -a -p /sys/class/video4linux/video0
```

Puis, écrivez des règles qui créent les liens symboliques, comme :

```
cat > /etc/udev/rules.d/83-duplicate_devs.rules << "EOF"

# Persistent symlinks for webcam and tuner
KERNEL=="video*", ATTRS{idProduct}=="1910", ATTRS{idVendor}=="0d81", SYMLINK+="w
KERNEL=="video*", ATTRS{device}=="0x036f",  ATTRS{vendor}=="0x109e", SYMLINK+="t

EOF
```

Il en résulte que les périphériques `/dev/video0` et `/dev/video1` renvoient encore de manière aléatoire au tuner et à la webcam (et donc ne devrait jamais être utilisé directement), mais il y a des liens symboliques `/dev/tvtuner` et `/dev/webcam` qui pointent toujours vers le bon périphérique.

9.5. Configuration de l'horloge système

Cette section parle de la manière de configurer le service système **systemd-timedated**, qui configure l'horloge et le fuseau horaire système.

Si vous ne vous souvenez pas si l'horloge matérielle est configurée en UTC ou non, trouvez le en lançant la commande **`hwclock --localtime --show`**. Cela va afficher l'heure actuelle d'après l'horloge matérielle. Si l'heure correspond à celle de votre montre, alors votre horloge système est configurée à l'heure locale. Si la sortie de **hwclock** n'est pas l'heure locale, il y a des chances qu'elle soit configurée en UTC. Vérifiez cela en ajoutant ou retirant le bon nombre d'heures par rapport à l'heure affichée par **hwclock**. Par exemple si vous êtes dans le fuseau horaire MST, qui est aussi connu comme GMT -7, ajoutez 7 heures à l'heure locale.

systemd-timedated lit `/etc/adjtime`, et en fonction de ce qui est écrit dans le fichier, met l'horloge en UTC ou en heure locale.

Créez le fichier `/etc/adjtime` avec le contenu suivant si votre horloge matérielle est configurée en heure locale :

```
cat > /etc/adjtime << "EOF"
0.0 0 0.0
0
LOCAL
EOF
```

Si le fichier `/etc/adjtime` n'est pas présent au premier démarrage, **systemd-timedated** va considérer que l'horloge matérielle est configurée en UTC et ajuster le fichier en accord avec cela.

Vous pouvez également utiliser l'utilitaire **timedatectl** pour signaler à **systemd-timedated** si votre horloge matérielle est initialisée en UTC ou en temps local :

```
timedatectl set-local-rtc 1
```

timedatectl peut également être utilisé pour changer l'heure système et le fuseau horaire.

Pour changer l'heure actuelle du système, tapez :

```
timedatectl set-time YYYY-MM-DD HH:MM:SS
```

L'horloge matérielle sera également mise à jour en conséquence.

Pour changer de fuseau horaire, tapez:

```
timedatectl set-timezone TIMEZONE
```

Vous pouvez avoir une liste des fuseaux horaires disponibles en tapant :

```
timedatectl list-timezones
```

Note

Merci de noter que la commande **timedatectl** peut être utilisée seulement sur un système démarré avec systemd.

9.5.1. Synchronisation réseau du temps

Depuis la version 213, systemd livre un démon appelé **systemd-timesyncd** qui peut être utilisé pour synchroniser l'heure système avec les serveurs NTP distants.

Le démon n'est pas prévu pour remplacer le démon bien établit NTP, mais comme client du protocole SNTP qui peut être utilisé pour les tâches moins avancées et sur les systèmes à ressources limitées.

Depuis la version 216, le démon **systemd-timesyncd** est activé par défaut. Si vous voulez le désactiver, tapez la commande suivante :

```
systemctl disable systemd-timesyncd
```

Le fichier `/etc/systemd/timesyncd.conf` peut être utilisé pour changer les serveurs NTP avec lesquels **systemd-timesyncd** se synchronisent

Merci de noter que quand l'horloge système est initialisée en temps local, **systemd-timesyncd** ne mettra pas à jour l'horloge système.

9.6. Configurer la Console Linux

Cette section traite de la manière de configurer le service système **systemd-vconsole-setup** qui configure la police de la console virtuelle et la disposition du clavier dans la console.

Le service **systemd-vconsole-setup** lit les informations de configuration dans le fichier `/etc/vconsole.conf`. Décidez de la disposition du clavier et de la police à l'écran qui seront utilisées. Il existe divers guides pratiques spécifiques à chaque langue qui peuvent vous aider, voir *http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*. Observez la sortie de **localectl list-keymaps** pour une liste des dispositions de clavier valides sur la console. Regardez le répertoire `/usr/share/consolefonts` pour une liste des polices d'écran valides.

Le fichier `/etc/vconsole.conf` devrait contenir des lignes de la forme : VARIABLE="valeur". Les variables suivantes sont reconnues :

KEYMAP
: Cette variable spécifie la table de disposition du clavier. Si vous ne la définissez pas, elle est par défaut `us`.

KEYMAP_TOGGLE
: Vous pouvez utiliser cette variable pour configurer une disposition alternative par défaut où vous pouvez basculer.

FONT
: Cette variable spécifie la police utilisée par la console virtuelle.

FONT_MAP
: Cette variable spécifie la présentation de la console.

FONT_UNIMAP
: Cette variable spécifie la police de l'Unicode.

Voici un exemple pour un clavier allemand :

```
cat > /etc/vconsole.conf << "EOF"
KEYMAP=de-latin1
FONT=Lat2-Terminus16
EOF
```

Vous pouvez modifier la valeur de KEYMAP lors de l'exécution en utilisant l'outil **localectl** :

```
localectl set-keymap MAP
```

Note

Veuillez remarquer que la commande **localectl** n'est utilisable que sur un système démarré avec systemd.

Vous pouvez aussi utiliser l'utilitaire **localectl** avec les paramètres correspondants pour changer la disposition, le modèle, la variante et d'autres options du clavier :

```
localectl set-x11-keymap LAYOUT [MODEL] [VARIANT] [OPTIONS]
```

Pour lister les différentes valeurs possibles pour les paramètres **localectl set-x11-keymap**, lancez **localectl** avec les paramètres ci-dessous :

list-x11-keymap-models
: Affiche les modèles connus de claviers X11.

list-x11-keymap-layouts
: Affiche les dispositions connues de claviers X11.

list-x11-keymap-variants
: Affiche les variantes connues de claviers X11.

list-x11-keymap-options
: Affiche les options connues de claviers X11.

Note

L'utilisation des paramètres listés ci-dessus nécessite le paquet XKeyboard-Config de BLFS.

9.7. Configuration des paramètres régionaux du système

Le fichier `/etc/locale.conf` ci-dessous défini certaines variables d'environnement nécessaires pour la prise en charge native des langues. Les configurer a pour résultat :

- La sortie des programmes est traduite dans votre langue maternelle
- Le classement correct des caractères en lettres, chiffres et autres classes. Cela est nécessaire pour **bash** pour accepter correctement les caractères non-ASCII pour les paramètres linguistiques non anglais
- Le classement alphabétique correct pour les autres pays
- La taille de papier par défaut appropriée
- Le formatage correct de la monnaie, du temps et de la date

Remplacez *<ll>* ci-dessous par les deux lettres de la langue désirée (par exemple « fr ») et remplacez *<CC>* par les deux lettres du pays approprié (par exemple « FR »). *<charmap>* doit être remplacé par la table de caractères classique pour le paramètre linguistique choisi. On peut aussi ajouter des modificateurs facultatifs comme « @euro ».

La liste des paramètre régionaux pris en charge par Glibc peut être obtenue en lançant la commande suivante :

```
locale -a
```

Les tables de caractères peuvent aussi avoir certains alias, par exemple « ISO-8859-1 » est aussi référencé comme « iso8859-1 » et « iso88591 ». Certaines applications ne peuvent pas gérer les différents synonymes (il est par exemple requis que « UTF-8 » soit écrit « UTF-8 » et pas « utf8 »), il est donc plus sur de choisir le nom classique pour un paramètre linguistique particulier. Pour déterminer le nom classique, lancez la commande suivante, où *<nom du paramètre linguistique>* est la sortie donnée par **locale -a** pour votre langue préférée (« fr_FR.iso88591 » dans notre exemple).

```
LC_ALL=<nom du paramètre linguistique> locale charmap
```

Pour le paramètre « fr_FR.iso885915 », la commande va renvoyer :

```
ISO-8859-15
```

Cela résulte en un paramètre de régionalisation final de « fr_FR.ISO-8859-15 ». Il est important que le paramètre trouvé en utilisant la syntaxe précédente soit testée avant d'être ajoutée aux fichiers de démarrage de bash :

```
LC_ALL=<nom du paramètre linguistique> locale language
LC_ALL=<nom du paramètre linguistique> locale charmap
LC_ALL=<nom du paramètre linguistique> locale int_curr_symbol
LC_ALL=<nom du paramètre linguistique> locale int_prefix
```

Les commandes ci-dessus devraient imprimer les langages, l'encodage des caractères utilisé par le paramètre, la monnaie locale ainsi que le préfixe de téléphone à composer afin d'entrer dans le pays. Si une de ces commandes précédentes échoue, cela signifie que votre paramètre n'a pas été installé dans le chapitre 6 ou n'est pas prise en charge par l'installation par défaut de Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Si cela arrive, vous devez soit installer le paramètre régional désiré en utilisant la commande **localedef**, ou en utilisant un autre paramètre. Les instructions suivantes supposent qu'il n'y a pas de tels message d'erreur de Glibc.

Certains paquets au-delà de LFS peuvent ne pas prendre en charge le paramètre linguistique que vous avez choisi. Un exemple est la bibliothèque X (partie du système X Window), qui renvoie les messages suivant si le paramètre ou que le nom de jeu de caractères ne correspond pas exactement à ses fichiers internes :

```
Warning: locale not supported by Xlib, locale set to C
```

Dans plusieurs cas Xlib s'attend à ce que le jeu de caractère soit inscrit en notation majuscule avec les tirets classiques. Par exemple « ISO-8859-1 » plutôt que « iso88591 ». Il est aussi possible de trouver la spécification appropriée en retirant la partie correspondant au jeu de caractères de la spécification du paramètre régional. Cela peut être testé en lançant la commande **locale charmap** des deux paramètres. Par exemple, il faudra changer « de_DE.ISO-8859-15@euro » en « de_DE@euro » pour que ce paramètre soit reconnu par Xlib.

D'autres programmes ne vont aussi pas fonctionner correctement (mais ne vont pas nécessairement afficher un message d'erreur) si le nom du paramètre régional ne rencontre pas leurs attentes. Dans d'autres cas, rechercher comment d'autres distributions Linux prennent en charge votre paramètre régional peut fournir des informations utiles.

Une fois que les bonnes configurations de paramètres linguistiques ont été effectuées, créez le fichier `/etc/locale.conf` :

```
cat > /etc/locale.conf << "EOF"
LANG=<ll>_<CC>.<charmap><@modifiers>
EOF
```

Notez que vous devez modifier `/etc/locale.conf` avec l'utilitaire systemd **localectl**. Pour utiliser **localectl** avec l'exemple précédent, lancez :

```
localectl set-locale LANG="<ll>_<CC>.<charmap><@modifiers>"
```

Vous pouvez aussi spécifier d'autres variables d'environnement comme `LANG`, `LC_CTYPE`, `LC_NUMERIC` ou n'importe quel autre varable de la sortie de **locale**. Un exemple où `LANG` est définie à en_US.UTF-8 mais `LC_CTYPE` est défini juste à en_US est :

```
localectl set-locale LANG="en_US.UTF-8" LC_CTYPE="en_US"
```

Note

Notez que la commande **localectl** ne peut-être lancée que sur un système lancé avec systemd.

Les paramètres « C » (défaut) et « en_US » (recommandé pour les utilisateurs anglophones des États-Unis) sont différents. « C » utilise le jeu de caractères US-ASCII 7-bit et traite les octets avec le bit de point fort à 1 comme des caractères invalides. C'est pourquoi, par exemple, la commande **ls** les remplace par des points d'interrogation sous ce paramètre. En outre, les messages contenant ces caractères envoyés avec Mutt ou Pine ne sont pas conformes aux RFC (le jeu de caractères dans le mail sortant est indiqué comme « unknown 8-bit »). Vous ne pouvez donc utiliser le paramètre régional « C » que si vous êtes sûr que vous n'aurez jamais besoins de caractère 8 bits.

9.8. Créer le fichier /etc/inputrc

Le fichier `inputrc` est un fichier de configuration pour la bibliothèque readline, qui fournit des fonctions d'édition lors de la saisie de commandes dans le terminal. Elle fonctionne en traduisant l'entrée du clavier en des actions spécifiques. Readline est utilisé par bash et par la plupart des autres shells ainsi que par de nombreuses autres applications.

La plupart des personnes n'ont pas besoin de fonctionnalités personnalisées, donc la commande ci-dessous crée un fichier `/etc/inputrc` global utilisé par tous ceux qui se connectent. Si vous décidez plus tard que vous avez besoin de surcharger les valeurs par défaut utilisateur par utilisateur, vous pouvez créer un fichier `.inputrc` dans le répertoire personnel de l'utilisateur avec les correspondances modifiées.

Pour plus d'informations sur l'édition du fichier `inputrc`, voir **info bash** à la section *Fichier d'initialisation Readline* (ou *Readline Init File*). **info readline** est aussi une bonne source d'informations.

Ci-dessous se trouve un fichier `inputrc` générique avec des commentaires expliquant l'utilité des différentes options. Remarquez que les commentaires ne peuvent pas être sur la même ligne que les commandes. Créez le fichier en utilisant la commande suivante :

```
cat > /etc/inputrc << "EOF"
# Début de /etc/inputrc
# Modifié par Chris Lynn <roryo@roryo.dynup.net>

# Permettre à l'invite de commande d'aller à la ligne
set horizontal-scroll-mode Off

# Activer l'entrée sur 8 bits
set meta-flag On
set input-meta On

# Ne pas supprimer le 8ème bit
set convert-meta Off

# Conserver le 8ème bit à l'affichage
set output-meta On

# none, visible ou audible
set bell-style none

# Toutes les indications qui suivent font correspondre la séquence
# d'échappement contenue dans le 1er argument à la fonction
# spécifique de readline
"\eOd": backward-word
"\eOc": forward-word

# Pour la console linux
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# pour xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# pour Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# Fin de /etc/inputrc
EOF
```

9.9. Créaction du fichier /etc/shells

Le fichier `shells` contient une liste des shells de connexion présents sur le système. Les applications utilisent ce fichier pour déterminer si un shell est valide. Pour chaque shell, une seule ligne doit être présente, contenant l'emplacement du shell relativement à la racine (/).

Par exemple, ce fichier est consulté par **chsh** pour déterminer si un utilisateur non privilégié peut modifier le shell de connexion de son compte. Si le nom de la commande n'est pas listé, l'utilisateur n'aura pas le droit d'en changer.

C'est nécessaire pour des applications telles que GDM qui ne peuplent pas le navigateur d'interface s'il ne peut pas trouver `/etc/shells`, ou les démons FTP qui interdisent traditionnellement aux utilisateurs l'accès avec des shells qui ne sont pas inclus dans ce fichier.

```
cat > /etc/shells << "EOF"
# Begin /etc/shells

/bin/sh
/bin/bash

# End /etc/shells
EOF
```

9.10. Utilisation et configuration de Systemd

9.10.1. Configuration basique

Le fichier `/etc/systemd/system.conf` contient un ensemble d'options pour contrôler les opérations de base de systemd. Le fichier par défaut a toutes ses entrées commentées indiquant les paramètres par défaut. Ce fichier est l'endroit où le niveau de journalisation (log) peut être modifié ainsi que les paramètres de base de journalisation. Voir la page de manuel de `systemd-system.conf(5)` pour plus de détails à propos de chaque option de configuration.

9.10.2. Désactiver l'effacement de l'écran durant le démarrage

Le comportement normal de systemd est d'effacer l'écran à la fin de la séquence de démarrage. Si désiré, ce comportement peut être changé en exécutant la commande suivante :

```
mkdir -pv /etc/systemd/system/getty@tty1.service.d

cat > /etc/systemd/system/getty@tty1.service.d/noclear.conf << EOF
[Service]
TTYVTDisallocate=no
EOF
```

Les messages de démarrage peuvent toujours être examinés en utilisant la commande **`journalctl -b`** en tant qu'utilisateur `root`.

9.10.3. Désactiver tmpfs pour /tmp

Par défaut, `/tmp` est créé comme un tmpfs. Si vous ne le voulez pas, il est possible de l'en empêcher de la manière suivante :

```
ln -sfv /dev/null /etc/systemd/system/tmp.mount
```

Autrement, si vous souhaitez avoir une partition séparée pour `/tmp`, spécifiez-la dans une entrée de `/etc/fstab`.

Avertissement

Ne créez pas le lien symbolique ci-dessus si vous utilisez une partition séparée pour `/tmp`. Cela empêche la partition racine (/) d'être remontée en lecture-écriture et rend le système inutilisable au démarrage.

9.10.4. Configurer la création et la suppression automatique de fichiers

Il existe de nombreux services pour créer ou supprimer des fichiers ou des dossiers :

- systemd-tmpfiles-clean.service
- systemd-tmpfiles-setup-dev.service
- systemd-tmpfiles-setup.service

L'emplacement système des fichiers de configuration est `/usr/lib/tmpfiles.d/*.conf`. Les fichiers locaux de configuration sont dans `/etc/tmpfiles.d`. Les fichiers dans `/etc/tmpfiles.d` prévallent sur les fichiers du même nom dans `/usr/lib/tmpfiles.d`. Voir la page de manuel `tmpfiles.d(5)` pour plus de détails sur le format de fichier.

Remarquez que la syntaxe pour les fichiers `/usr/lib/tmpfiles.d/*.conf` peut être déroutante. Par exemple, la suppression de fichiers par défaut dans le répertoire /tmp se trouve dans `/usr/lib/tmpfiles.d/tmp.conf` à cette ligne :

```
q /tmp 1777 root root 10d
```

. Le champ type, q, parle de créer un sous-volume avec des quotas, ce qui n'est vraiment possible que pour les système de fichiers btrfs. Il référence le type v qui lui-même référence le type d (répertoire). Cela crée ensuite le répertoire spécifié s'il n'est pas présent et ajuste ensuite les permissions et la propriété comme indiqué. Le contenu du répertoire sera sujet au nettoyage basé sur l'âge si l'argument d'âge est spécifié.

Si les paramètres par défaut ne sont pas désirés, alors vous devrez copier le fichier vers `/etc/tmpfiles.d` et le modifier comme vous le voulez. Par exemple :

```
mkdir -p /etc/tmpfiles.d
cp /usr/lib/tmpfiles.d/tmp.conf /etc/tmpfiles.d
```

9.10.5. Redéfinition des comportements par défaut des services

Les paramètres d'une unité peuvent être redéfinis en créant un dossier et un fichier de configuration dans `/etc/systemd/system`. Par exemple :

```
mkdir -pv /etc/systemd/system/foobar.service.d

cat > /etc/systemd/system/foobar.service.d/foobar.conf << EOF
[Service]
Restart=always
RestartSec=30
EOF
```

Voir la page de manuel `systemd.unit(5)` pour plus d'informations. Après la création du fichier de configuration, exécutez **`systemctl daemon-reload`** et **`systemctl restart foobar`** pour activer les changements à un service.

9.10.6. Débogage de la séquence de démarrage

Plutôt que des scripts shell utilisés par les systèmes d'init du style de SysVinit ou BSD, systemd utilise un format unifié pour différents types de fichiers de démarrage (ou unités). La commande **systemctl** est utilisée pour activer, désactiver, contrôler l'état et obtenir le statut d'un fichier d'unité. Voici quelques exemples de commandes fréquentes :

- **systemctl list-units -t *<service>* [--all]**: liste les fichiers d'unité chargés de type service.
- **systemctl list-units -t *<target>* [--all]**: liste les fichiers d'unité chargés de type target.
- **systemctl show -p Wants *<multi-user.target>*** : montre toutes les unités qui dépendent de la cible multi-user. Les cibles sont des fichiers d'unité spéciaux qui sont analogues aux niveaux d'exécution de SysVinit.
- **systemctl status *<servicename.service>***: montre le statut du service servicename. L'extension .service peut être omise s'il n'y a pas d'autres fichiers d'unité portant le même nom, comme des fichiers .socket (qui créent un socket en écoute qui fourni les même fonctionnalités qu'inetd/xinetd).

9.10.7. Utiliser le journal Systemd

La journalisation d'un système démarré avec systemd est géré par systemd-journald (par défaut), plutôt qu'un démon syslog unix typique. Vous pouvez aussi ajouter un démon syslog normal et faire travailler les deux côte à côte si vous le souhaitez. Le programme systemd-journald stocke les entrées du journal dans un format binaire plutôt que dans un fichier en texte brut. La commande **journalctl** est fournie pour aider à analyser le fichier. Voici quelques exemples de commandes usuelles :

- **journalctl -r** : montre tout le contenu du journal en sens chronologique inverse.
- **journalctl -u *UNIT*** : montre les entrées du journal associées avec le fichier UNIT spécifié.
- **journalctl -b[=ID] -r** : montre les entrées du journal depuis le dernier démarrage réussi (ou pour le démarrage d'identifiant ID) en sens chronologique inverse.
- **journalctl -f** : fournit une fonctionnalité similaire à tail -f (suivre).

9.10.8. Travailler avec les core dumps

Les « core dumps » sont utiles pour déboguer des programmes crashés, surtout quand un processus démon plante. Sur un système démarré avec systemd, les core dumps sont gérés par **systemd-coredump**. Il enregistrera le core dump das le journal et stockera le fichier core dans `/var/lib/systemd/coredump`. Pour récupérer et utiliser le fichier core, système fourni l'outil **coredumpctl**. Voici quelques exemples de commandes fréquentes :

- **coredumpctl -r** : montre tous les core dumps en ordre chronologique inverse.
- **coredumpctl -1 info** : montre les informations du dernier core dump.
- **coredumpctl -1 debug** : charge le dernier core dump dans *GDB*.

Les core dumps peuvent utiliser beaucoup d'espace disque. L'espace disque maximal utilisé par les core dumps se limite en créant un fichier de configuration dans `/etc/systemd/coredump.conf.d`. Par exemple :

```
mkdir -pv /etc/systemd/coredump.conf.d

cat > /etc/systemd/coredump.conf.d/maxuse.conf << EOF
[Coredump]
MaxUse=5G
EOF
```

Voir les pages de manuel `systemd-coredump(8)`, `coredumpctl(1)`, and `coredump.conf.d(5)` pour plus d'information.

9.10.9. Processus lancés durablement

Depuis la version systemd-230, tous les processus utilisateurs sont tués lorsque la session utilisateur se termine, même en utilisant nohup, ou que le processus utilise les fonctions `daemon()` ou `setsid()`. Ceci est un changement délibéré par rapport à un environnement historiquement plus permissif vers un environnement plus restrictif. Le nouveau comportement peut causer des problèmes si vous dépendez de programmes lancés durablement (par exemple, **screen** ou **tmux**) qui restent actifs après la fin de votre session utilisateur. Il y a trois moyens de permettre la persistance des processus après la fin d'une session utilisateur.

- *N'activez les processus persistants que pour les utilisateurs qui en ont besoin* : les utilisateurs ont la possibilité d'activer les processus persistants avec la commande **loginctl enable-linger** pour eux-mêmes. Les administrateurs systèmes peuvent utiliser la même commande avec un argument *`utilisateur`* pour les activer pour un utilisateur. Cet utilisateur peut alors utiliser la commande **systemd-run** pour débuter une tâche durable. Par exemple : **systemd-run --scope --user /usr/bin/screen**. Si vous souhaitez activer les tâches durables pour votre utilisateur, le service user@.service sera toujours présent même après la fermeture de toutes les sessions, et démarrera automatiquement au démarrage du système. Ceci a l'avantage d'autoriser et d'interdire explicitement aux programmes de s'exécuter après que la session utilisateur est fermée, mais cela casse la rétro-compatibilité avec des outils comme **nohup** et les utilitaires qui utilisent `daemon()`.
- *Activer les processus persistant sur tout le système* : vous pouvez définir *`KillUserProcesses=no`* dans `/etc/systemd/logind.conf` pour autoriser globalement la persistance pour tous les utilisateurs. Ceci a l'avantage de laisser disponibles les vieilles méthodes à tous les utilisateurs au prix de la perte de contrôle explicite.
- *Désactiver à la construction* : vous pouvez autoriser les processus persistants par défaut pendant la construction de systemd en ajoutant le paramètre *`-Ddefault-kill-user-processes=false`* à la commande **meson** de systemd. Ceci désactive complètement la capacité que systemd a de tuer les processus utilisateurs à la fin de la session.

Chapitre 10. Rendre le système LFS amorçable

10.1. Introduction

Il est temps de rendre amorçable le système LFS. Ce chapitre traite de la création d'un fichier `fstab`, de la construction d'un noyau pour le nouveau système LFS et de l'installation du chargeur de démarrage GRUB afin que le système LFS puisse être sélectionné au démarrage.

10.2. Créer le fichier /etc/fstab

Le fichier `/etc/fstab` est utilisé par quelques programmes pour déterminer les systèmes de fichiers à monter par défaut, dans quel ordre, et lesquels doivent être vérifiés (recherche d'erreurs d'intégrité) avant le montage. Créez une nouvelle table des systèmes de fichiers comme ceci :

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab

# file system  mount-point  type     options             dump  fsck
#                                                              order

/dev/<xxx>     /            <fff>    defaults            1     1
/dev/<yyy>     swap         swap     pri=1               0     0

# End /etc/fstab
EOF
```

Remplacez *<xxx>*, *<yyy>* et *<fff>* par les valeurs appropriées pour votre système, par exemple `sda2`, `sda5` et `ext4`. Pour tous les détails sur les six champs de cette table, voyez **man 5 fstab**.

Les systèmes de fichiers ayant pour origine MS-DOS ou Windows (c-à-d. vfat, ntfs, smbfs, cifs, iso9660, udf) ont besoin d'une option spéciale, utf8, pour que les caractères non-ascii dans les noms de fichiers soient interprétés correctement. Pour les paramètres linguistiques non-utf-8, la valeur de `iocharset` devrait être la même que le jeu de caractères de votre locale, ajustée de telle sorte que le noyau la comprenne. Cela fonctionne si la définition du codage adéquat (que vous trouverez sous File systems -> Native Language Support lors de la configuration du noyau) a été compilée en dur dans le noyau ou en module. Cependant, si le jeu de caractères des paramètres linguistiques est UTF-8, l'option correspondante `iocharset=utf8` rendrait le système de fichier sensible à la casse. Pour corriger cela, utilisez l'option spéciale `utf8` au lieu de `iocharset=utf8` pour les paramètres linguistiques UTF-8. L'option « codepage » est aussi nécessaire aux systèmes de fichiers vfat et smbfs. Elle devrait être paramétrée pour correspondre à la page de code utilisée sous MS-DOS dans votre pays. Par exemple, pour monter des lecteurs flash USB, un utilisateur ru_RU.KOI8-R aurait besoin de ce qui suit dans la partie des options de sa ligne de montage dans `/etc/fstab` :

```
noauto,user,quiet,showexec,codepage=866,iocharset=koi8r
```

Le fragment d'options correspondantes pour les utilisateurs ru_RU.UTF-8 est :

```
noauto,user,quiet,showexec,codepage=866,utf8
```

Remarquez que l'utilisation de `iocharset` se fait par défaut pour `iso8859-1` (ce qui laisse le système de fichiers insensible à la casse) et l'option `utf8` dit au noyau de convertir les noms de fichiers en UTF-8 pour qu'ils puissent être interprétés dans les paramètres linguistiques UTF-8.

Il est aussi possible de spécifier les valeurs de page de code et de codage entrée/sortie (iocharset) par défaut pour certains systèmes de fichiers pendant la configuration du noyau. Les paramètres pertinents sont nommés « Default NLS Option » (`CONFIG_NLS_DEFAULT`), « Default Remote NLS Option » (`CONFIG_SMB_NLS_`

DEFAULT), « Default codepage for FAT » (CONFIG_FAT_DEFAULT_CODEPAGE) et « Default iocharset for FAT » (CONFIG_FAT_DEFAULT_IOCHARSET). Il n'y a aucun moyen de spécifier ces paramètres pour les systèmes de fichiers ntfs au moment de la compilation du noyau.

Il est possible de rendre le système de fichiers ext3 résistant aux coupures de courant pour certains types de disques durs. Pour cela, ajoutez l'option de montage `barrier=1` à l'entrée appropriée dans `/etc/fstab`. Pour vérifier si le périphérique prend en charge cette option, lancez *hdparm* sur le périphérique où elle s'appliquera. Par exemple, si :

```
hdparm -I /dev/sda | grep NCQ
```

ne retourne pas une sortie non vide, l'option est prise en charge.

Remarque : Les partitions basées sur *Logical Volume Management* (LVM) ne peuvent pas utiliser l'option `barrier`.

10.3. Linux-5.10.17

Le paquet Linux contient le noyau Linux.

Temps de construction approximatif: 5.0 - 125.0 SBU (en gÃ©nÃ©ral environ 9 SBU)

Espace disque requis: 1200 - 6750 Mo (en gÃ©nÃ©ral environ 1500 Mo)

10.3.1. Installation du noyau

Construire le noyau implique un certain nombre d'étapes — configuration, compilation et installation. Pour connaître les autres méthodes que celle employée par ce livre pour configurer le noyau, lisez le fichier README contenu dans les sources du noyau.

Préparez la compilation en lançant la commande suivante :

```
make mrproper
```

Ceci nous assure que le répertoire du noyau est propre. L'équipe du noyau recommande le lancement de cette commande avant chaque compilation du noyau. Vous ne devez pas supposer que le répertoire des sources est propre juste après avoir été déballé.

Il y a plusieurs manière de configurer les options du noyau. Habituellement, à travers une interface à menus, par exemple :

```
make menuconfig
```

Voici la signification des variables d'environnement facultatives de make :

LANG=<valeur_LANG_de_l_hote> LC_ALL=

Ceci rend identique les paramétrages régionaux à ceux utilisés sur l'hôte. C'est indispensable pour que l'interface ncurses de menuconfig soit correctement dessinée sur la console texte de Linux en UTF-8.

Assurez-vous si besoin de remplacer *<valeur_LANG_de_l_hote>* par la valeur de la variable $LANG de votre hôte. Vous pouvez utiliser à la place les valeurs $LC_ALL ou $LC_CTYPE de l'hôte.

make menuconfig

Cela lance une interface à menus en ncurses. Pour d'autres interfaces (graphiques), tapez **make help**.

Pour des informations d'ordre général sur la configuration du noyau, consultez *http://www.fr.linuxfromscratch.org/view/astuces/kernel-configuration-fr.txt*. BLFS offre aussi quelques informations complémentaires concernant les besoins particuliers de configuration pour les paquets en dehors de LFS sur *http://fr.linuxfromscratch.org/blfs/../view/blfs-10.1/longindex.html#kernel-config-index*. Vous pouvez trouver des informations supplémentaires sur la configuration et la construction du noyau sur *http://www.kroah.com/lkn/*

Note

Un bon point de départ pour effectuer la configuration du noyau est de lancer **make defconfig**. Cela opérera une configuration de base de bonne qualité en prenant en compte l'architecture actuelle de votre système.

Assurez-vous d'activer, désactiver ou indiquer les fonctionnalités suivantes ou le système ne démarrera pas correctement voir pas du tout :

```
General setup -->
   [ ] Auditing Support [CONFIG_AUDIT]
   [*] Control Group support [CONFIG_CGROUPS]
   [ ] Enable deprecated sysfs features to support old userspace tools [CO
   [*] Configure standard kernel features (expert users) [CONFIG_EXPERT] -
      [*] open by fhandle syscalls [CONFIG_FHANDLE]
Processor type and features  --->
   [*] Enable seccomp to safely compute untrusted bytecode [CONFIG_SECCOMP
Firmware Drivers  --->
   [*] Export DMI identification via sysfs to userspace [CONFIG_DMIID]
Networking support  --->
  Networking options  --->
   <*> The IPv6 protocol [CONFIG_IPV6]
Device Drivers  --->
  Generic Driver Options  --->
   [ ] Support for uevent helper [CONFIG_UEVENT_HELPER]
   [*] Maintain a devtmpfs filesystem to mount at /dev [CONFIG_DEVTMPFS]
   Firmware Loader --->
      [ ] Enable the firmware sysfs fallback mechanism [CONFIG_FW_LOADER_U
File systems  --->
   [*] Inotify support for userspace [CONFIG_INOTIFY_USER]
  Pseudo filesystems  --->
   [*] Tmpfs POSIX Access Control Lists [CONFIG_TMPFS_POSIX_ACL]
```

Note

Même si "Le Protocole IPv6" n'est pas strictement nécéssaire, il est fortement recommandé par les dévelopeurs de systemd.

Note

Si votre matériel utilise UEFI, alors le « make defconfig » ci-dessus devrait automatiquement ajouter certaines options du noyau liées à EFI.

Pour permettre à votre noyau LFS de démarrer depuis l'environnement de démarrage UEFI de votre hôte, vous devez avoir sélectionné cette option dans votre noyau :

```
Processor type and features  --->
   [*]   EFI stub support  [CONFIG_EFI_STUB]
```

Une description plus avancée sur la gestion des environnements UEFI dans LFS est disponible dans l'astuce lfs-uefi.txt sur *http://fr.linuxfromscratch.org/view/astuces/lfs-uefi-fr.txt*.

Voici pourquoi on vise les éléments de configuration ci-dessus :

`Support for uevent helper`
L'activation de cette option peut interférer avec la gestion de périphériques quand on utilise Udev/Eudev.

`Maintain a devtmpfs`
Ceci créera des nœuds de périphérique automatiquement, générés par le noyau même sans Udev. Udev fonctionne alors sur cette base pour gérer les droits et l'ajout de liens symboliques. Cet élément de configuration est nécessaire pour tous les utilisateurs d'udev/eudev.

Sinon, **make oldconfig** peut être plus approprié dans certaines situations. Voir le fichier `README` pour plus d'informations.

Si vous le désirez, vous pouvez sauter la configuration du noyau en copiant le fichier de configuration, `.config`, du système hôte (en supposant qu'il est disponible) dans le répertoire `linux-5.10.17` tout juste déballé. Néanmoins, nous ne recommandons pas cette option. Il est souvent meilleur d'explorer tous les menus de configuration et de créer la configuration du noyau à partir de zéro.

Compilez l'image du noyau et les modules :

```
make
```

Si vous utilisez des modules du noyau, il peut être nécessaire de les configurer dans le fichier `/etc/modprobe.d`. Des informations au sujet de la configuration du noyau et des modules se trouvent à la Section 9.3, « Manipulation des périphériques et modules » et dans le répertoire `linux-5.10.17/Documentation` de la documentation du noyau. Enfin, `modprobe.d(5)` pourrait aussi être intéressant.

À moins d'avoir désactivé la prise en charge des modules dans la configuration du noyau, installez les modules :

```
make modules_install
```

Une fois la compilation du noyau terminée, des étapes supplémentaires sont encore nécessaires pour terminer l'installation. Certains fichiers ont besoin d'être copiés dans le répertoire `/boot`.

Attention

Si le système hôte a une partition /boot séparée, les fichiers copiés ci-dessous devraient aller là. La manière la plus simple de procéder est de lier /boot sur l'hôte (en dehors du chroot) à /mnt/lfs/boot avant de continuer. En tant qu'utilisateur root sur le *système hôte* :

```
mount --bind /boot /mnt/lfs/boot
```

Le chemin vers l'image du noyau pourrait varier suivant la plateforme utilisée. Vous pouvez changer le nom du fichier ci-dessous selon votre goût, mais la nomenclature du nom de fichier devrait ressembler à *vmlinuz* pour être compatible avec le paramétrage automatique du processus de démarrage décrit dans la section à venir. La commande suivante présuppose une architecture x86 :

```
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-5.10.17-lfs-10.1-systemd
```

`System.map` est un fichier de symboles pour le noyau. Il cartographie les points d'entrée de chaque fonction dans l'API du noyau, ainsi que les adresses de ses structures de données pendant l'exécution. Il sert de référence lors des investigations sur les problèmes de noyau. Lancez la commande suivante pour installer le fichier de symboles :

```
cp -iv System.map /boot/System.map-5.10.17
```

Le fichier de configuration du noyau `.config` produit à l'étape **make menuconfig** ci-dessus contient toutes les options de configuration choisies pour le noyau qui vient d'être compilé. Conserver ce fichier est une bonne idée pour pouvoir s'y référer plus tard :

```
cp -iv .config /boot/config-5.10.17
```

Installez la documentation du noyau Linux :

```
install -d /usr/share/doc/linux-5.10.17
cp -r Documentation/* /usr/share/doc/linux-5.10.17
```

Il est important de noter que les fichiers dans le répertoire des sources du noyau n'appartiennent pas à *root*. Chaque fois qu'un paquet est déballé par l'utilisateur *root* (comme on a fait dans chroot), les fichiers ont les ID de l'utilisateur et du groupe de l'empaqueteur sur son système hôte. En principe ce n'est pas un problème car l'arborescence des sources est supprimée après l'installation. En revanche, l'arborescence de Linux est souvent conservée longtemps. Du coup, il y a des chances que tout ce que l'ID de l'utilisateur ayant déballé le paquet a utilisé ne soit affecté à quelqu'un d'autre sur la machine. Cette personne pourrait alors avoir un droit d'écriture sur les sources du noyau.

Note

Dans de nombreux cas, la configuration du noyau aura besoin d'être mise à jour pour les paquets qui serojnt installés plutard dans BLFS. Contrairement aux autres paquets, il n'est pas nécessaire de supprimer les sources du noyau après l'installation du noyau nouvellement construit.

Si vous conservez l'arborescence des sources du noyau, lancez **chown -R 0:0** sur le répertoire `linux-5.10.17` pour vous assurer que tous les fichiers appartiennent à *root*.

Avertissement

Certaines documentations du noyau recommandent de créer un lien symbolique à partir de `/usr/src/linux` pointant vers le répertoire des sources du noyau. Ceci est spécifique aux noyaux antérieurs à la série 2.6 et *ne doit pas* être réalisé sur un système LFS car il peut poser des problèmes pour les paquets que vous souhaitez construire une fois votre système LFS de base complet.

Avertissement

Les en-têtes du répertoire système `include` (`/usr/include`) devraient *toujours* être ceux avec lesquels Glibc a été compilée, à savoir, les en-têtes nettoyés installés au Section 5.4, « Linux-5.10.17 API Headers ». Donc, ils ne devraient *jamais* être remplacés par les en-têtes du noyau brut ou par d'autres en-têtes nettoyés du noyau.

10.3.2. Configuration de l'ordre de chargement des modules Linux

La plupart du temps, les modules Linux sont chargés automatiquement, mais il faut parfois des directives supplémentaires. Le programme qui charge les modules, **modprobe** ou **insmod**, utilise `/etc/modprobe.d/usb.conf` à cette fin. Il faut créer ce fichier afin que, si les pilotes USB (ehci_hcd, ohci_hcd et uhci_hcd) ont été construits en module, ils soient chargés dans le bon ordre ; ehci_hcd doit être chargé avant ohci_hcd et uhci_hcd afin d'éviter un avertissement au moment du démarrage.

Créez un nouveau `/etc/modprobe.d/usb.conf` en lançant ce qui suit :

```
install -v -m755 -d /etc/modprobe.d
cat > /etc/modprobe.d/usb.conf << "EOF"
# Début de /etc/modprobe.d/usb.conf

install ohci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i ohci_hcd ; true
install uhci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i uhci_hcd ; true

# Fin de /etc/modprobe.d/usb.conf
EOF
```

10.3.3. Contenu de Linux

Fichiers installés: config-5.10.17, vmlinuz-5.10.17-lfs-10.1-systemd, et System.map-5.10.17
Répertoires installés: /lib/modules, /usr/share/doc/linux-5.10.17

Descriptions courtes

`config-5.10.17`	Contient toutes les options de configuration choisies pour le noyau
`vmlinuz-5.10.17-lfs-10.1-systemd`	Le moteur du système Linux. Au démarrage de l'ordinateur, le noyau est la première partie du système d'exploitation à être chargée. Il détecte et initialise tous composants matériels de l'ordinateur, puis rend disponible les composants dans une arborescence de fichiers pour les logiciels qui en ont besoin, et transforme une machine monoprocesseur en une machine multitâche capable d'exécuter plusieurs programmes quasi simultanément
`System.map-5.10.17`	Une liste d'adresses et de symboles donnant la correspondance entre les points d'entrée, et les adresses de toutes les fonctions et structures de données du noyau

10.4. Utiliser GRUB pour paramétrer le processus de démarrage

10.4.1. Introduction

Avertissement

Une mauvaise configuration de GRUB peut rendre votre système inutilisable si vous n'avez pas d'autre périphérique d'amorçage comme un cédérom. Cette section n'est pas obligatoire pour démarrer votre système LFS. Il se peut que vous vouliez simplement modifier votre chargeur de démarrage actuel, comme Grub-Legacy, GRUB2 ou LILO.

Assurez-vous d'avoir un disque de démarrage de façon à pouvoir « dépanner » l'ordinateur si celui-ci devenait inutilisable (non amorçable). Si vous n'avez pas déjà de périphérique de démarrage, vous pouvez en créer un. Afin que la procédure ci-dessous fonctionne, vous devez faire un tour du côté de BLFS et installer **`xorriso`** qui est dans le paquet *libisoburn*.

```
cd /tmp
grub-mkrescue --output=grub-img.iso
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

Note

Pour démarrer LFS sur les systèmes hôte qui ont l'UEFI activé, le noyau doit être construit avec la fonctionnalité CONFIG_EFI_STUB décrite dans la section précédente. Cependant, LFS peut être démarré avec GRUB2 sans cela. Pour cela, vous devez désactiver le mode UEFI et le Secure Boot dans le BIOS du système hôte. Pour des détails, voir *l'astuce lfs-uefi.txt* sur http://fr.linuxfromscratch.org/view/astuces/lfs-uefi.txt.

10.4.2. Conventions de nommage de GRUB

GRUB utilise sa propre nomenclature de disques et partitions, de la forme *(hdn,m)*, où *n* est le numéro du disque dur et *m* le numéro de la partition. Le numéro du disque dur commence à zéro, mais le numéro de la partition commence à un pour les partitions normales et à cinq pour les partitions étendues. Remarquez que ceci diffère des versions précédentes où les deux numéros commençaient à zéro. Par exemple, les partitions `sda1` correspond à *(hd0,1)* pour GRUB et `sdb3` correspond à *(hd1,3)*. Contrairement à Linux, GRUB ne considère pas les lecteurs de cédérom comme des disques durs. Par exemple, si un CD se trouve sur `hdb` et un second disque dur sur `hdc`, ce dernier disque sera malgré tout *(hd1)*."

10.4.3. Réglage de la configuration

GRUB fonctionne en écrivant les données sur le premier secteur physique du disque dur. Ce secteur ne fait partie d'aucun système de fichiers. Les programmes accèdent alors aux modules de GRUB dans la partition de démarrage. L'emplacement par défaut est /boot/grub/.

L'emplacement de la partition de démarrage est un choix de l'utilisateur qui conditionne la configuration. Une bonne pratique consiste à avoir une petite partition distincte (la taille suggérée est de 200 Mo) pour les informations d'amorçage. De cette façon, chaque construction, que ce soit LFS ou d'autres distributions commerciales, peut accéder aux mêmes fichiers de démarrage et n'importe quel système amorcé peut y accéder. Si vous choisissez cette option, vous aurez besoin de monter la partition séparément, de déplacer tous les fichiers du répertoire `/boot` actuel (par exemple, le noyau linux que vous venez de construire à l'étape précédente) vers la nouvelle partition. Vous aurez ensuite besoin de démonter la partition puis de la remonter en tant que `/boot`. Si vous le faites, assurez-vous de mettre à jour `/etc/fstab`.

L'utilisation de la partition lfs actuelle fonctionnera également, mais la configuration de plusieurs systèmes sera plus difficile.

En utilisant les informations ci-dessus, déterminez le nom adapté à la partition racine (ou partition de démarrage, s'il en existe une distincte). Pour l'exemple suivant, supposons que la partition racine (ou la partition de démarrage) est `sda2`.

Installez les fichiers de GRUB dans `/boot/grub` et paramétrez le secteur d'amorçage :

Avertissement

La commande suivante va écraser le chargeur de démarrage actuel. Ne lancez pas la commande si ce n'est pas ce que vous désirez, par exemple si vous utilisez un gestionnaire de démarrage extérieur pour gérer le Master Boot Record (MBR).

```
grub-install /dev/sda
```

Note

Si le système a été démarré en UEFI, **grub-install** essayera d'installer des fichiers pour la cible *x86_64-efi*, mais ces fichiers n'ont pas été installés au chapitre 6. Si c'est le cas, ajoutez `--target i386-pc` à la commande ci-dessus.

10.4.4. Créer le fichier de configuration de GRUB

Générez `/boot/grub/grub.cfg` :

```
cat > /boot/grub/grub.cfg << "EOF"
# Début de /boot/grub/grub.cfg
set default=0
set timeout=5

insmod ext2
set root=(hd0,2)

menuentry "GNU/Linux, Linux 5.10.17-lfs-10.1-systemd" {
        linux   /boot/vmlinuz-5.10.17-lfs-10.1-systemd root=/dev/sda2 ro
}
EOF
```

Note

Du point de vue de GRUB, les fichiers du noyau sont relatifs à la partition utilisée. Si vous avez utilisé une partition /boot distincte, supprimez /boot de la ligne *linux* ci-dessus. Vous devrez aussi modifier la ligne *set root* pour pointer vers la partition d'amorçage.

GRUB est un programme extrêmement puissant et il offre un très grand nombre d'options pour démarrer depuis une large gamme de périphériques, de systèmes d'exploitation et de types de partition. Il a aussi beaucoup d'options de personnalisation telles que les écrans d'accueil graphiques, les annonces sonores, l'entrée à la souris, etc. Les détails de ces options vont au-delà des objectifs de cette introduction.

Attention

Il existe une commande, grub-mkconfig qui peut écrire automatiquement un fichier de configuration. Elle utilise un ensemble de scripts situés dans /etc/grub.d/ et elle détruira les personnalisations que vous aurez faites. Ces scripts sont d'abord conçus pour des distributions qui ne se basent pas sur les sources et ils ne sont pas recommandés pour LFS. Si vous installez une distribution Linux commerciale, il est fort probable que ce programme soit lancé. Assurez-vous de sauvegarder votre fichier grub.cfg.

Chapitre 11. La fin

11.1. Fin

Bien joué ! Le nouveau système LFS est installé ! Nous vous souhaitons de bien vous amuser avec votre tout nouveau système Linux fabriqué sur mesure.

Il est recommandé de créer le fichier `/etc/lfs-release`. Avec ce fichier, il vous est très facile (ainsi que pour nous si vous avez besoin de demander de l'aide) de savoir quelle version de LFS vous avez installé sur votre système. Créez ce fichier en lançant :

```
echo 10.1-systemd > /etc/lfs-release
```

Certains paquets que vous installerez sur votre système pourront utiliser deux fichiers décrivant le système installé, soit sous forme binaire, soit à la construction.

Le premier affiche l'état de votre nouveau système, en respectant la Linux Standards Base (LSB). Pour créer ce fichier, lancez :

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="10.1-systemd"
DISTRIB_CODENAME="<votre nom ici>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

Le deuxième contient à peu près les mêmes informations et est utilisé par systemd et certains environnements de bureau. Pour créer ce fichier, lancez :

```
cat > /etc/os-release << "EOF"
NAME="Linux From Scratch"
VERSION="10.1-systemd"
ID=lfs
PRETTY_NAME="Linux From Scratch 10.1-systemd"
VERSION_CODENAME="<votre nom ici>"
EOF
```

Assurez-vous de personnaliser les champs « DISTRIB_CODENAME » et « VERSION_CODENAME » pour que ce système ne soit que le vôtre.

11.2. Enregistrez-vous

Maintenant que vous avez terminé le livre, voulez-vous être enregistré comme utilisateur de LFS ? Allez directement sur *http://www.linuxfromscratch.org/cgi-bin/lfscounter.php* et enregistrez-vous comme utilisateur LFS en entrant votre nom et la première version de LFS que vous ayez utilisée.

Redémarrons sur LFS maintenant.

11.3. Redémarrer le système

Maintenant que tous les logiciels ont été installés, il est temps de redémarrer votre ordinateur. Néanmoins, vous devez savoir certaines choses. Le système que vous avez créé dans ce livre est vraiment minimaliste et a toutes les chances de ne pas avoir les fonctionnalités dont vous aurez besoin pour continuer. En installant quelques paquets

supplémentaires à partir du livre BLFS en restant dans l'environnement chroot actuel, vous serez dans une bien meilleure position pour continuer une fois que vous aurez redémarré votre nouvelle installation LFS. Voici quelques suggestions :

- Un navigateur web en mode texte tel que *Lynx* vous permettra de lire facilement le livre BLFS dans un terminal virtuel tout en construisant les paquets dans un autre.
- Le paquet *GPM* vous permettra de réaliser des copier-coller dans vos terminaux virtuels.
- Si vous êtes dans une situation où la configuration IP statique ne correspond pas à vos besoins en termes de réseau, installer des paquets comme *dhcpcd* ou la partie client de *dhcp* peut être utile.
- Installer *sudo* peut être utile pour construire des paquets en tant qu'utilisateur non root et pour installer facilement les paquets qui en résultent dans votre nouveau système.
- Si vous voulez accéder à votre nouveau système depuis un environnement graphique confortable, installez *openssh*.
- Pour faciliter le rapatriement de fichiers par Internet, installez *wget*.
- Enfin, une relecture des fichiers de configuration suivants s'impose aussi à ce moment.
  - /etc/bashrc
  - /etc/dircolors
  - /etc/fstab
  - /etc/hosts
  - /etc/inputrc
  - /etc/profile
  - /etc/resolv.conf
  - /etc/vimrc
  - /root/.bash_profile
  - /root/.bashrc

Après cet intermède, démarrons notre toute nouvelle installation LFS pour la première fois ! Tout d'abord, quittez l'environnement chroot :

```
logout
```

Démontez la hiérarchie du système de fichiers LFS :

```
umount -Rv $LFS
```

Maintenant, redémarrez le système avec :

```
shutdown -r now
```

En supposant que le chargeur de démarrage GRUB a été initialisé comme indiqué plus tôt, le menu est prêt pour démarrer automatiquement *LFS 10.1*.

Quand le redémarrage est terminé, le système LFS est fonctionnel et davantage de logiciels peuvent être installés pour satisfaire vos besoins.

11.4. Et maintenant ?

Merci d'avoir lu le livre LFS. Nous espérons que vous avez trouvé ce livre utile et que vous en avez appris davantage sur le processus de création d'un système.

Maintenant que le système LFS est installé, vous êtes peut-être en train de vous demander « Et ensuite ? » Pour répondre à cette question, nous vous avons préparé une liste de ressources.

- Maintenance

  Les bogues et informations de sécurité sont rapportés régulièrement pour tous les logiciels. Comme un système LFS est compilé à partir des sources, c'est à vous de prendre en compte ces rapports. Il existe plusieurs ressources en ligne pour garder trace de tels rapports, quelques-unes d'entre elles sont indiquées ci-dessous :

  - *CERT* (Computer Emergency Response Team)

    CERT a une liste de diffusion publiant les alertes de sécurité concernant différents systèmes d'exploitation et applications. Les informations de souscription sont disponibles sur *http://www.us-cert.gov/cas/signup.html*.

  - Bugtraq

    Bugtraq est une liste de diffusion divulguant tous les problèmes de sécurité informatique. Elle publie les problèmes de sécurité qui viennent d'être découvert et occasionnellement leurs corrections potentielles. Les informations de souscription sont disponibles sur *http://www.securityfocus.com/archive*.

- Beyond Linux From Scratch

  Le livre Beyond Linux From Scratch (au-delà de Linux From Scratch) couvre les procédures d'installation d'un grand nombre de logiciels en dehors du livre LFS. Le projet BLFS est disponible sur *http://fr.linuxfromscratch.org/blfs/../view/blfs-10.1/*.

- Astuces LFS

  Les astuces LFS sont une collection de documents éducatifs soumis par des volontaires à la communauté LFS. Ces astuces sont disponibles sur *http://fr.linuxfromscratch.org/view/astuces/*.

- Listes de diffusion

  Il existe plusieurs listes de diffusion LFS auxquelles vous pouvez vous abonner si vous cherchez de l'aide, voulez suivre les derniers développements, voulez contribuer au projet et plus. Consultez le Chapitre 1 - Mailing Lists pour plus d'informations.

- Le projet de documentation Linux

  Le projet de documentation Linux (LDP) a pour but de favoriser la collaboration concernant la documentation de Linux. Le LDP offre une large collection de guides pratiques, livres et pages de manuel. Il est disponible sur *http://fr.tldp.org/*.

Partie V. Annexes

Annexe A. Acronymes et Termes

ABI *Application Binary Interface* ou Interface binaire-programme

ALFS *Automated Linux From Scratch*

API Interface de programmation d'application

ASCII *American Standard Code for Information Interchange* ou Code américain normalisé pour l'échange d'information

BIOS *Basic Input/Output System* ou Système d'entrées/sorties de base

BLFS *Beyond Linux From Scratch*

BSD *Berkeley Software Distribution*

chroot *change root*

CMOS *Complementary Metal Oxide Semiconductor* ou Semi-conducteur à oxyde métallique complémentaire

COS *Class Of Service*

CPU *Central Processing Unit* ou Unité centrale de traitement

CRC Contrôle de redondance cyclique

CVS *Concurrent Versions System* ou Système de gestion de versions

DHCP *Dynamic Host Configuration Protocol* ou Protocole de configuration dynamique d'adressage serveur

DNS *Domain Name Service* ou service de nom de domaine

EGA *Enhanced Graphics Adapter* ou Adaptateur graphique amélioré

ELF Format exécutable et liable

EOF *End of File* ou Fin de fichier

EQN Équation

ext2 *second extended file system*

ext3 *third extended file system*

ext4 *fourth extended file system*

FAQ Foire aux questions

FHS *Filesystem Hierarchy Standard* ou Hiérarchie standard des systèmes de fichiers

FIFO *First-In, First Out* ou premier arrivé, premier servi

FQDN *Fully Qualified Domain Name* ou Nom de domaine pleinement qualifié

FTP Protocole de transfert de fichiers

Go Gigaoctet

GCC *GNU Compiler Collection*

GID Identificateur de groupe

GMT Temps moyen de Greenwich

HTML *Hypertext Markup Language*

IDE *Integrated Drive Electronics* ou Gestion de périphériques à électronique intégrée

IEEE *Institute of Electrical and Electronic Engineers*

IO *Input/Output* ou Entrées/Sorties

IP Protocole Internet

IPC Communication inter-processus

IRC *Internet Relay Chat* ou Service d'échanges textuels en temps réel

ISO *International Organization for Standardization*

ISP *Internet Service Provider* ou Fournisseur d'accès Internet

Ko Kilooctet

LED *Light Emitting Diode* ou Diode électroluminescente

LFS *Linux From Scratch*

LSB *Linux Standard Base*

Mo Mégaoctet

MBR *Master Boot Record* ou Secteur d'amorçage

MD5 *Message Digest 5*

NIC *Network Interface Card* ou carte d'interface réseau

NLS Support de la langue maternelle

NNTP *Network News Transport Protocol* ou Protocole de transfert UseNet

NPTL *Native POSIX Threading Library*

OSS *Open Sound System*

PCH *Pre-Compiled Headers*

PCRE *Perl Compatible Regular Expression*

PID Identificateur de processus

PTY Pseudo terminal

QOS *Quality Of Service* ou Qualité de service

RAM *Random Access Memory* ou Mémoire vive

RPC *Remote Procedure Call* ou Appel de procédure distante

RTC *Real Time Clock* ou Horloge temps réel

SBU *Standard Build Unit*

SCO *The Santa Cruz Operation*

SHA1 *Secure-Hash Algorithm 1*

TLDP *The Linux Documentation Project*

TFTP *Trivial File Transfer Protocol* ou Protocole simplifié de transfert de fichiers

TLS *Thread-Local Storage* ou Mémoire locale de thread

UID Identificateur utilisateur

umask *user file-creation mask*

USB *Universal Serial Bus*

UTC Temps universel coordonné

UUID Identificateur universellement unique

VC Console Virtuelle

VGA Adaptateur graphique vidéo

VT Terminal virtuel

Annexe B. Remerciements

Nous aimerions remercier les personnes et organisations suivantes pour leurs contributions au projet Linux From Scratch.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – Créateur de LFS
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – Rédacteur en chef de LFS
- *Jim Gifford* <jim@linuxfromscratch.org> – Co-Leader du projet CLFS
- *Pierre Labastie* <pierre@linuxfromscratch.org> – Éditeur de BLFS et meneur de ALFS
- *DJ Lucas* <dj@linuxfromscratch.org> – éditeur de LFS et de BLFS
- *Ken Moffat* <ken@linuxfromscratch.org> – Éditeur de BLFS
- Sans compter les autres personnes sur les diverses listes de diffusion de LFS et BLFS qui ont aidé à rendre possible ce livre par leurs suggestions, leurs tests ; leurs soumissions de rapports de bogue, d'instructions et leurs retours d'expérience en installant divers paquets.

Traducteurs

- *Manuel Canales Esparcia* <macana@macana-es.com> – Projet de traduction de LFS en espagnol
- *Johan Lenglet* <johan@linuxfromscratch.org> – Projet de traduction de LFS en français jusqu'en 2008
- *Jean-Philippe Mengual* <jmengual@linuxfromscratch.org> – Projet de traduction de LFS en français 2008-2016
- *Julien Lepiller* <jlepiller@linuxfromscratch.org> – Projet de traduction de LFS en français depuis 2017
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Projet de traduction de LFS en portugais
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – Projet de traduction de LFS en allemand
- *Anton Maisak* <info@linuxfromscratch.org.ru> – Projet de traduction de LFS en russe
- *Elena Shevcova* <helen@linuxfromscratch.org.ru> – Projet de traduction de LFS en russe

Mainteneurs de miroirs

Miroirs Nord-Américains

- *Scott Kveton* <scott@osuosl.org> – miroir lfs.oregonstate.edu
- *William Astle* <lost@l-w.net> – miroir ca.linuxfromscratch.org
- *Eujon Sellers* <jpolen@rackspace.com> – miroir lfs.introspeed.com
- *Justin Knierim* <tim@idge.net> – miroir lfs-matrix.net

Miroirs Sud-américains

- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – miroir lfsmirror.lfs-es.info
- *Luis Falcon* <Luis Falcon> – miroir torredehanoi.org

Miroirs européens

- *Guido Passet* <guido@primerelay.net> – miroir nl.linuxfromscratch.org
- *Bastiaan Jacques* <baafie@planet.nl> – miroir lfs.pagefault.net
- *Sven Cranshoff* <sven.cranshoff@lineo.be> – miroir lfs.lineo.be

- Scarlet Belgium – miroir lfs.scarlet.be
- *Sebastian Faulborn* <info@aliensoft.org> – miroir lfs.aliensoft.org
- *Stuart Fox* <stuart@dontuse.ms> – miroir lfs.dontuse.ms
- *Ralf Uhlemann* <admin@realhost.de> – miroir lfs.oss-mirror.org
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – miroir at.linuxfromscratch.org
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – miroir se.linuxfromscratch.org
- *Franck* <franck@linuxpourtous.com> – miroir lfs.linuxpourtous.com
- *Philippe Baque* <baque@cict.fr> – miroir lfs.cict.fr
- *Vitaly Chekasin* <gyouja@pilgrims.ru> – miroir lfs.pilgrims.ru
- *Benjamin Heil* <kontakt@wankoo.org> – miroir lfs.wankoo.org
- *Anton Maisak* <info@linuxfromscratch.org.ru> – miroir linuxfromscratch.org.ri

Miroirs asiatiques

- *Satit Phermsawang* <satit@wbac.ac.th> – miroir lfs.phayoune.org
- *Shizunet Co.,Ltd.* <info@shizu-net.jp> – miroir lfs.mirror.shizu-net.jp
- *Init World* <http://www.initworld.com/> – miroir lfs.initworld.com

Miroirs australiens

- *Jason Andrade* <jason@dstc.edu.au> – miroir au.linuxfromscratch.org

Anciens membres de l'équipe du projet

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – éditeur du livre LFS
- Archaic <archaic@linuxfromscratch.org> – rédacteur technique LFS/éditeur, leader du projet HLFS, éditeur de BLFS, mainteneur des projets d'astuces et correctifs
- *Matthew Burgess* <matthew@linuxfromscratch.org> – leader du projet LFS, rédacteur technique LFS/éditeur
- *Nathan Coulson* <nathan@linuxfromscratch.org> – mainteneur de LFS-Bootscripts
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Développeur du site Web, mainteneur de la FAQ
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – mainteneur de LFS/BLFS/HLFS en XML et XSL
- Alex Groenewoud – rédacteur technique LFS
- Marc Heerdink
- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – rédacteur technique LFS, mainteneur du LiveCD LFS
- *Bryan Kadzban* <bryan@linuxfromscratch.org> – rédacteur technique LFS
- Mark Hymers
- Seth W. Klein – mainteneur de la FAQ
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – mainteneur du Wiki
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – mainteneur des scripts d'arrière-plan du site Web

- *Randy McMurchy* <randy@linuxfromscratch.org> – Leader du projet BLFS, éditeur LFS
- *Dan Nicholson* <dnicholson@linuxfromscratch.org> – éditeur de LFS et BLFS
- *Alexander E. Patrakov* <alexander@linuxfromscratch.org> – rédacteur Technique LFS, éditeur de LFS international, mainteneur du LiveCD LFS
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – mainteneur de LFS NNTP Gateway
- *Douglas R. Reno* <renodr@linuxfromscratch.org> – Éditeur Systemd
- *Ryan Oliver* <ryan@linuxfromscratch.org> – Co-Leader du projet CLFS
- *Greg Schafer* <gschafer@zip.com.au> – rédacteur technique LFS et architecte de la nouvelle méthode de construction activant le 64 bits
- Jesse Tie-Ten-Quee – rédacteur technique LFS
- *James Robertson* <jwrober@linuxfromscratch.org> – mainteneur Bugzilla
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – éditeur du livre BLFS, leader du projet d'astuces et correctifs
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – rédacteur technique LFS, mainteneur Bugzilla, mainteneur de LFS-Bootscripts
- *Zack Winkles* <zwinkles@gmail.com> – rédacteur technique LFS

Annexe C. Dépendances

La compilation et l'installation correcte de chaque paquet compilé dans LFS dépend d'un ou plusieurs autres paquets. Certains paquets participent même aux dépendances circulaires, c'est-à-dire que le premier paquet dépend du second qui dépend à son tour du premier. À cause de ces dépendances, l'ordre dans lequel les paquets sont compilés dans LFS est très important. Le but de cette page est de documenter les dépendances de chaque paquet compilé dans LFS.

Pour chaque paquet que nous compilons, nous avons listé trois, parfois quatre types de dépendances. La première concerne les autres paquets qui doivent être disponibles afin de compiler et d'installer le paquet en question. La deuxième concerne les paquets qui, en plus de ceux de la première liste, doivent être disponibles afin de lancer les suites de test. La troisième liste de dépendances contient les paquets qui exigent ce paquet pour être compilés et installés à l'emplacement final avant qu'ils ne soient compilés et installés. Dans la plupart des cas, c'est parce que ces paquets lieront les chemins aux binaires à l'intérieur de leurs scripts. S'ils ne sont pas compilés dans un certain ordre, ceci pourrait aboutir à ce que des chemins vers /tools/bin/[binaire] soient placés à l'intérieur de scripts installés dans le système final. Cela n'est évidemment pas souhaitable.

La dernière indique les dépendances facultatives qui ne sont pas destinées à LFS mais qui pourraient être utiles à l'utilisateur. Ces paquets peuvent avoir eux-mêmes des dépendances supplémentaires obligatoires ou facultatives. Pour ces dépendances, la pratique recommandée consiste à les installer après avoir terminé le livre LFS puis à revenir en arrière pour reconstruire le paquet LFS. Dans certains cas, la réinstallation est traitée dans BLFS.

Acl

L'installation dépend de: Attr, Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed et Texinfo
La suite de tests dépend de: Automake, Diffutils, Findutils et Libtool
Doit être installé avant: Coreutils, Sed, Tar et Vim
Dépendances facultatives: Aucun

Attr

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed et Texinfo
La suite de tests dépend de: Automake, Diffutils, Findutils et Libtool
Doit être installé avant: Acl et Libcap
Dépendances facultatives: Aucun

Autoconf

L'installation dépend de: Bash, Coreutils, Grep, M4, Make, Perl, Sed et Texinfo
La suite de tests dépend de: Automake, Diffutils, Findutils, GCC et Libtool
Doit être installé avant: Automake
Dépendances facultatives: Emacs

Automake

L'installation dépend de: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed et Texinfo
La suite de tests dépend de: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool et Tar
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Bash

L'installation dépend de: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed et Texinfo
La suite de tests dépend de: Shadow
Doit être installé avant: Aucun
Dépendances facultatives: Xorg

Bc

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Grep et Make
La suite de tests dépend de: Gawk
Doit être installé avant: Noyau Linux
Dépendances facultatives: Aucun

Binutils

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, File, Flex, Gawk, GCC, Glibc, Grep, Make, Perl, Sed, Texinfo et Zlib
La suite de tests dépend de: DejaGNU et Expect
Doit être installé avant: Aucun
Dépendances facultatives: Debuginfod

Bison

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl et Sed
La suite de tests dépend de: Diffutils, Findutils et Flex
Doit être installé avant: Kbd et Tar
Dépendances facultatives: Doxygen (suite de tests)

Bzip2

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, et Patch
La suite de tests dépend de: Aucun
Doit être installé avant: File
Dépendances facultatives: Aucun

Check

L'installation dépend de: GCC, Grep, Make, Sed et Texinfo
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Coreutils

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Libcap, Make, Patch, Perl, Sed et Texinfo

La suite de tests dépend de: Diffutils, E2fsprogs, Findutils, Shadow et Util-linux

Doit être installé avant: Bash, Diffutils, Eudev, Findutils et Man-DB

Dépendances facultatives: Perl Expect et les modules IO:Tty (pour la suite de tests)

DejaGNU

L'installation dépend de: Bash, Coreutils, Diffutils, GCC, Grep, Make, Sed et Texinfo

La suite de tests dépend de: Aucun

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

Diffutils

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo

La suite de tests dépend de: Perl

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

E2fsprogs

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Sed, Texinfo et Util-linux

La suite de tests dépend de: Procps-ng et Psmisc

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

Eudev

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Gperf, Make, Sed et Util-linux

La suite de tests dépend de: Aucun

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

Expat

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make et Sed

La suite de tests dépend de: Aucun

Doit être installé avant: Python et XML::Parser

Dépendances facultatives: Aucun

Expect

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed et Tcl

La suite de tests dépend de: Aucun

Doit être installé avant: Aucun

Dépendances facultatives: *tk*

File

L'installation dépend de: Bash, Binutils, Bzip2, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, Xz et Zlib

La suite de tests dépend de: Aucun

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

Findutils

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo

La suite de tests dépend de: DejaGNU, Diffutils et Expect

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

Flex

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed et Texinfo

La suite de tests dépend de: Bison et Gawk

Doit être installé avant: Binutils, IProute2, Kbd, Kmod et Man-DB

Dépendances facultatives: Aucun

Gawk

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, MPFR, Patch, Readline, Sed et Texinfo

La suite de tests dépend de: Diffutils

Doit être installé avant: Aucun

Dépendances facultatives: libsigsegv

GCC

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, Texinfo et Zstd
La suite de tests dépend de: DejaGNU, Expect et Shadow
Doit être installé avant: Aucun
Dépendances facultatives: *GNAT* et *ISL*

GDBM

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make et Sed
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Gettext

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Sed et Texinfo
La suite de tests dépend de: Diffutils, Perl et Tcl
Doit être installé avant: Automake et Bison
Dépendances facultatives: Aucun

Glibc

L'installation dépend de: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Linux API Headers, Make, Perl, Python, Sed et Texinfo
La suite de tests dépend de: File
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

GMP

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed et Texinfo
La suite de tests dépend de: Aucun
Doit être installé avant: MPFR et GCC
Dépendances facultatives: Aucun

Gperf

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc et Make
La suite de tests dépend de: Diffutils et Expect
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Grep

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed et Texinfo

La suite de tests dépend de: Gawk

Doit être installé avant: Man-DB

Dépendances facultatives: Pcre et libsigsegv

Groff

L'installation dépend de: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed et Texinfo

La suite de tests dépend de: Aucune suite de tests disponible

Doit être installé avant: Man-DB et Perl

Dépendances facultatives: Ghostscript

GRUB

L'installation dépend de: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo et Xz

La suite de tests dépend de: Aucun

Doit être installé avant: Aucun

Dépendances facultatives: Aucun

Gzip

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, et Texinfo

La suite de tests dépend de: Diffutils et Less

Doit être installé avant: Man-DB

Dépendances facultatives: Aucun

Iana-Etc

L'installation dépend de: Coreutils, Gawk et Make

La suite de tests dépend de: Aucune suite de tests disponible

Doit être installé avant: Perl

Dépendances facultatives: Aucun

Inetutils

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo et Zlib
La suite de tests dépend de: Aucun
Doit être installé avant: Tar
Dépendances facultatives: Aucun

Intltool

L'installation dépend de: Bash, Gawk, Glibc, Make, Perl, Sed et XML::Parser
La suite de tests dépend de: Perl
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

IProute2

L'installation dépend de: Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, Libcap, Libelf et Linux API Headers
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Aucun
Dépendances facultatives: Berkeley DB et Iptables

Kbd

L'installation dépend de: Bash, Binutils, Bison, Check, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch et Sed
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Kmod

L'installation dépend de: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Pkg-config, Sed, Xz-Utils et Zlib
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Eudev
Dépendances facultatives: Aucun

Less

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses et Sed
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Gzip
Dépendances facultatives: Pcre

Libcap

L'installation dépend de: Attr, Bash, Binutils, Coreutils, GCC, Glibc, Perl, Make et Sed
La suite de tests dépend de: Aucun
Doit être installé avant: IProute2 et Shadow
Dépendances facultatives: Linux-PAM

Libelf

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc et Make
La suite de tests dépend de: Aucun
Doit être installé avant: IProute2 et le noyau Linux
Dépendances facultatives: Aucun

Libffi

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Make et Sed
La suite de tests dépend de: DejaGNU
Doit être installé avant: Python
Dépendances facultatives: Aucun

Libpipeline

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed et Texinfo
La suite de tests dépend de: Check
Doit être installé avant: Man-DB
Dépendances facultatives: Aucun

Libtool

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed et Texinfo
La suite de tests dépend de: Autoconf, Automake et Findutils
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Noyau Linux

L'installation dépend de: Bash, Bc, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Libelf, Make, Ncurses, OpenSSL, Perl et Sed
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

M4

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, et Texinfo
La suite de tests dépend de: Diffutils
Doit être installé avant: Autoconf et Bison
Dépendances facultatives: libsigsegv

Make

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo
La suite de tests dépend de: Perl et Procps-ng
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Man-DB

L'installation dépend de: Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Sed et Xz
La suite de tests dépend de: Util-linux
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Man-Pages

L'installation dépend de: Bash, Coreutils et Make
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Meson

L'installation dépend de: Ninja et Python
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Systemd
Dépendances facultatives: Aucun

MPC

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed et Texinfo
La suite de tests dépend de: Aucun
Doit être installé avant: GCC
Dépendances facultatives: Aucun

MPFR

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed et Texinfo
La suite de tests dépend de: Aucun
Doit être installé avant: Gawk et GCC
Dépendances facultatives: Aucun

Ncurses

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch et Sed
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Bash, GRUB, Inetutils, Less, Procps-ng, Psmisc, Readline, Texinfo, Util-linux et Vim
Dépendances facultatives: Aucun

Ninja

L'installation dépend de: Binutils, Coreutils, GCC et Python
La suite de tests dépend de: Aucun
Doit être installé avant: Meson
Dépendances facultatives: Asciidoc, Doxygen, Emacs et re2c

Openssl

L'installation dépend de: Binutils, Coreutils, GCC, Make et Perl
La suite de tests dépend de: Aucun
Doit être installé avant: Linux
Dépendances facultatives: Aucun

Patch

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make et Sed
La suite de tests dépend de: Diffutils
Doit être installé avant: Aucun
Dépendances facultatives: Ed

Perl

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Groff, Make, Sed et Zlib
La suite de tests dépend de: Iana-Etc et Procps-ng
Doit être installé avant: Autoconf
Dépendances facultatives: Aucun

Pkg-config

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Popt et Sed
La suite de tests dépend de: Aucun
Doit être installé avant: Kmod
Dépendances facultatives: Aucun

Popt

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep et Make
La suite de tests dépend de: Diffutils et Sed
Doit être installé avant: Pkg-config
Dépendances facultatives: Aucun

Procps-ng

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Make et Ncurses
La suite de tests dépend de: DejaGNU
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Psmisc

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, et Sed
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Python

L'installation dépend de: Bash, Binutils, Coreutils, Expat, GCC, Gdbm, Gettext, Glibc, Grep, Libffi, Make, Ncurses, Sed et Util-linux
La suite de tests dépend de: GDB et
Doit être installé avant: Ninja
Dépendances facultatives: Berkeley DB, OpenSSL, SQLite et Tk

Readline

L'installation dépend de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed et Texinfo
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Bash et Gawk
Dépendances facultatives: Aucun

Sed

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed et Texinfo
La suite de tests dépend de: Diffutils et Gawk
Doit être installé avant: E2fsprogs, File, Libtool et Shadow
Dépendances facultatives: Aucun

Shadow

L'installation dépend de: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Libcap, Make et Sed
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Coreutils
Dépendances facultatives: Cracklib et PAM

Sysklogd

L'installation dépend de: Binutils, Coreutils, GCC, Glibc, Make et Patch
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Systemd

L'installation dépend de: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Expat, Gawk, GCC, Glibc, Gperf, Grep, Intltool, Libcap, Meson, Sed et Util-linux
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: Beaucoup, voir *la page systemd de BLFS*

Sysvinit

L'installation dépend de: Binutils, Coreutils, GCC, Glibc, Make et Sed
La suite de tests dépend de: Aucune suite de tests disponible
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Tar

L'installation dépend de: Acl, Attr, Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed et Texinfo
La suite de tests dépend de: Autoconf, Diffutils, Findutils, Gawk et Gzip
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Tcl

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make et Sed
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Texinfo

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch et Sed
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: Aucun

Util-linux

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, Eudev, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed et Zlib
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: *Libcap-ng*

Vim

L'installation dépend de: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses et Sed
La suite de tests dépend de: Aucun
Doit être installé avant: Aucun
Dépendances facultatives: Xorg, GTK+2, LessTif, Python, Tcl, Ruby et GPM

XML::Parser

L'installation dépend de: Bash, Binutils, Coreutils, Expat, GCC, Glibc, Make et Perl
La suite de tests dépend de: Perl
Doit être installé avant: Intltool
Dépendances facultatives: Aucun

Xz

L'installation dépend de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc et Make
La suite de tests dépend de: Aucun
Doit être installé avant: Eudev, File, GRUB, Kmod et Man-DB
Dépendances facultatives: Aucun

Zlib

L'installation dépend de: Bash, Binutils, Coreutils, GCC, Glibc, Make et Sed

La suite de tests dépend de: Aucun

Doit être installé avant: File, Kmod, Perl et Util-linux

Dépendances facultatives: Aucun

Zstd

L'installation dépend de: Binutils, Coreutils, GCC, Glibc, Gzip, Make et Xz

La suite de tests dépend de: Aucun

Doit être installé avant: GCC

Dépendances facultatives: Aucun

Annexe D. Licences LFS

Ce livre est couvert par la licence Creative Commons Attribution-NonCommercial-ShareAlike 2.0.

Les instructions destinées à l'ordinateur peuvent être extraites selon les termes de la licence MIT.

D.1. Creative Commons License

Creative Commons Legal Code

Attribution-NonCommercial-ShareAlike 2.0

Important

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

   a. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.

   b. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.

   c. "Licensor" means the individual or entity that offers the Work under the terms of this License.

   d. "Original Author" means the individual or entity who created the Work.

   e. "Work" means the copyrightable work of authorship offered under the terms of this License.

   f. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

g. "License Elements" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, Noncommercial, ShareAlike.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

   a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;

   b. to create and reproduce Derivative Works;

   c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

   d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works;

   The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Sections 4(e) and 4(f).

4. Restrictions.The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

   a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any reference to such Licensor or the Original Author, as requested.

   b. You may distribute, publicly display, publicly perform, or publicly digitally perform a Derivative Work only under the terms of this License, a later version of this License with the same License Elements as this License, or a Creative Commons iCommons license that contains the same License Elements as this License (e.g. Attribution-NonCommercial-ShareAlike 2.0 Japan). You must include a copy of, or the Uniform Resource Identifier for, this License or other license specified in the previous sentence with every copy or phonorecord of each Derivative Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Derivative Works that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder, and You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Derivative Work with any technological measures that control access or use

of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Derivative Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Derivative Work itself to be made subject to the terms of this License.

c. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

d. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Derivative Works or Collective Works, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

e. For the avoidance of doubt, where the Work is a musical composition:

   i. Performance Royalties Under Blanket Licenses. Licensor reserves the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work if that performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

   ii. Mechanical Rights and Statutory Royalties. Licensor reserves the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions), if Your distribution of such cover version is primarily intended for or directed toward commercial advantage or private monetary compensation. 6. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

f. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR

OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

   a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

   b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

   a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

   b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

   c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

   d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

   e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Important

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, neither party will use the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time.

Creative Commons may be contacted at *http://creativecommons.org/*.

D.2. The MIT License

Copyright © 1999-2021 Gerard Beekmans

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Index

Paquets

Acl: 126
Attr: 125
Autoconf: 160
Automake: 161
Bash: 147
  tools: 56
Bash: 147
  tools: 56
Bc: 112
Binutils: 118
  outils, passe 1: 42
  tools, passe 2: 69
Binutils: 118
  outils, passe 1: 42
  tools, passe 2: 69
Binutils: 118
  outils, passe 1: 42
  tools, passe 2: 69
Bison: 145
  tools: 81
Bison: 145
  tools: 81
Bzip2: 103
Check: 177
Coreutils: 172
  tools: 57
Coreutils: 172
  tools: 57
D-Bus: 211
DejaGNU: 117
Diffutils: 178
  tools: 58
Diffutils: 178
  tools: 58
E2fsprogs: 220
Expat: 152
Expect: 116
File: 108
  tools: 59
File: 108
  tools: 59
Findutils: 180
  tools: 60
Findutils: 180
  tools: 60
Flex: 113
Gawk: 179
  tools: 61
Gawk: 179
  tools: 61
GCC: 132
  outils, passe 1: 44
  tools, libstdc++ passe 1: 51
  tools, libstdc++ passe 2: 79
  tools, passe 2: 70
GCC: 132
  outils, passe 1: 44
  tools, libstdc++ passe 1: 51
  tools, libstdc++ passe 2: 79
  tools, passe 2: 70
GCC: 132
  outils, passe 1: 44
  tools, libstdc++ passe 1: 51
  tools, libstdc++ passe 2: 79
  tools, passe 2: 70
GCC: 132
  outils, passe 1: 44
  tools, libstdc++ passe 1: 51
  tools, libstdc++ passe 2: 79
  tools, passe 2: 70
GCC: 132
  outils, passe 1: 44
  tools, libstdc++ passe 1: 51
  tools, libstdc++ passe 2: 79
  tools, passe 2: 70
GDBM: 150
Gettext: 143
  tools: 80
Gettext: 143
  tools: 80
Glibc: 95
  outils: 48
Glibc: 95
  outils: 48
GMP: 121
Gperf: 151
Grep: 146
  tools: 62
Grep: 146
  tools: 62
Groff: 181
GRUB: 184
Gzip: 187
  tools: 63
Gzip: 187
  tools: 63

Iana-Etc: 94
Inetutils: 153
Intltool: 159
IPRoute2: 189
Kbd: 191
Kmod: 162
Less: 186
Libcap: 127
Libelf: 164
libffi: 165
Libpipeline: 193
Libtool: 149
Linux: 245
  outils, en-têtes API: 47
Linux: 245
  outils, en-têtes API: 47
M4: 111
  tools: 53
M4: 111
  tools: 53
Make: 194
  tools: 64
Make: 194
  tools: 64
Man-DB: 196
Man-pages: 93
Meson: 171
MPC: 124
MPFR: 123
Ncurses: 138
  tools: 54
Ncurses: 138
  tools: 54
Ninja: 170
OpenSSL: 166
Patch: 195
  tools: 65
Patch: 195
  tools: 65
Perl: 155
  tools: 82
Perl: 155
  tools: 82
Pkgconfig: 137
Procps-ng: 213
Psmisc: 142
Python: 168
  temporary: 83
Python: 168
  temporary: 83
Readline: 109
Sed: 141
  tools: 66
Sed: 141
  tools: 66
Shadow: 128
  configuration: 129
Shadow: 128
  configuration: 129
systemd: 205
Tar: 199
  tools: 67
Tar: 199
  tools: 67
Tcl: 114
Texinfo: 200
  temporary: 84
Texinfo: 200
  temporary: 84
Udev
  utilisation: 229
Util-linux: 215
  tools: 85
Util-linux: 215
  tools: 85
Vim: 202
XML::Parser: 158
Xz: 105
  tools: 68
Xz: 105
  tools: 68
Zlib: 102
zstd: 107

Programmes

[: 172, 173
2to3: 168
accessdb: 196, 197
aclocal: 161, 161
aclocal-1.16: 161, 161
addftinfo: 181, 181
addpart: 215, 216
addr2line: 118, 119
afmtodit: 181, 181
agetty: 215, 216
apropos: 196, 197
ar: 118, 119
as: 118, 119
attr: 125, 125
autoconf: 160, 160

autoheader: 160, 160
autom4te: 160, 160
automake: 161, 161
automake-1.16: 161, 161
autopoint: 143, 143
autoreconf: 160, 160
autoscan: 160, 160
autoupdate: 160, 160
awk: 179, 179
b2sum: 172, 173
badblocks: 220, 221
base64: 172, 173, 172, 173
base64: 172, 173, 172, 173
basename: 172, 173
basenc: 172, 173
bash: 147, 148
bashbug: 147, 148
bc: 112, 112
bison: 145, 145
blkdiscard: 215, 216
blkid: 215, 216
blkzone: 215, 216
blockdev: 215, 216
bootctl: 205, 208
bridge: 189, 189
bunzip2: 103, 104
busctl: 205, 208
bzcat: 103, 104
bzcmp: 103, 104
bzdiff: 103, 104
bzegrep: 103, 104
bzfgrep: 103, 104
bzgrep: 103, 104
bzip2: 103, 104
bzip2recover: 103, 104
bzless: 103, 104
bzmore: 103, 104
c++: 132, 135
c++filt: 118, 119
cal: 215, 216
capsh: 127, 127
captoinfo: 138, 139
cat: 172, 173
catchsegv: 95, 100
catman: 196, 197
cc: 132, 135
cfdisk: 215, 216
chacl: 126, 126
chage: 128, 130
chattr: 220, 221
chcon: 172, 173
chcpu: 215, 216
checkmk: 177, 177
chem: 181, 181
chfn: 128, 130
chgpasswd: 128, 130
chgrp: 172, 174
chmem: 215, 216
chmod: 172, 174
choom: 215, 216
chown: 172, 174
chpasswd: 128, 130
chroot: 172, 174
chrt: 215, 216
chsh: 128, 130
chvt: 191, 192
cksum: 172, 174
clear: 138, 140
cmp: 178, 178
col: 215, 216
colcrt: 215, 216
colrm: 215, 216
column: 215, 216
comm: 172, 174
compile_et: 220, 221
coredumpctl: 205, 208
corelist: 155, 156
cp: 172, 174
cpan: 155, 156
cpp: 132, 135
csplit: 172, 174
ctrlaltdel: 215, 216
ctstat: 189, 189
cut: 172, 174
c_rehash: 166, 166
date: 172, 174
dbus-cleanup-sockets: 211, 212
dbus-daemon: 211, 212
dbus-launch: 211, 212
dbus-monitor: 211, 212
dbus-run-session: 211, 212
dbus-send: 211, 212
dbus-test-tool: 211, 212
dbus-update-activation-environment: 211, 212
dbus-uuidgen: 211, 212
dc: 112, 112
dd: 172, 174
deallocvt: 191, 192
debugfs: 220, 221
delpart: 215, 216

depmod: 162, 162
df: 172, 174
diff: 178, 178
diff3: 178, 178
dir: 172, 174
dircolors: 172, 174
dirname: 172, 174
dmesg: 215, 216
dnsdomainname: 153, 154
du: 172, 174
dumpe2fs: 220, 221
dumpkeys: 191, 192
e2freefrag: 220, 221
e2fsck: 220, 221
e2image: 220, 221
e2label: 220, 221
e2mmpstatus: 220, 221
e2scrub: 220, 221
e2scrub_all: 220, 221
e2undo: 220, 221
e4crypt: 220, 221
e4defrag: 220, 221
echo: 172, 174
egrep: 146, 146
eject: 215, 216
elfedit: 118, 119
enc2xs: 155, 156
encguess: 155, 156
env: 172, 174
envsubst: 143, 143
eqn: 181, 181
eqn2graph: 181, 181
ex: 202, 204
expand: 172, 174
expect: 116, 116
expiry: 128, 130
expr: 172, 174
factor: 172, 174
faillog: 128, 130
fallocate: 215, 216
false: 172, 174
fdformat: 215, 216
fdisk: 215, 216
fgconsole: 191, 192
fgrep: 146, 146
file: 108, 108
filefrag: 220, 222
fincore: 215, 217
find: 180, 180
findfs: 215, 217
findmnt: 215, 217
flex: 113, 113
flex++: 113, 113
flock: 215, 217
fmt: 172, 174
fold: 172, 174
free: 213, 214
fsck: 215, 217
fsck.cramfs: 215, 217
fsck.ext2: 220, 222
fsck.ext3: 220, 222
fsck.ext4: 220, 222
fsck.minix: 215, 217
fsfreeze: 215, 217
fstrim: 215, 217
ftp: 153, 154
fuser: 142, 142
g++: 132, 135
gawk: 179, 179
gawk-5.1.0: 179, 179
gcc: 132, 135
gc-ar: 132, 135
gc-nm: 132, 135
gc-ranlib: 132, 135
gcov: 132, 136
gcov-dump: 132, 136
gcov-tool: 132, 136
gdbmtool: 150, 150
gdbm_dump: 150, 150
gdbm_load: 150, 150
gdiffmk: 181, 181
gencat: 95, 100
genl: 189, 189
getcap: 127, 127
getconf: 95, 100
getent: 95, 100
getfacl: 126, 126
getfattr: 125, 125
getkeycodes: 191, 192
getopt: 215, 217
getpcaps: 127, 127
gettext: 143, 143
gettext.sh: 143, 143
gettextize: 143, 143
glilypond: 181, 181
gpasswd: 128, 130
gperf: 151, 151
gperl: 181, 181
gpinyin: 181, 181
gprof: 118, 120

grap2graph: 181, 181
grep: 146, 146
grn: 181, 182
grodvi: 181, 182
groff: 181, 182
groffer: 181, 182
grog: 181, 182
grolbp: 181, 182
grolj4: 181, 182
gropdf: 181, 182
grops: 181, 182
grotty: 181, 182
groupadd: 128, 130
groupdel: 128, 130
groupmems: 128, 130
groupmod: 128, 130
groups: 172, 174
grpck: 128, 130
grpconv: 128, 130
grpunconv: 128, 130
grub-bios-setup: 184, 184
grub-editenv: 184, 185
grub-file: 184, 185
grub-fstest: 184, 185
grub-glue-efi: 184, 185
grub-install: 184, 185
grub-kbdcomp: 184, 185
grub-macbless: 184, 185
grub-menulst2cfg: 184, 185
grub-mkconfig: 184, 185
grub-mkimage: 184, 185
grub-mklayout: 184, 185
grub-mknetdir: 184, 185
grub-mkpasswd-pbkdf2: 184, 185
grub-mkrelpath: 184, 185
grub-mkrescue: 184, 185
grub-mkstandalone: 184, 185
grub-ofpathname: 184, 185
grub-probe: 184, 185
grub-reboot: 184, 185
grub-render-label: 184, 185
grub-script-check: 184, 185
grub-set-default: 184, 185
grub-setup: 184, 185
grub-syslinux2cfg: 184, 185
gunzip: 187, 187
gzexe: 187, 187
gzip: 187, 187
h2ph: 155, 156
h2xs: 155, 156
halt: 205, 208
head: 172, 174
hexdump: 215, 217
hostid: 172, 174
hostname: 153, 154
hostnamectl: 205, 208
hpftodit: 181, 182
hwclock: 215, 217
i386: 215, 217
iconv: 95, 100
iconvconfig: 95, 100
id: 172, 174
idle3: 168
ifcfg: 189, 189
ifconfig: 153, 154
ifnames: 160, 160
ifstat: 189, 189
indxbib: 181, 182
info: 200, 201
infocmp: 138, 140
infotocap: 138, 140
init: 205, 208
insmod: 162, 163
install: 172, 174
install-info: 200, 201
instmodsh: 155, 156
intltool-extract: 159, 159
intltool-merge: 159, 159
intltool-prepare: 159, 159
intltool-update: 159, 159
intltoolize: 159, 159
ionice: 215, 217
ip: 189, 189
ipcmk: 215, 217
ipcrm: 215, 217
ipcs: 215, 217
isosize: 215, 217
join: 172, 174
journalctl: 205, 208
json_pp: 155, 156
kbdinfo: 191, 192
kbdrate: 191, 192
kbd_mode: 191, 192
kernel-install: 205, 208
kill: 215, 217
killall: 142, 142
kmod: 162, 163
last: 215, 217
lastb: 215, 217
lastlog: 128, 130

ld: 118, 120
ld.bfd: 118, 120
ld.gold: 118, 120
ldattach: 215, 217
ldconfig: 95, 100
ldd: 95, 100
lddlibc4: 95, 100
less: 186, 186
lessecho: 186, 186
lesskey: 186, 186
lex: 113, 113
lexgrog: 196, 198
lfskernel-5.10.17: 245, 249
libasan: 132, 136
libatomic: 132, 136
libcc1: 132, 136
libnetcfg: 155, 156
libtool: 149, 149
libtoolize: 149, 149
link: 172, 174
linux32: 215, 217
linux64: 215, 217
lkbib: 181, 182
ln: 172, 174
lnstat: 189, 190
loadkeys: 191, 192
loadunimap: 191, 192
locale: 95, 100
localectl: 205, 208
localedef: 95, 100
locate: 180, 180
logger: 215, 217
login: 128, 130
loginctl: 205, 208
logname: 172, 175
logoutd: 128, 130
logsave: 220, 222
look: 215, 217
lookbib: 181, 182
losetup: 215, 217
ls: 172, 175
lsattr: 220, 222
lsblk: 215, 217
lscpu: 215, 217
lsipc: 215, 217
lslocks: 215, 217
lslogins: 215, 217
lsmem: 215, 217
libkmod: 162, 163
lsns: 215, 217
lzcat: 105, 105
lzcmp: 105, 105
lzdiff: 105, 105
lzegrep: 105, 105
lzfgrep: 105, 105
lzgrep: 105, 105
lzless: 105, 105
lzma: 105, 105
lzmadec: 105, 106
lzmainfo: 105, 106
lzmore: 105, 106
m4: 111, 111
machinectl: 205, 209
make: 194, 194
makedb: 95, 100
makeinfo: 200, 201
man: 196, 198
mandb: 196, 198
manpath: 196, 198
mapscrn: 191, 192
mcookie: 215, 218
md5sum: 172, 175
mesg: 215, 218
meson: 171, 171
mkdir: 172, 175
mke2fs: 220, 222
mkfifo: 172, 175
mkfs: 215, 218
mkfs.bfs: 215, 218
mkfs.cramfs: 215, 218
mkfs.ext2: 220, 222
mkfs.ext3: 220, 222
mkfs.ext4: 220, 222
mkfs.minix: 215, 218
mklost+found: 220, 222
mknod: 172, 175
mkswap: 215, 218
mktemp: 172, 175
mk_cmds: 220, 222
mmroff: 181, 182
lsmod: 162, 163
modinfo: 162, 163
more: 215, 218
mount: 215, 218
mountpoint: 215, 218
msgattrib: 143, 143
msgcat: 143, 144
msgcmp: 143, 144
msgcomm: 143, 144
msgconv: 143, 144

msgen: 143, 144
msgexec: 143, 144
msgfilter: 143, 144
msgfmt: 143, 144
msggrep: 143, 144
msginit: 143, 144
msgmerge: 143, 144
msgunfmt: 143, 144
msguniq: 143, 144
mtrace: 95, 100
mv: 172, 175
namei: 215, 218
ncursesw6-config: 138, 140
neqn: 181, 182
networkctl: 205, 209
newgidmap: 128, 130
newgrp: 128, 130
newuidmap: 128, 130
newusers: 128, 130
ngettext: 143, 144
nice: 172, 175
ninja: 170, 170
nl: 172, 175
nm: 118, 120
nohup: 172, 175
nologin: 128, 130
nproc: 172, 175
nroff: 181, 182
nscd: 95, 100
nsenter: 215, 218
nstat: 189, 190
numfmt: 172, 175
objcopy: 118, 120
objdump: 118, 120
od: 172, 175
openssl: 166, 166
openvt: 191, 192
partx: 215, 218
passwd: 128, 130
paste: 172, 175
patch: 195, 195
pathchk: 172, 175
pcprofiledump: 95, 100
pdfmom: 181, 182
pdfroff: 181, 182
pdftexi2dvi: 200, 201
peekfd: 142, 142
perl: 155, 156
perl5.32.1: 155, 156
perlbug: 155, 156
perldoc: 155, 156
perlivp: 155, 156
perlthanks: 155, 156
pfbtops: 181, 182
pgrep: 213, 214
pic: 181, 182
pic2graph: 181, 182
piconv: 155, 156
pidof: 213, 214
ping: 153, 154
ping6: 153, 154
pinky: 172, 175
pip3: 168
pivot_root: 215, 218
pkg-config: 137, 137
pkill: 213, 214
pl2pm: 155, 156
pldd: 95, 100
pmap: 213, 214
pod2html: 155, 157
pod2man: 155, 157
pod2texi: 200, 201
pod2text: 155, 157
pod2usage: 155, 157
podchecker: 155, 157
podselect: 155, 157
portablectl: 205, 209
post-grohtml: 181, 182
poweroff: 205, 209
pr: 172, 175
pre-grohtml: 181, 182
preconv: 181, 182
printenv: 172, 175
printf: 172, 175
prlimit: 215, 218
prove: 155, 157
prtstat: 142, 142
ps: 213, 214
psfaddtable: 191, 192
psfgettable: 191, 192
psfstriptable: 191, 192
psfxtable: 191, 192
pslog: 142, 142
pstree: 142, 142
pstree.x11: 142, 142
ptar: 155, 157
ptardiff: 155, 157
ptargrep: 155, 157
ptx: 172, 175
pwait: 213, 214

pwck: 128, 131
pwconv: 128, 131
pwd: 172, 175
pwdx: 213, 214
pwunconv: 128, 131
pydoc3: 168
python3: 168
ranlib: 118, 120
raw: 215, 218
readelf: 118, 120
readlink: 172, 175
readprofile: 215, 218
realpath: 172, 175
reboot: 205, 209
recode-sr-latin: 143, 144
refer: 181, 182
rename: 215, 218
renice: 215, 218
reset: 138, 140
resize2fs: 220, 222
resizepart: 215, 218
resolvconf: 205, 209
resolvectl: 205, 209
rev: 215, 218
rkfill: 215, 218
rm: 172, 175
rmdir: 172, 175
modprobe: 162, 163
roff2dvi: 181, 182
roff2html: 181, 183
roff2pdf: 181, 183
roff2ps: 181, 183
roff2text: 181, 183
roff2x: 181, 183
routef: 189, 190
routel: 189, 190
rtacct: 189, 190
rtcwake: 215, 218
rtmon: 189, 190
rtpr: 189, 190
rtstat: 189, 190
runcon: 172, 175
runlevel: 205, 209
runtest: 117, 117
rview: 202, 204
rvim: 202, 204
script: 215, 218
scriptreplay: 215, 218
sdiff: 178, 178
sed: 141, 141
seq: 172, 175
setarch: 215, 218
setcap: 127, 127
setfacl: 126, 126
setfattr: 125, 125
setfont: 191, 192
setkeycodes: 191, 192
setleds: 191, 192
setmetamode: 191, 192
setsid: 215, 218
setterm: 215, 218
setvtrgb: 191, 192
sfdisk: 215, 218
sg: 128, 131
sh: 147, 148
sha1sum: 172, 175
sha224sum: 172, 175
sha256sum: 172, 175
sha384sum: 172, 175
sha512sum: 172, 175
shasum: 155, 157
showconsolefont: 191, 192
showkey: 191, 192
shred: 172, 175
shuf: 172, 175
shutdown: 205, 209
size: 118, 120
slabtop: 213, 214
sleep: 172, 176
sln: 95, 100
soelim: 181, 183
sort: 172, 176
sotruss: 95, 100
splain: 155, 157
split: 172, 176
sprof: 95, 100
ss: 189, 190
stat: 172, 176
stdbuf: 172, 176
strings: 118, 120
strip: 118, 120
stty: 172, 176
su: 128, 131
sulogin: 215, 218
sum: 172, 176
swaplabel: 215, 218
swapoff: 215, 218
swapon: 215, 218
switch_root: 215, 218
sync: 172, 176

sysctl: 213, 214
systemctl: 205, 209
systemd-analyze: 205, 209
systemd-ask-password: 205, 209
systemd-cat: 205, 209
systemd-cgls: 205, 209
systemd-cgtop: 205, 209
systemd-delta: 205, 209
systemd-detect-virt: 205, 209
systemd-dissect: 205, 209
systemd-escape: 205, 209
systemd-hwdb: 205, 209
systemd-id128: 205, 209
systemd-inhibit: 205, 209
systemd-machine-id-setup: 205, 210
systemd-mount: 205, 210
systemd-notify: 205, 210
systemd-nspawn: 205, 210
systemd-path: 205, 210
systemd-repart: 205, 210
systemd-resolve: 205, 210
systemd-run: 205, 210
systemd-socket-activate: 205, 210
systemd-tmpfiles: 205, 210
systemd-tty-ask-password-agent: 205, 210
systemd-umount: 205, 210
tabs: 138, 140
tac: 172, 176
tail: 172, 176
tailf: 215, 219
talk: 153, 154
tar: 199, 199
taskset: 215, 219
tbl: 181, 183
tc: 189, 190
tclsh: 114, 115
tclsh8.6: 114, 115
tee: 172, 176
telinit: 205, 210
telnet: 153, 154
test: 172, 176
texi2dvi: 200, 201
texi2pdf: 200, 201
texi2any: 200, 201
texindex: 200, 201
tfmtodit: 181, 183
tftp: 153, 154
tic: 138, 140
timedatectl: 205, 210
timeout: 172, 176
tload: 213, 214
toe: 138, 140
top: 213, 214
touch: 172, 176
tput: 138, 140
tr: 172, 176
traceroute: 153, 154
troff: 181, 183
true: 172, 176
truncate: 172, 176
tset: 138, 140
tsort: 172, 176
tty: 172, 176
tune2fs: 220, 222
tzselect: 95, 101
udevadm: 205, 210
ul: 215, 219
umount: 215, 219
uname: 172, 176
uname26: 215, 219
uncompress: 187, 187
unexpand: 172, 176
unicode_start: 191, 192
unicode_stop: 191, 192
uniq: 172, 176
unlink: 172, 176
unlzma: 105, 106
unshare: 215, 219
unxz: 105, 106
updatedb: 180, 180
uptime: 213, 214
useradd: 128, 131
userdel: 128, 131
usermod: 128, 131
users: 172, 176
utmpdump: 215, 219
uuidd: 215, 219
uuidgen: 215, 219
uuidparse: 215, 219
vdir: 172, 176
vi: 202, 204
view: 202, 204
vigr: 128, 131
vim: 202, 204
vimdiff: 202, 204
vimtutor: 202, 204
vipw: 128, 131
vmstat: 213, 214
w: 213, 214
wall: 215, 219

watch: 213, 214
wc: 172, 176
wdctl: 215, 219
whatis: 196, 198
whereis: 215, 219
who: 172, 176
whoami: 172, 176
wipefs: 215, 219
x86_64: 215, 219
xargs: 180, 180
xgettext: 143, 144
xmlwf: 152, 152
xsubpp: 155, 157
xtrace: 95, 101
xxd: 202, 204
xz: 105, 106
xzcat: 105, 106
xzcmp: 105, 106
xzdec: 105, 106
xzdiff: 105, 106
xzegrep: 105, 106
xzfgrep: 105, 106
xzgrep: 105, 106
xzless: 105, 106
xzmore: 105, 106
yacc: 145, 145
yes: 172, 176
zcat: 187, 187
zcmp: 187, 187
zdiff: 187, 187
zdump: 95, 101
zegrep: 187, 187
zfgrep: 187, 187
zforce: 187, 187
zgrep: 187, 187
zic: 95, 101
zipdetails: 155, 157
zless: 187, 187
zmore: 187, 187
znew: 187, 188
zramctl: 215, 219
zstd: 107, 107
zstdgrep: 107, 107
zstdless: 107, 107

Bibliothèques

Expat: 158, 158
ld-2.33.so: 95, 101
libacl: 126, 126
libanl: 95, 101
libasprintf: 143, 144
libattr: 125, 125
libbfd: 118, 120
libblkid: 215, 219
libBrokenLocale: 95, 101
libbz2: 103, 104
libc: 95, 101
libcap: 127, 127
libcheck: 177, 177
libcom_err: 220, 222
libcrypt: 95, 101
libcrypto.so: 166, 167
libctf: 118, 120
libctf-nobfd: 118, 120
libcursesw: 138, 140
libdbus-1: 211, 212
libdl: 95, 101
libe2p: 220, 222
libelf: 164, 164
libexpat: 152, 152
libexpect-5.45: 116, 116
libext2fs: 220, 222
libfdisk: 215, 219
libffi: 165
libfl: 113, 113
libformw: 138, 140
libg: 95, 101
libgcc: 132, 136
libgcov: 132, 136
libgdbm: 150, 150
libgdbm_compat: 150, 150
libgettextlib: 143, 144
libgettextpo: 143, 144
libgettextsrc: 143, 144
libgmp: 121, 122
libgmpxx: 121, 122
libgomp: 132, 136
libhistory: 109, 110
rmmod: 162
liblsan: 132, 136
libltdl: 149, 149
liblto_plugin: 132, 136
liblzma: 105, 106
libm: 95, 101
libmagic: 108, 108
libman: 196, 198
libmandb: 196, 198
libmcheck: 95, 101
libmemusage: 95, 101
libmenuw: 138, 140

libmount: 215, 219
libmpc: 124, 124
libmpfr: 123, 123
libncursesw: 138, 140
libnsl: 95, 101
libnss: 95, 101
libopcodes: 118, 120
libpanelw: 138, 140
libpcprofile: 95, 101
libpipeline: 193
libprocps: 213, 214
libpsx: 127, 127
libpthread: 95, 101
libquadmath: 132, 136
libreadline: 109, 110
libresolv: 95, 101
librt: 95, 101
libSegFault: 95, 101
libsmartcols: 215, 219
libss: 220, 222
libssl.so: 166, 167
libssp: 132, 136
libstdbuf: 172, 176
libstdc++: 132, 136
libstdc++fs: 132, 136
libsupc++: 132, 136
libsystemd: 205, 210
libtcl8.6.so: 114, 115
libtclstub8.6.a: 114, 115
libtextstyle: 143, 144
libthread_db: 95, 101
libtsan: 132, 136
libubsan: 132, 136
libudev: 205, 210
libutil: 95, 101
libuuid: 215, 219
liby: 145, 145
libz: 102, 102
libzstd: 107, 107
preloadable_libintl: 143, 144

Scripts

clock
  configuration: 233
console
  configuration: 234
hostname
  configuration: 228
localnet
  /etc/hosts: 229
network
  /etc/hosts: 229
  configuration: 226
network
  /etc/hosts: 229
  configuration: 226
dwp: 118, 119

Autres

/boot/config-5.10.17: 245, 249
/boot/System.map-5.10.17: 245, 249
/dev/*: 72
/etc/fstab: 243
/etc/group: 75
/etc/hosts: 229
/etc/inputrc: 237
/etc/ld.so.conf: 99
/etc/lfs-release: 253
/etc/localtime: 98
/etc/lsb-release: 253
/etc/modprobe.d/usb.conf: 248
/etc/nsswitch.conf: 98
/etc/os-release: 253
/etc/passwd: 75
/etc/protocols: 94
/etc/resolv.conf: 228
/etc/services: 94
/etc/vimrc: 203
/run/utmp: 75
/usr/include/asm-generic/*.h: 47, 47
/usr/include/asm/*.h: 47, 47
/usr/include/drm/*.h: 47, 47
/usr/include/linux/*.h: 47, 47
/usr/include/misc/*.h: 47, 47
/usr/include/mtd/*.h: 47, 47
/usr/include/rdma/*.h: 47, 47
/usr/include/scsi/*.h: 47, 47
/usr/include/sound/*.h: 47, 47
/usr/include/video/*.h: 47, 47
/usr/include/xen/*.h: 47, 47
/var/log/btmp: 75
/var/log/lastlog: 75
/var/log/wtmp: 75
/etc/locale.conf: 236
/etc/shells: 239
man pages: 93, 93
Personnalisation de Systemd: 239